情報処理学会論文誌 Vol.56 No.2 701–713 (Feb. 2015)

アジャイルソフトウェア開発教育のためのチケットシステムを用いたプロジェクト定量的評価手法の提案

井垣 宏[1,a)] 福安 直樹[2,b)] 佐伯 幸郎[3,c)] 柗本 真佑[3,d)] 楠本 真二[1,e)]

受付日 2014年5月19日, 採録日 2014年11月10日

概要：ソフトウェア開発教育の一環として，Scrum や XP といったアジャイルソフトウェア開発を採り入れたチームによるソフトウェア開発演習を実施する大学が増加しつつある．一方で，演習によるアジャイルソフトウェア開発教育には複数の課題が存在する．代表的なアジャイル開発フレームワークの 1 つとして知られる Scrum では，プロジェクトを検査し，自己組織的にプロジェクトの状況に適応し続けることが求められる．しかしながら，チームでの開発経験が少ない受講生には，プロジェクトの状況がどうであるかを検査すること自体が困難である．また，受講生によるチーム開発では，担当するタスクの種類や量が偏りがちである．我々はこれらの課題の解決を目指し，チケット駆動開発と呼ばれる開発手法と Scrum フレームワークを組み合わせることにより，プロジェクトを定量的に評価する枠組みを構築した．実際に我々のプロジェクト定量評価の枠組みを用いてチームによるソフトウェア開発演習を実施し，複数の定量評価基準に基づいて，チームごとのプロジェクト評価を実施した．

キーワード：ソフトウェア工学教育，アジャイルソフトウェア開発，Scrum，PBL，プロジェクト定量評価

Quantitative Project Evaluation for Agile Software Development using Ticket Management System

Hiroshi Igaki[1,a)] Naoki Fukuyasu[2,b)] Sachio Saiki[3,c)] Shinsuke Matsumoto[3,d)] Shinji Kusumoto[1,e)]

Received: May 19, 2014, Accepted: November 10, 2014

Abstract: The universities which teach agile software development, such as Scrum and XP are increasing in number. On the other hand, there are some problems in agile software development exercise. The Scrum known as one of the leading agile software development framework requires inspection and adaptation of projects. However, it is difficult for students with few team software development experiences to inspect their projects. Moreover, the kind and quantity of the tasks which each student takes charge of tend to become imbalanced. In this paper, we propose a framework which combines a ticket management system and the Scrum for quantitative evaluation in team software development exercise. We conducted practical software development exercise with using our framework as a case study.

Keywords: software engineering education, agile software development, Scrum, PBL, quantitative project evaluation

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[2] 和歌山大学システム工学部
Faculty of Systems Engineering, Wakayama University, Wakayama 640–8510, Japan
[3] 神戸大学大学院システム情報学研究科
Graduate School of System Informatics, Kobe University, Nada, Hyogo 657–0013, Japan
[a)] igaki@ist.osaka-u.ac.jp
[b)] fukuyasu@sys.wakayama-u.ac.jp
[c)] sachio@carp.kobe-u.ac.jp
[d)] shinsuke@cs.kobe-u.ac.jp
[e)] kusumoto@ist.osaka-u.ac.jp

This work is based on an earlier work: Quantitative Assessment with Using Ticket Driven Development for Teaching Scrum Framework, in ICSE Companion 2014 Companion Proceedings of the 36th International Conference on Software Engineering, June, 2014 ACM, 2014. http://dx.doi.org/10.1145/2591062. 2591162

1. はじめに

近年，プロジェクト失敗のリスク軽減や生産性の向上等を目的とし，アジャイルソフトウェア開発を採り入れるIT企業が増加しつつある [21]．大学においてもアジャイルソフトウェア開発をカリキュラムに採り入れる事例が増加しつつある [10], [11], [15]．我々も代表的なアジャイルソフトウェア開発フレームワークの1つであるScrumに基づくチームソフトウェア開発を教育するカリキュラムを2013年4月より開始している [25]．Scrumは経験主義に基づき，反復的かつ漸進的な手法を用いて，予測可能性の最適化とリスク管理を行うことを目標としたプロセスフレームワークである [17]．そのフレームワークには，プロジェクトの見える化や見える化に基づく現状の検査および適応といったいくつかの重要なコンセプトがまとめられている．我々の目的は，受講生にScrumのコンセプトを理解してもらい，チームソフトウェア開発における各種プロセス・プロダクトに関する知識・経験を獲得してもらうことにある．

