Vol.2014-SE-185 No.9
2014/7/10

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

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

Abstract: ソフトウェア開発は複数の開発者がチームを組んで，設計や実装，開発スケジュールなどの議論や検討を繰り返し，開発プロセスを進めていく．そのため，人と人とのコミュニケーションが欠かせない活動である．コミュニケーションには，開発経験や得意分野，人間関係などあらゆる人間的要因が関係する．それにも関わらず，開発経験以外の要因は，プライバシーの観点や，そもそも分析する手段が確立されていないことにより，考慮されていない．つまり，ソフトウェア開発の最適なチームや開発組織としての能力成熟といった課題はこれまで見過ごされてきた．そこで，本研究の目的は，ソフトウェア開発者の様々な特性を，プロジェクトマネージャの評価表や SVN と Trac の開発履歴を，より多面的にかつ実証的に明らかにすることである．本稿では大学院生を対象とした PBL 合宿に焦点をあて，開発者の様々な特性を分析する．

Keywords: 開発履歴，風林火山モデル，開発者の特性，リポジトリマイニング

1. はじめに

ソフトウェア開発では，複数の開発者がそれぞれの役割を遂行し，かつ，互いにコミュニケーションを取らなければならない．また，開発者は人間であるため，開発において，得意なタスクと，不得意なタスクがあるのは当然である．また，開発者の得意分野がわかっていれば，より効率的なチーム編成が可能となる．一般的には，互いに得意な分野が異なっていれば，開発者が相互にタスクを補い合えるため，理想的なチームの 1 つとなる．また，一方で，プロトタイプ開発や，リファクタリング開発など，特定の目的の開発においては，その目的に合う開発者を割り当てたい．このことから，開発者の得意・不得意を踏まえたチーム編成が行えると，より良い開発に繋がると考えられる．

従来，チーム編成は，スキルレポートとそれまでの人間関係に基づいて管理者が行っていた．スキルレポートとは，プログラミング言語の経験や，プロジェクトの経験など，それまでの開発者の経歴と，所有資格など能力を客観的に判断するための指標が記載されている文書を指す [1]．スキルレポートは，開発者の経験を記したものであり，開発者の能力を表したものではない．また，開発の様々なタスクに対応付けられるような詳細が記載されていないことも多い．すなわち，チーム編成のための目安ではあるものの，指標としては扱いづらい．

また，従来，プログラマの性格診断のような判断基準がブログなどで公開されている．例えば，10 種類に分類する方法 [2] や，6 種類に分類する方法 [3] などがあり，開発者には様々な特性が必要であると直感的に認識されている．ただし，これらは主観による判断であるため，科学的な根拠はない．更に，興味を持った者により，分類結果が追加される場合があるため，分類結果が 1 つに定まらず信頼できる判断基準になり得ない．

一方で，人間関係に基づく判断によるチーム編成も，根拠が薄弱である．客観的な指標ではなく，個人の主観に基づいた判断であるためである．従来のチームの編成方法は定性的な，もしくは主観に基づいた判断であることが多い．そして，より効率的なチーム編成を目指す場合は，客観的な指標に基づいた判断が必要となる．

そこで，我々は開発中に得られたデータを元に開発者の能力を測定するモデルを提案する．能力モデルとして，開発者の能力を風林火山という 4 つのタイプに分ける [4] モデルを採用する．提案する風林火山モデルは，この 4 タイプそれぞれに，開発ログから得られたメトリクスを適切に割り当て，それぞれの能力を測定するものである．我々は予備調査として，学部 3 年生の演習を対象に風林火山モデルを当てはめ，特性の自動計測の可能性を議論した [5]．本稿は，大学院生を対象に行ったプロジェクトに対して，風林火山モデルを適用し，有効性を確認した．

2. 関連研究

角田らは，ソフトウェア開発において人間的側面が与える影響についての調査を行った [6]．調査は開発メンバのパーソナリティ（性格）に特に着目して行われた．調査結果によると，ソフトウェア開発において，チームメンバの性格は無視できない要因であると述べられている．ただし，(1) 調査方法がソフトウェア工学に特化していないこと，(2) アンケートでの調査であることの 2 点が課題として提示されている．(1) は，他の分野に応用できる反面，チーム編成のためには，より効率的な方法の可能性がある．(2) は，客観的指標に基づい

[1] Division of Frontier Informatics, Graduate school of Kyoto Sangyo Univercity , Kyoto, Japan
[2] Faculty of Computer Science and Engineering, Kyoto Sangyo Univercity, Kyoto, Japan
[3] Graduate School of Information Science, Nara Institute of Science and Technology, Kyoto, Japan
[4] Department of Informatics, Kinki Univercity, Kyoto, Japan
[5] Graduate School of Information Science and Technology, Osaka University