通常このような教育カリキュラムでは，受講生らがチームを組み，実際にアプリケーションを開発することで学習するPBL（Project-based Learning）と呼ばれる形式が採用されることが多い．ソフトウェア開発を対象としたPBL（以降SDPBLと呼ぶ）はチームソフトウェア開発を実際に経験することができるため，非常に有効な学習手段の1つである．一方でSDPBLの実施にはいくつかの課題が存在する．まず，守るべきプロセスやルールに対する受講生の理解不足があげられる．我々が想定するチームによるソフトウェア開発経験が乏しい受講生にとって，レビューや単体テストといった多様なプロセスの重要性を理解し，関連するルールを遵守し続けることは非常に困難である．

2つ目の課題として，プロジェクトを通して受講生が得る経験に偏りが生じることである [24]．SDPBLでは，スキルに偏りのある複数の受講生が集まってチームを構成し，ソフトウェアを開発する．そのため，実装経験が豊富な受講生のみが実装を行い，そうでない受講生は実装をしない，といった作業内容の偏りが発生することがある．

そこで我々はScrumに基づくSDPBLにおいて，これまで多くのSDPBLで計測されていた納期だけでなく，プロセスやルールの遵守状況やタスクの割当て状況を定量的に評価するための評価基準「QAD」と関連する複数のメトリクスを提案する．ここでQADはそれぞれQuality，Assignment，Deliveryの略であり，それぞれプロセスおよびプロダクトの品質，タスク割当て，納期に関する基準を示している．QADでは，チームによって開発された様々なコンポーネントのうち，レビューが実施されているコンポーネントの割合を示すQ_{RV}というQualityに関するメトリクスや，単体テストの実装という種別のタスクをチーム内でどの程度均等に分担しているかを表すA_{WUT}というAssignmentに関するメトリクス等が定義されている．メトリクスの計測には，チケットシステムと呼ばれる，誰が，いつ，何をしたかといったプロセス情報を受講生自身が記録するシステムを用いる．チケットシステムから算出された各種メトリクスをプロジェクトの進捗に合わせて受講生にフィードバックすることで，遵守すべきプロセスやタスク割当てを受講生自身が意識し，プロジェクトを進めることが可能となる．

以降では，2章でScrumフレームワークとScrumを前提としたSDPBLにおける課題について述べる．3章では我々が提案するSDPBLのための定量的評価手法を詳述し，4章では定量的評価手法に基づくSDPBLの設計について述べる．5章では3章で述べたメトリクスを4章の環境でどのように計測したかを具体的に述べている．6章では定量的評価手法を実際のSDPBLについて適用した結果について述べ，7章で考察を行い，8章でまとめとする．

2. 準備

本章では，我々が実施するScrumフレームワークおよびソフトウェア開発を対象としたPBLの概要と教育における課題について説明する．

2.1 ソフトウェア開発PBL

PBL（Project-based Learning）とは，期限が定められており，新たな価値の創造を目的としたプロジェクトと呼ばれる形式で，受講生が主体的にプロジェクト運用に必要な多様なスキルを獲得することを目的とした学習手法である [1]．我々は特に，ソフトウェア開発をともなうPBLのことをソフトウェア開発PBL（以降SDPBL）と呼んでいる．SDPBLはその目的に応じて多様な形態で実施されている．沢田ら [26] は飛行船制御ソフトウェアを対象とし，仕様の分析から受け入れテストまでの一連の流れを体験するSDPBLを実施している．松澤ら [27] はPMを目指す企業の技術者と大学生の協同によるSDPBLを実施している．