てチーム編成を行いたいにもかかわらず，アンケートという個人の主観に基づいた指標を元にしている点が問題である．

Onoue らは，アクティブなソフトウェア開発において，どのようなタイプの開発者がいるのかを調査した [7]．オープンソースソフトウェア開発での開発者の活動から能力診断をすることに注目し，GitHub の node[*1] と jQuery[*2] プロジェクトを対象に調査を行った．その結果，それぞれの開発者の能力によって行動が異なることが明らかになった．

3. 準備

3.1 開発者の能力測定

3.1.1 開発者の能力

良いチーム編成のためには，開発者の得意分野などの向き，不向きを知っていることが望ましい．そして，従来開発者がどのようなタイプであるかの分類方法は，過去に数多く提案されている．例えば，10 種類に分類する方法 [2] や，6 種類に分類する方法 [3] などがある．ただし，これらの分類方法をソフトウェア開発のチーム編成に用いるには，以下の 2 点において，問題がある．まず第一に，これらの分類のカテゴリ数は増減する可能性があることが挙げられる．例えば，新たな分類を公開したとき，興味を持った第三者が新たな分類を追加することがあるためである．このようなことが起こると，どの段階の分類法が信頼できるものか不明となり，根拠のある分類方法ではなくなる．第二の問題は，開発者がどのような分類になるのかの判断が，個人の感覚に委ねられている点である．つまり，どの分類に当てはまるのか判断する者が異なれば，異なる結果になり得る．また，その分類の根拠も感覚的なものであることが多いため，チーム編成の判断材料には採用し難い．

我々は，客観的な指標に基づいたチーム編成のために，ソフトウェア開発のエンピリカルなデータを元に開発者の能力を診断するシステムの構築を目指す．更に，分類後のカテゴリ数が第三者により追加や削除が行われないような分類法が必要である．これにより，スキルレポートに比べ，より細かな粒度での測定が可能になり，客観的な指標に基づいたチーム編成が可能になる．

3.1.2 風林火山モデル

第 3.1.1 節で述べたように，能力分類手法として，(1) 分類のカテゴリ数が固定され，追加・削除が行われないこと，(2) エンピリカルデータに基づいた判断を行うことの 2 点が必要である．

そこで，小野が提案している開発者の風林火山モデルに着目する [4]．このモデルは開発者には，風林火山の 4 属性に分けられた能力がそれぞれ必要であるという，能力の分類法である．それぞれの属性は表 1 に示す特徴を持つ．風は実装能力を表し，林はプロジェクト管理能力を示している．火はプロダクトとプロセスの価値を向上させる能力，そして，山はプロダクトの品質を向上させる能力を表している．どれもが開発者にとって必要な能力であるため，4 つの属性を兼ね備える開発者が理想的な開発者であると言える．

また，この風林火山は，追加されること，削除されることもないと考えられる．例えば，[2] では，カテゴリが職業で表されているため，新たなカテゴリを追加することは非常に容易である．しかし，風林火山は 4 つのカテゴリに限定されており，カテゴリ数は固定される．

Table 1 風林火山モデル

分類	特徴
風	迅速な設計・実装によってチームを加速させる．
林	突発的なトラブルに冷静に対処し，チームに乱れぬペースを提供する．
火	新しい技術・方法・ツールの積極的な導入によって，チームの成果物の競争力を高める．
山	厳密なエラー・チェックと堅牢なプログラミングによって成果物の安定性を高める．

そこで，我々は，第 2 の問題点を解決するため，開発データを使って，風林火山モデルによる開発者の能力測定手法を提案する．

3.2 対象プロジェクト

3.2.1 対象プロジェクトの概要

本稿で対象とするプロジェクトについて述べる．本稿で対象とする開発プロジェクトは，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ある．この教育プログラムは，神戸大学と大阪大学が中心となって行っている．そして，神戸大学，大阪大学に加え，7 大学の大学院 1 年次生を対象に，以下の 3 点を考慮して開発されている [8]．

- 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．

この教育プログラムにおいて，1 週間の間に集中的に開発を行うプロジェクトを実施した．仕様は与えられ，1 チーム 5, 6 人体制で，Web アプリケーションを開発するものである．このプロジェクトを本稿で得られた開発データの分析対象とした．