我々が対象とするSDPBLは，Webアプリケーション開発に必要なプロダクトスキルとScrumフレームワークに基づくチーム開発に求められるプロセススキルを受講生に獲得させることを目的としている．そのため，要求分析や設計以降の実装，単体テスト，結合テストといった工程を対象としてプロジェクトを設計している．

2.2 Scrumフレームワーク

Scrumは1990年台初頭から複雑なプロダクト開発の管理に使用されてきた経験主義を基本とするプロセスフレームワークである [17]．Scrumは，反復的かつ漸進的な手法を用いて，予測可能性の最適化とリスク管理を行うことを目的としており，その実現のためのコンセプトとして，以下に示す透明性・検査・適応が重要視されている．

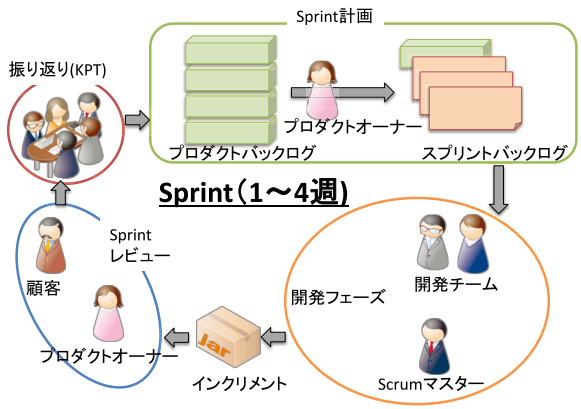


図 1 Scrum プロジェクトにおけるイベントフロー
Fig. 1 An event flow of a Scrum project.

透明性 プロセスおよびプロダクトが見える化されており，ステークホルダ間で共通理解を持つこと

検査 見える化された Scrum プロジェクトの成果物や進捗を頻繁に確認し，変化を検知すること

適応 プロセスの不備が許容値を超えていないかを検査によって判断し，必要に応じてプロセスやその構成要素を調整すること

Scrum では，検査と適応を行うイベント（Sprint 計画，デイリースクラム，Sprint レビュー，振り返り）が規定されており，プロジェクトを遂行する Scrum チームが繰り返し行うことを求めている．Scrum プロジェクトにおける 1 回の繰返しは Sprint と呼ばれており，1 週間〜1 カ月の決まった期間単位で実施される．また，Scrum チームにはプロダクトオーナ（PO），Scrum マスタ（SM），開発チームの各メンバが含まれる．Scrum では自己組織化と呼ばれるコンセプトに基づいて，各メンバが主体的に判断し，誰かに与えられるのではなく各自がタスクを割り当て，遂行する．そのため，Scrum チームには，代表的なソフトウェア開発プロセスの 1 つであるウォーターフォールモデル [18] において存在するような，進捗やタスクの割当てをコントロールする PM（Project Manager）に相当する役割が存在しない．Scrum チームの中心的な役割を果たす Scrum マスタはサーヴァントリーダと呼ばれ，指示ではなくメンバが成果をあげるために支援や奉仕を行うことが求められる．

図 1 に Scrum プロジェクトにおける Sprint の流れを示す．以降では，Sprint 計画から始まる Sprint の流れについて説明する．

2.2.1 Sprint 計画

Sprint 計画において，何ができるか，またどのようにしてなしとげるか，を PO が中心となって決定する．PO は顧客との交渉によって，プロダクトバックログと呼ばれる顧客に提供する価値のリストを構築する．開発チームのメンバは自分たちの過去の生産性に基づき，プロダクトバックログ項目のいくつをこの Sprint で開発するか決定する．

さらに開発チームは，選択したプロダクトバックログ項目ごとに，各項目を “完成させる” ために何をすればよいかを決定する．ここで，過去および今回の Sprint で完成したプロダクトバックログ項目のことをインクリメントと呼び，インクリメントを作成するための計画を示したリストをスプリントバックログと呼ぶ．これらのバックログを構築する際に最も重要なことの 1 つは，“完成の定義”（Definition of Done，以降 DoD と呼ぶ）である．完成したかどうかの判断について，全メンバが DoD を通して共通の理解を持つことが求められる．DoD を策定することにより，プロダクトバックログ項目等の「完成」を評価することが可能となる．

Scrum チームはプロダクトバックログとスプリントバックログを Sprint 計画の成果物として構築し，開発フェーズに移行する．

2.2.2 開発フェーズとデイリースクラム

開発チームは Sprint 計画の成果物に基づき開発を進める．このとき，一定期間ごと（通常 24 時間）に，各メンバが以下の 3 つの質問に対する回答を共有するためのミーティング（デイリースクラム）を開催する．

- 昨日やったことは何か？
- 今日やることは何か？
- 何かゴール達成の障害となるものを目撃したか？

デイリースクラムを通じて，開発チームは進捗状況や発生した問題の透明化を行い，計画どおりのインクリメント完成を目指す．Scrum マスタは，事前に定められた DoD が遵守されているかの確認や，開発チームのメンバが開発に集中できる環境の構築，といった支援を開発フェーズ中に行う．開発中に発生した問題の解決や進捗に応じたタスク割当てといった，進捗管理やプロジェクト管理の責任は開発チーム全体で負う．

2.2.3 Sprint レビュー

Sprint レビューでは，まず PO が顧客に対して何が完成していて何が完成していないかを説明する．その後，開発チームがインクリメントのデモンストレーションを行う．実際に動作するアプリケーションとして開発成果のデモンストレーションを行うことで，チームと顧客の間で，顧客要求と成果物のずれや進捗状況の共有が可能となる．Sprint レビューの結果に基づいて，振り返りおよび次の Sprint 計画が進められる．

2.2.4 振り返り

振り返りは各 Sprint の最後に実施され，Scrum チームの検査と次の Sprint に向けた改善計画の作成が行われる．振り返りの目的は下記の 3 つである．

- 人・関係・プロセス・ツールの観点から今回の Sprint

を検査する．

- うまくいった項目や今後の改善が必要な項目を特定・整理する．
- スクラムチームの作業の改善実施計画を作成する．

2.3 Scrum 教育における課題

Scrum フレームワークをチームによるソフトウェア開発の経験がない受講生に指導することは容易ではない．ここでは，我々が Scrum 教育における課題としてとらえているプロセス理解，学習機会の不均衡，プロジェクト評価の 3 つについて詳述する．

2.3.1 プロセス理解の困難さ

SDPBL では，受講生がソフトウェア開発におけるライフサイクルを体験し，開発プロダクトおよびプロセスに関する知識や技術を実体験に基づいて獲得することが 1 つの目的である．そのためには，受講生自身が所属するチームの開発プロセスを振り返り，問題点を考察し，工夫・改善するという PDCA のサイクルを回すことが重要となる．

しかしながらチームでのソフトウェア開発の経験が少ない受講生の多くは，納期のみを重視し，開発プロセスや付随するルールを理解し，遵守することを軽視しがちである．たとえば，開発したソフトウェアの正常系のみの動作を確認し，異常系を考慮しなかったり，単体テストやコードレビューといった実装以外のプロセスを実施しなかったり，といったことが行われる．また，プロセスの軽視はプロダクト品質の低下だけでなく，進捗の遅れを招く場合もある．このようなプロジェクトでは，締め切り切直前に多くのエフォートがさかれることになる [14]．

このようなプロセス軽視は一般的にはプロダクト品質の悪化を招くが，短期間で実施される教育プロジェクトでは，プロセスが軽視されていたにもかかわらず，結果としてプロダクトが完成し，品質の低下が顕在化しないことがある．そのようなプロジェクトでは，ソフトウェア開発プロセスおよび付随するルールを受講生が理解し，改善し，遵守するといった Scrum 教育における目的が達成されないことになる．

2.3.2 学習機会の不均衡

チームでの作業を行う際，social loafing や free rider という特定のメンバがチームに貢献をしなくなるという問題が起きることがある [22]．特に Scrum に基づくプロジェクトでは，開発チームのメンバは主体的に各自が実施するタスクを選択することが求められる．また，ソフトウェア開発における実装のように，開発者間で生産性が大きく変わるようなタスクを含むプロジェクトでは，生産性の高いメンバのみで開発を進めたほうが効率が良いため，プロジェクト開始時のプロダクト開発スキルが劣るメンバが開発に関与しない状況が発生しうる．