3.2.2 対象プロジェクトの進め方

対象プロジェクトは，Scrum[9] を適用したアジャイル開発を行った [10]．Scrum では，スプリントと呼ばれる短い期間に，1 つのまとまった単位の開発を行う．そして，スプリントの最初に計画を立て，最後に振り返りを行う．本来の Scrum であれば，1 スプリントは 1 週から 4 週程度であるが，対象プロジェクトは 4 日間の開発であったため，1 日を 1 スプリントとしている．ただし，この開発合宿に先立ち，1 日だけ練習としてスプリントを実行している．そのため，合計 5 スプリントを実行したことになる．

各チームは，いずれのスプリントにおいても，最初に開発するコンポーネントを決め，見積もり後，役割分担の上開発に臨む．役割は，プロダクトオーナー 1 名，スクラムマスター 1 名，そして残りのメンバが開発者の計 3 役である．スクラムマスターはチームが円滑に開発できることを最優先事項としているため，必ずしも開発を行うわけではない．

また，開発を進める上で，Trac と Subversion を使った TiDD[11] を採用しており，各チケットに，見積もり時間，開始・終了時刻，所要時間を記録した．チケットにはそのチケットの性格を表すチケットタイプが付随している．チケットタイプは，ソースコード作成 (Java)，ソースコード作成 (HTML, JavaScript)，テストコード作成，レビュー，結合テスト作成，結合テスト実施，バグ修正の 7 種類に分けられる．

各チームに CI ツールとして，Jenkins[*3] を用意しており，Subversion へのコミット毎に Jenkins でのフルビルドが行われるようになっている．

[*1] https://github.com/joyent/node
[*2] https://github.com/jquery/jquery
[*3] http://jenkins-ci.org/

Table 2 収集したメトリクスと収集方法

ID	メトリクス	概要	取得方法
M1	スクラムマスターランク	メンバからのスクラムマスターとしての評価 (５段階) の平均	アンケート
M2	スプリント別の UC 目標達成率	実装した UC 数 (実績) / 目標となる UC 数	Trac，アンケート
M3	個人の総実装行数の割合	個人の総実装行数/チームの総実装行数	Subversion
M4	実装行数／時間	単位時間辺りの実装行数	Subversion
M5	個人の成長指数	(4th スプリントコミット数-1st スプリントコミット数)/4th コミット数	Subversion
M6	チケット対応コミットの割合	チケットとコミットが対応しているコミット数/全コミット数	Trac
M7	レビュー回数	チケットタイプがレビューであるチケットの解決数	Trac
M8	レビュー時間／回	レビュー時間/レビュー回数	Trac
M9	バグ修正回数	チケットタイプがバグであるチケットの解決数	Trac
M10	バグ修正時間／回	バグ修正時間/バグ修正回数	Trac
M11	Jenkins ビルド成功率	Jenkins でのビルド成功数/Jenkins でのビルド数	Jenkins
M12	Eclipse ビルド成功率	Eclipse でのビルド成功数/Eclipse でのビルド数	Eclipse

3.2.3 収集したメトリクス

対象プロジェクトでは，Trac, Subversion, Jenkins, Eclipse を利用しており，すべて開発ログを収集している．また，一方で，スプリント終了後，アンケートに回答してもらい，スクラムマスターの評価も行っている．そこで，表 2 に示す 12 のメトリクスが，個人毎に収集できるようになっている．

スクラムマスターランクと，スプリント別の UC(ユースケース) 目標達成率は，各スプリントにメトリクス値が導出される．しかし，各スプリントにスクラムマスターが異なる．そのため，当該スプリントでのスクラムマスターを担当した個人のメトリクス値とする．

3.2.4 対象プロジェクトの教育上の制約

対象プロジェクトにおいては，教育上の観点から様々な制約を設けている．制約がなければ，ソースコード編集に多くの時間が費やされ，目的の１つであるソフトウェア工学に基づいた PBL が達成できない．同じく，制約がなければ，徹夜での作業を行うチームも起こり得る．更に，少数のプログラムを得意とする学生だけがアプリケーションを開発し，そうでない学生が何もしない時間を持つことも考えられる．このような理由から，対象プロジェクトにおいては，時間的な制約とタスク割り当ての制約を設けた．

時間的な制約は，開発時間は午前 10 時半から，午後 6 時までと決めたことである．この時間以外には打ち合わせしか認められていない．また，タスク割り当ての制約は，アサインメント制約と呼んでおり，次の通りである．

A. 開発の期間中ですべての役割 (プロダクトオーナー，スクラムマスター，開発者) を担当すること．
B. ソースコード作成（Java），ソースコード作成（HTML, JavaScript），テストコード作成，レビューの各メンバの担当数がチーム平均値の 0.8 倍から 1.2 倍以内に収まること．

このアサインメント制約はチームの自己組織化や個人の能力の向上，また，チームとしての成長を狙ったものである．例えば，スクラムマスターはチームを健全に保つという，Scrum を実践する上で不可欠な要素である．また，プロダクトオーナーも，チームに明示的な支持をせず，チームの方向性を決めるという，非常に難しい役割が求められている．そこで，短い期間ではあるが，実際に行ってみることで，それぞれの学生が，自らの感覚でそれぞれの役割を掴むことを狙っている．つまり，同じチームで 5 スプリントを実施するため，各スプリントでスクラムマスターとプロダクトオーナーが異なることになる．

Table 3 メトリクスの風林火山モデルへの割り当て

分類	メトリクス
風	実装行数／時間，個人の総実装行数の割合
林	チケット対応コミットの割合，個人の成長指数
火	スクラムマスターランク，スプリント別の UC 目標達成率
山	レビュー回数，レビュー時間／回，バグ修正回数，バグ修正時間／回，Jenkins ビルド成功率，Eclipse ビルド成功率

Table 4 ある受講生のメトリクスの計測値とその順位

ID	メトリクス	実測値	順位
M1	スクラムマスターランク	3.77	19
M2	スプリント別の UC 目標達成率	0.50	7
M3	個人の総実装行数の割合	0.24	9
M4	実装行数／時間	120	13
M5	個人の成長指数	0.64	19
M6	チケット対応コミットの割合	0.58	48
M7	レビュー回数	10	30
M8	レビュー時間／回	18 分	40
M9	バグ修正回数	5	26
M10	バグ修正時間／回	11 分	23
M11	Jenkins ビルド成功率	0.86	8
M12	Eclipse ビルド成功率	0.86	27

3.3 風林火山モデルの適用

3.3.1 風林火山モデルのフレームワーク

風林火山モデルのフレームワークを図 1 に示す．図 1 に示す通り，まず，利用する開発ツールやアンケートから，様々なメトリクスを収集する．次に，それらを風林火山モデルに従って，それぞれの属性に割り当てる．

どのメトリクスをどの属性に割り当てるか，割り当て後，各属性にどのように計測結果を導出するかは，プロジェクトに最適化させる必要がある．本稿においては，どのメトリクスをどの属性に割り当てるかは，第一著者の主観により決めた．また，割り当て後，各属性の値を算出する方法として，今回はまず各メトリクスを 49 名の順位に変換した．その後，各属性に割り当てたメトリクスの順位の平均値を属性の値とした．

3.3.2 メトリクスの風林火山モデルへの対応付け

本節では，収集したメトリクスそれぞれを，風林火山で表

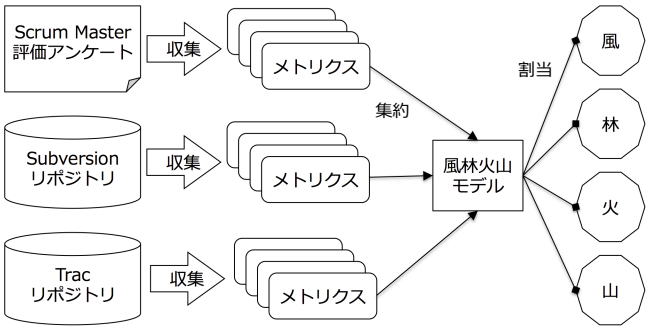

Fig. 1 風林火山の概観図

した能力モデルに対応付ける．これにより，風林火山のそれぞれの値が算出できるようになる．なお，対応付けには，以下の点を考慮して筆者の主観により割り当てた．割り当てを表 3 に一覧で示す．

風には，迅速な設計・実装という風の特徴から，開発速度や納期に関連するメトリクスを割り当てる．具体的には，実装行数／時間，個人の総実装行数の割合である 2 つのメトリクスを割り当てた．林は，チームに乱れぬペースを提供することから，チームがしっかりと運営されているか，決められたルールの遵守しているかを表すメトリクスを割り当てる．そのため，チケット対応コミットの割合，個人の成長指数の 2 つのメトリクスを割り当てた．火は，チームの成果物の競争力を高めることから，より良い成果物を生み出すため，チームのタスク遂行力を高めるメトリクスを割り当てる．具体的にはスクラムマスターランク，スプリント別の UC 目標達成率を割り当てた．山は，成果物の安定性を高めることから，成果物の品質を高めるために必要なメトリクスとした．品質向上のために行うレビュー関連のメトリクス，バグ修正関連のメトリクス，そして，ビルド成功率を割り当てた．