結果として SDPBL において，プロダクト開発スキルやプロセス遂行にともなうスキルについて，十分な経験を積むことができない受講生が発生してしまう可能性がある．さらに，各受講生が自分にできるタスクしか選択しなくなった結果，受講生間での知識伝達の機会が失われてしまうことも想定される．

一般的に，SDPBL のような経験による学習を目的とした教育手法において，このような学習機会の不均衡をどの程度考慮するべきかは，カリキュラムの設計に依存する．たとえば，実際のソフトウェア開発現場における不均衡を想定し，あえて教員による是正措置をとらず，受講生ら自身が解決するというカリキュラム設計も考えられる．しかしながら，チームによるソフトウェア開発経験がない受講生に，Scrum フレームワークの初歩を教育するという我々のカリキュラムにおいては，できる限り多くの受講生に，多様なプロセス・プロダクトに関する経験を積んでもらい，1 つでも多くの新しいスキルを習得してもらうことが望ましい．

2.3.3 プロジェクト評価

プロジェクト評価は PBL 実施において重要な課題の 1 つである [2]．SDPBL においても，受講生等によって実施されるプロジェクトの何を，どのように評価するかは重要な課題として認識されている．Paasivaara ら [11] は，受講生間の email やチャットのログとインタビュ結果を組み合わせて，分散 Scrum プロジェクトにおける各受講生の役割を定量的，定性的に評価する仕組みを，Kilamo ら [20] はチームにおける貢献度合いの質と量を評価する karma model を提案している．ほかにも，アンケートによる評価や開発されたソフトウェア，プレゼンテーション，作業計画，日報等のプロジェクトの成果物にともなう評価等，多様な評価手法が提案されている [3], [5], [12], [19], [23]．

このように多くの既存研究において，プロジェクトの成果物やチームへの貢献度・コミュニケーション内容を対象とした評価手法が提案されている．一方で，2.3.1，2.3.2 項で述べた，ソフトウェア開発プロセスや DoD のような付随する様々なルールおよびタスク割当てを考慮した評価手法やメトリクスはほとんど提案されていないのが現状である．

3. Scrum プロジェクトにおける定量的評価手法

我々は受講生による Scrum プロジェクトにおけるソフトウェア開発プロセスの遵守度合いやタスク割当ての偏りにともなう学習機会の不均衡を評価し，受講生にフィードバックすることを目的として，チケットシステムを利用した定量的評価手法を提案する．

3.1 チケット駆動開発

チケット駆動開発とは，開発者の実施するタスク（スプリントバックログ項目）をチケットとして Trac[*1]や Redmine[*2]といったチケットシステムに登録しておき，開発者のタスク実施状況とチケットの状態を連動させることによって開発を駆動するソフトウェア開発手法である [6], [8], [28]．たとえば，「a.java をレビューする」というチケットが存在した場合，そのチケットの担当者に割り当てられた開発者は自分がそのタスクを実施する際に，チケットの状態を「着手」に変更する．他の開発者はチケットの状態を見ることで，誰が a.java のレビューを実施しているのかを把握することができる．

SDPBL にチケット駆動開発を導入することで，受講生自身によって開発プロセスの透明化が行われるようになる．一般的な Scrum プロジェクトにおいてもプロセス透明化の一環として，スプリントバックログを登録する Scrum Wise[*3]や Agilefant[*4]といったサービスが利用されている．我々はチケットシステムに登録されたプロセス記録だけでなく，版管理システムやビルドシステムといったプロジェクト支援環境を用いることで，受講生の実施しているプロセスやルールの遵守度合い，タスク割当て状況の定量的な評価を実現する．

3.2 QAD（Quality, Assignment, and Delivery）

一般に，ソフトウェア開発プロジェクトの評価においては，Quality，Cost，Delivery を表す QCD [7] と呼ばれる基準が用いられることが多い．我々は SDPBL における Scrum に基づくプロジェクトを評価するために，Quality，Assignment，Delivery を表す QAD という評価基準を提案している．以降では，QAD の各項目に属するメトリクスについて紹介する．