例えば，ある受講生のメトリクス値とその順位は表 4 に示す通りである．なお，順位は実測値を昇順に並べたときの順番であり，実測値が一番良い場合は 1 となる．この受講生の風林火山の値は，それぞれ，風が (M3 + M4)/2 = 11，林が (M5 + M6)/2 = 33.5，火が (M1 + M2)/2 = 13，そして，山が (M7 + M8 + M9 + M10 + M11 + M12)/6 = 25.67 である．この受講生の風林火山の結果をレーダーチャートに表した図が図 2 である．

ただし，この割り当ては，主観によるものであるため，より良い割り当て方法は今後の課題としたい．

4. ケーススタディ

4.1 風林火山モデルを用いた個人の能力測定

対象プロジェクトの受講者 49 人のメトリクスを取得し，風林火山メトリクスに割り当てた結果を図 3 に示す．図 3 では，受講生に 1〜49 までの ID を付けて一覧にしている．

第一著者の受けた印象と，各受講生の能力を表すグラフは概ね一致した．特に，14, 25 は風火山に高い値を示しており，林が低くなっている．この 2 名は，レーダーチャートでは上向き三角形の形状となっている．この風林火山のグラフから読み取れるのは，高い品質のソフトウェアを迅速に実装し，また，高いタスク遂行能力を有している．しかし，その反面，チーム運営は苦手ということである．彼らの実際の印象は，ミーティングでは相手の意見を鵜呑みにせず，慎重かつ積極的に意見を交わす行動が見られた．しかし一方で，自らの意見に固執する一面も見られた．このことから，全体的に高い能力をもつものの，チーム運営能力は十分ではないという，グラフから読み取れる内容と一致している．

19 は山と林のみが高い値を示しており，他の属性は低くなっている．この受講生は慎重で静かな性格であり，その反面，コツコツと物事を着実にこなす印象であった．また，積極的に回りに声をかけることは少なく，風林火山の結果も印象通りの結果となった．

また，10, 37 は風林火山の各属性それぞれに高い値を示しており，どの役割においてもしっかりと仕事を果たす人物であるとグラフからは読み取れる．彼らの実際の印象は，プロダクトに対するコーディングやテストのタスク，プロジェクトに対する適切な指示や管理のタスクをそつなくこなす，オールマイティーな印象を受けた．つまり印象通りの結果であった．

4.2 風林火山モデルを用いたチームの能力測定

チームごとに所属メンバの風林火山の値の平均を取ったところ，すべてのチームにおいて，全属性値が 22 ± 4 の範囲に収まった．これはつまり，どのチームも風林火山のバランスが取れていると言える．それを裏付けるように，実装を完了したユースケース数（実績）も突出して多かったチーム，少なかったチームは存在せず，5 ± 2 の範囲に収まった．

各チームの風林火山のバランスはとれているにも関わらず，UC 実績で差が出た理由を直接的に納期に関わる風と山で分析をした．正当な比較を行うためにアサインメント制約を 60%以上守った 3 チームに絞った．ユースケースを 5 つ完成させた

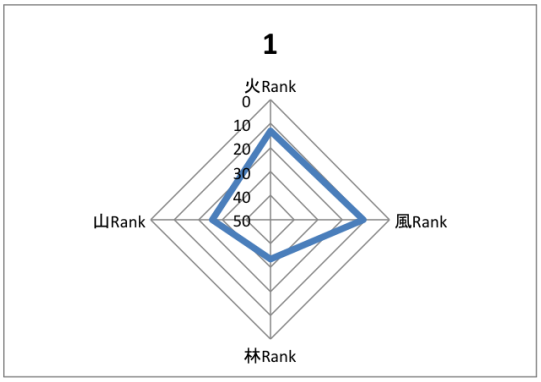

Fig. 2 ある受講生の風林火山グラフ

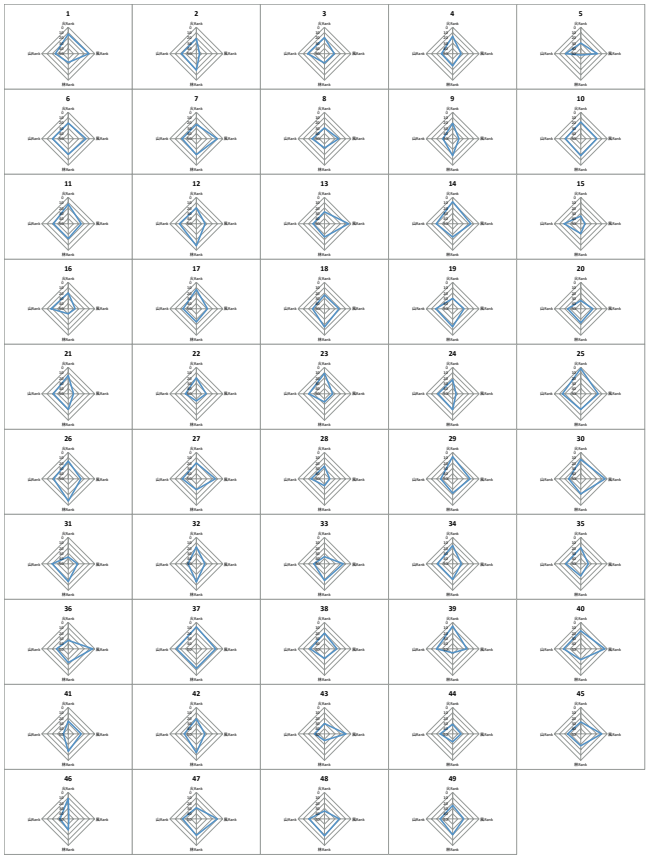

Fig. 3 風林火山モデルの適用結果

Table 5 全てのメトリクス間における相関係数

		M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
M1.	**スクラムマスターランク**	1.00	0.07	-0.16	-0.01	-0.15	0.11	-0.09	-0.05	0.05	-0.14	0.12	0.22
M2.	**スプリント別の UC 目標達成率**		1.00	0.19	0.10	-0.12	-0.02	-0.18	-0.04	0.14	-0.09	-0.11	0.05
M3.	**個人の総実装行数の割合**			1.00	0.42	-0.08	0.09	0.04	-0.02	0.22	-0.04	0.06	0.06
M4.	**実装行数／時間**				1.00	-0.16	0.11	-0.01	-0.14	0.07	-0.14	-0.35	0.05
M5.	**個人の成長指数**					1.00	-0.19	-0.32	0.19	-0.46	0.03	0.27	-0.12
M6.	**チケット対応コミットの割合**						1.00	-0.09	-0.01	0.36	-0.02	-0.17	-0.06
M7.	**レビュー回数**							1.00	-0.19	0.36	0.00	-0.07	0.06
M8.	**レビュー時間／回**								1.00	-0.33	0.22	0.11	-0.18
M9.	**バグ修正回数**									1.00	-0.08	-0.20	0.16
M10.	**修正時間／回**										1.00	-0.09	-0.38
M11.	**Jenkins ビルド成功率**											1.00	-0.03
M12.	**Eclipse ビルド成功率**												1.00

Table 6 チームの風林火山の属性値

	Average				Minimum			
	風	**林**	**火**	**山**	**風**	**林**	**火**	**山**
チーム A	26	30	21	23	11	19	13	18
チーム B	21	28	18	21	4	13	7	13
チーム C	24	23	26	26	11	18	12	21

チーム A とチーム B，そして，3 つ完成させたチーム C であった．結果を表 6 に示す．風の平均値と最小値 (ランキングであるため低いほど良い) は A, B, C に大きな差は無かった．しかし，山の平均値と最小値はチーム C はチーム A, B に劣っていた．山の平均値も最小値も低いということは，チームとしても個人としても，プロダクトの品質保持能力は，チーム A, B の方がチーム C より優れていたと言える．最終的に UC を終了させるには，教員による受け入れテストをパスする必要がある．そのため，山が低いため，納品できる状態に到達するまでに時間を要し，その結果，UC 実績の差につながったと考えられる．

5. 評価実験

本章では，風林火山モデルの妥当性を以下の 2 つの観点で評価した．

1. 利用メトリクスは妥当か．
2. 各メトリクスの割り当ては妥当か．

理想的には，能力診断には互いに関連のないメトリクスを利用することが望ましい．そのため，利用したメトリクス間の相関を調べ，利用メトリクスの妥当性を実験 1 にて調査した．

一方で，利用するメトリクスを風林火山のどの属性に分類するかの評価も重要である．実験 2 では，風林火山への割り当てを評価した．

5.1 実験 1: メトリクス間の相関

本手法は，風林火山の 1 つの属性を算出するために単一のメトリクスではなく，複数のメトリクスを用いる．複数のメトリクスを用いた方が外れ値への対応や，平準化が期待できるためである．そして，理想的には，風林火山各モデルの属性それぞれに割り当てるメトリクスは互いに独立していることが望ましい．似たような関係であれば，一方で良いと言えるためである．

そこで，本実験では，計測したすべてのメトリクスの実測値に対して，互いに相関関係があるかを調査した．対象となるメトリクスは表 3 で示した全メトリクスである．相関分析にはピアソンの相関係数を用いた．