3.2.1 Quality

Quality は受講生の Scrum プロジェクトにおけるプロセスとプロダクト両方の品質を評価する基準である．ここでは，単体テストのカバレッジのような一般的なプロダクト品質評価のメトリクスのほかに我々が定義したプロセス評価のためのメトリクスを示す．

Q_{SB}： チームによるソフトウェア開発では，Git や Subversion といった版管理システムが利用されることが多い．チームで版管理システムを利用する際には，守るべきいくつかのルールが存在する [13]．その中で最も重要なものの 1 つに，ビルドに失敗する不完全なソースコードを版管理システムにコミットしない，というものがある．不完全なソースコードが開発者によってコミットされると，他の開発者がそのソースコードをチェックアウトした際にプロジェクトのビルドが失敗してしまう．結果として，開発の継続が困難になり，他の開発者によって引き起こされた問題を修正しなければならなくなる．我々の SDPBL では，このルールをチームでどの程度遵守できているかを評価するために Q_{SB} を定義する．Q_{SB} はチームのある Sprint（あるいは Sprint 群）における版管理システムへの総コミット回数のうち，ビルドが成功したコミット回数の割合として表される．

Q_{CT}： Scrum に基づいてプロジェクトを進める際には，2.2 節で述べたとおり，プロセスの透明化が求められる．そこで，各メンバによる版管理システムへのコミットがチケットシステムに登録されているスプリントバックログ項目（チケット）とどの程度対応づいているかを評価する指標として Q_{CT} を定義する．もし，あるメンバがチケットシステムにチケットを登録せずにコミットを行っていた場合，そのコミットに対応するスプリントバックログ項目はチケットとして他のメンバと共有・透明化されていないことになる．Q_{CT} は全コミットのうち，対応するチケットが存在するコミットの割合として計算される．ここで，本論文で定義する対応するチケットが存在するコミットの条件を下記に示す．

- チケットに記載されている担当者情報とコミットを行ったメンバが一致する．
- チケットに記載されているコンポーネント（そのスプリントバックログ項目の成果物）とコミットされたコンポーネントが一致する．
- チケットに記載されている作業開始・終了時刻の範囲内にコミットが行われている（ただし，終了時刻直後のコミットも許容する）．

Q_{RV}： Sprint における開発フェーズ終了時のインクリメントに含まれるすべてのコンポーネントは開発者とは異なるメンバにレビューされていることが望ましい．Q_{RV} はインクリメントに含まれるその Sprint で開発されたすべてのコンポーネントのうち，レビューが適切に行われている（レビューが行われており，レビュワが開発者と異なっている）コンポーネントの割合を示す．

Q_{IT}： Sprint で開発されたすべてのプロダクトバックログ項目は，Sprint レビュー前にテストされているべきである．Q_{IT} はその Sprint で開発されたすべてのプロダクトバックログ項目のうち，結合テストが実施されたものの割合を示す．

[*1] http://trac.edgewall.org/
[*2] http://www.redmine.org/
[*3] https://www.scrumwise.com/
[*4] http://agilefant.com/

表 1 スプリントバックログ項目種別一覧
Table 1 Sprint backlog items.

T_{COD}	ソースコード（テストケースを除く）の実装
T_{WUT}	テストケースの設計・実装
T_{RV}	レビュー．なお，ここでのレビューとはコードレビューおよびテストケースレビュー（テスト実施および結果の評価）を含むものとする
T_{WIT}	結合テストの設計および実装
T_{EIT}	結合テストの実施および結果の評価

3.2.2 Assignment

Assignment はスプリントバックログ項目の種別ごとに，全開発メンバがどの程度均等に分担できたかを評価する指標である．ここでは，受講生が経験すべきスプリントバックログ項目の種別を**表 1** のように定義する．なお，我々が実施した SDPBL におけるスプリントバックログ項目の種別としては，バグ修正等上記以外のものも定義したが，評価の対象とはしていない．これはバグ修正をメンバ全員で均等に分担することは非常に困難なためである．Assignment において評価基準とする分類をどのように決定するかは，カリキュラム設計に依存する．