実験結果として，計測したすべてのメトリクス間の相関を表 5 に示す．$|r| > 0.3$ の優位な相関関係が見られたペアのセルは網掛けをしている．有意な相関関係が得られたのは 8 つのペアであった．有意な相関関係を持つペアを表 7 に示す．メトリクス 1 とメトリクス 2 の間に r の相関係数 ($p < 0.05$) があったことを示している．なお，各メトリクスに，風林火山のどの属性に対応するのか括弧内に示している．

Table 7 相関関係のあったメトリクス

	メトリクス 1	**メトリクス 2**	r
a.	実装行数／時間 (風)	個人の実装行数の割合 (風)	0.42
b.	バグ修正回数 (山)	レビュー回数 (山)	0.36
c.	バグ修正回数 (山)	チケット対応コミットの割合 (林)	0.36
d.	レビュー回数 (山)	個人の成長指数 (林)	-0.32
e.	バグ修正回数 (山)	レビュー時間／回 (山)	-0.33
f.	実装行数／時間 (風)	Jenkins ビルド成功率 (山)	-0.35
g.	バグ修正時間／回 (山)	Eclipse ビルド成功率 (山)	-0.38
h.	バグ修正回数 (山)	個人の成長指数 (林)	-0.46

ここで，風林火山の同タイプ間で相関があるものは，a, b, e, g の 4 つのペアである．a, b は正の相関であるため問題にはならないものの，e, g は負の相関が見られた．同タイプへ割り当てるメトリクス間で負の相関があると，一方のメトリクスが高く，もう一方が低い値になる場合が往々にして起こることになる．この場合，両者を含むメトリクスで平均を取ると，期待した値にならない可能性が高い．この問題点への対応は今後の課題としたい．ただし，対象プロジェクトで採用したアサインメント制約が，このような相関に繋がったとも考えられる．

一方で，c, d, f, h の 4 つのペアでは，風林火山の異なる属性に用いるメトリクス間での相関が見られた．c, d, h では，山と林に割り当てたメトリクス間に負の相関が見られた．c はバグ修正が多い人は，チケット対応コミットの割合が低いもしくはその逆であると言える．これは，一部の受講生に TiDD が十分に浸透していなかったためと考えられる．d, h は，レビュー回数が多い人もしくはバグ修正回数が多い人は，成長指数が低い，すなわち，最初のスプリントでのコミット回数が最後のスプリントのコミット回数よりも多かったことを表す．この理由として，スプリントが進むにつれ，チーム全体でアサインメント制約を守るようになったため，もしくは，スプリントが進むにつれ，コミットを慎重に行うようになったと考えられる．これらの相関は，別のプロジェクトでも同様の結果となるか追加実験で確認したい．

f では，速くコードを書く人は，Jenkins でのビルドで失敗が多い，もしくは，Jenkins でのビルドで成功が多い人は，コードを書く速度が遅いということを示している．これは，一般的にコードを速く書く人はコードを書く機会が多く，その分，Jenkins でのビルド回数も多くなる．そのため，ビルド成功率が低くなっているものと思われる．同じように，風属性と山

Table 8 実験 2: 風林火山に割り当てたメトリクスの妥当性評価

属性	m_i: 取り除いたメトリクス	M_i: v_i を算出するためのメトリクス	r
風	実装行数/時間	個人の実装行数の割合	0.37
林	チケット対応コミットの割合	個人の成長指数	-0.27
火	ScrumMaster ランク	スプリント別の UC 目標達成率	0.07
山	レビュー回数	レビュー時間/回, バグ修正回数, バグ修正時間/回 , Jenkins ビルド成功率, Eclipse ビルド成功率	-0.06
	レビュー時間/回	レビュー回数, バグ修正回数, バグ修正時間/回 , Jenkins ビルド成功率, Eclipse ビルド成功率	-0.25
	バグ修正回数	レビュー回数, レビュー時間/回, バグ修正時間/回 , Jenkins ビルド成功率, Eclipse ビルド成功率	0.10
	バグ修正時間/回	レビュー回数, レビュー時間/回, バグ修正回数 , Jenkins ビルド成功率, Eclipse ビルド成功率	-0.01
	Jenkins ビルド成功率	レビュー回数, レビュー時間/回, バグ修正回数 , バグ修正時間/回, Eclipse ビルド成功率	-0.16
	Eclipse ビルド成功率	レビュー回数, レビュー時間/回, バグ修正回数 , バグ修正時間/回, Jenkins ビルド成功率	-0.04

属性も負の相関があると直感的に思われる．そのため，この相関は不自然ではないと考えられる．

5.2 実験 2: 風林火山モデルの評価

ここでは，各メトリクスの割り当てが妥当であるかを検証する．検証は各属性値について，次の手順で行う．

(1) 属性値を v とする．
(2) 属性値を計算するためのメトリクス群 M から 1 つのメトリクス $m_i(1 \leq i \leq n)$ を取り出す．
(3) m_i 以外のメトリクス群 M_i で属性値 v_i を計算する．
(4) 各メンバに算出された v_i と m_i の相関を取る．

なお，各属性値を計算するためのメトリクス群 M は表 3 に示している．実験結果を表 8 に示す．m_i 列に示したメトリクスが取り除いたメトリクスであり，r 列が相関係数である．相関はピアソンの相関係数を用いた．

風は，0.37 であり，やや正の相関がある結果となった．また，火については，0.07 とほとんど相関がない結果となった．これらの結果は風林火山それぞれの属性へのメトリクスの割り当てに問題がないことを示している．

一方，林は-0.27 と低いながらも負の相関が見られた．また，山のレビュー時間／回を除いたときの相関も-0.25 と負の相関が見られた．山は，バグ修正回数を除いた場合を除き，すべて負の相関となっている．これらは，割り当てるメトリクスが相応しくない可能性がある．なぜなら，負の相関であれば，値を打ち消し合う関係にあるためである．

そのため，林と山の属性へのメトリクスの割り当ては，何らかの手法を用いて適切に選択される必要がある．

6. まとめ

本稿では，開発者の能力を客観的に測定することを目的とし，風林火山モデルを提案した．これは，風，林，火，山で特徴付けされた 4 つの能力に，測定したメトリクスを割り当てて，それぞれの能力値を客観的に表すモデルである．理想的には，開発者は 4 属性すべての能力を兼ね備えることが望ましい．

ケーススタディとして，大学院生を対象とした Scrum の開発演習からメトリクス値を算出し，受講生 49 名の能力値を算出した．その結果，多くの受講生の能力は主観と合致したものとなった．

また，実験で，利用したメトリクスの妥当性とメトリクスの割り当ての妥当性を評価した．この実験結果で，一定の妥当性は見られたものの，より多くのメトリクスを利用することや，より良い割り当て方法の模索など今後の課題も明らかになった．

具体的な今後の課題として，(1) メトリクスの割り当て方法の改善，(2) 使用するメトリクスの洗い出し，(3) 負の相関関係にあるメトリクス同士の扱い，(4) 他プロジェクトへの適用，などが挙げられる．

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。

References

[1] Feigenspan, J., Kästner, C., Liebig, J., Apel, S. and Hanenberg, S.: Measuring Programming Experience, *20th International Conference on Program Comprehension (ICPC 2012)*, pp. 73–82 (2012). Passau, Germany.
[2] James, J.: 10 types of programmers you'll encounter in the field (2007). http://www.techrepublic.com/blog/10-things/10-types-of-programmers-youll-encounter-in-the-field/ (Last Access: 2014/1/21).
[3] Manager, C. P.: The 6 Types of Software Engineers: Identification, Care and Feeling (2008). http://crankypm.com/2008/08/the-6-types-of-software-engineers-identification-care-and-feeding/ (Last Access: 2014/2/20).
[4] 小野和俊：Developer のための 5 つの習慣 —日本をソフトウェア輸出大国にしていくために，デベロッパーサミット 2006 (2006).
[5] 藤原 新，五田篤志，玉田春昭，角田雅照，畑 秀明：ソフトウェア開発履歴を利用した風林火山モデルによる開発者の特性診断の試み，ソフトウェア工学の基礎 XX, 日本ソフトウェア科学会 FOSE2013，pp. 295–296 (2013).
[6] 角田雅照，玉田春昭，畑 秀明：ソフトウェア開発チームにおけるパーソナリティの影響に関する調査，SES2013 併設ワークショップ「開発マネジメントにおける産学の問題共有と連携強化」(2013).
[7] Onoue, S., Hata, H. and ichi Matsumoto, K.: A Study of the Characteristics of Developers' Activities in GitHub, *5th International Workshop on Empirical Software Engineering in Practice*, pp. 1–6 (2013).
[8] 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).
[9] Rising, L. and Janoff, N. S.: The Scrum software development process for small teams, *IEEE Software*, Vol. 17, pp. 26–32 (2000).
[10] Fukuyasu, N., Saiki, S., Igaki, H., Matsumoto, S. and Kusumoto, S.: A Case Study of Cloud-enabled Software Development PBL, *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 499–504 (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).