これらの各種別について，それぞれどの程度均等に分担できたかを下記に示す基準に基づいて評価する．

A_{COD}： ソースコードの実装をどれだけ全員で均等に分担できているかを表す．ある Sprint（あるいは Sprint 群）において n 人で構成されるチーム M のメンバ m_i（$\in M$）が実施した T_{COD} に相当するスプリントバックログの項目数を $N_{COD}(m_i)$ とする．また，メンバ 1 人あたりの平均担当スプリントバックログ項目数を

$$\overline{N_{COD}} = \frac{1}{n}\sum_{i=1}^{n} N_{COD}(m_i)$$

としたときに，

$$\left|\frac{N_{COD}(m_i)}{\overline{N_{COD}}} - 1\right| \leq \alpha$$

が成立するようなメンバ m_i の全メンバに占める割合を A_{COD} とする．α を均衡度と呼び，0 以上 1 未満の数値で指定するものとする．たとえば，あるチーム M のある Sprint において，$\sum_{i=1}^{n} N_{COD}(m_i)$ が 30，n が 5 で α が 0.2 であり，全メンバのうち，$N_{COD}(m_i)$ が 5，6，7 のいずれかであるメンバが 3 人存在した場合，そのチームのその Sprint における A_{COD} の値は 0.6 となる．

A_{WUT}： テストケースの設計・実装をどれだけ全員で均等に分担できているかを表す．具体的には T_{WUT} に相当するスプリントバックログ項目について，全メンバがどの程度均等に担当したかを A_{COD} と同様に算出する．

A_{RV}： レビューをどれだけ全員で均等に分担できているかを表す．具体的には T_{RV} に相当するスプリントバックログ項目について，全メンバがどの程度均等に担当したかを A_{COD} と同様に算出する．

A_{WIT}： 結合テストの設計および実装をどれだけ全員で均等に分担できているかを表す．具体的には T_{WIT} に相当するスプリントバックログ項目について，全メンバがどの程度均等に担当したかを表す．ただし，結合テストは T_{COD} 等の種別ほど数が多くないため，ここでは，少なくとも 1 回以上実施したメンバがどの程度の割合存在するかを算出するものとしている．

A_{EIT}： 結合テストの実施をどれだけ全員で均等に分担できているか表す．具体的には T_{EIT} に相当するスプリントバックログ項目について，全メンバがどの程度均等に担当したかを表す．ここでは，A_{WIT} と同様に，少なくとも 1 回以上実施したメンバがどの程度の割合存在するかを算出するものとしている．

これらのメトリクスを収集することで，受講生に経験してほしい種別ごとに学習機会を均等化することが可能となる．

3.2.3 Delivery

Scrum プロジェクトでは，プロダクトバックログ項目のうち，どれをいつ完成させるか（あるいはさせないか）は Scrum チーム自身が顧客と調整のうえ決定する．そこで，Delivery に関する評価基準としては，納期に間に合ったかどうかではなく，下記 2 つについて評価する．

D_{NOU}： 完成したプロダクトバックログ項目の数

D_{POA}： ある Sprint（あるいは Sprint 群）において開発を予定していたプロダクトバックログ項目のうち，実際に完成した項目の割合

通常の Scrum プロジェクトにおいては，プロダクトバックログの内容は Sprint ごとに変更になる可能性がある．我々が想定する SDPBL では 2.1 節で述べたとおり詳細設計より後を対象とするため，プロダクトバックログ項目自体は教員から提示し，受講生はどのプロダクトバックログ項目を開発するかを教員と調整のうえ決定することになる．

4. 定量的評価基準に基づく SDPBL の設計

QAD 基準に含まれる各メトリクスに基づいて我々が設計した SDPBL について詳述する．本論文で対象とする SDPBL は *Educational Network for Practical Information Technologies*（enPiT）の一部として実施されている，アジャイルソフトウェア開発およびクラウドコンピューティ

ング教育のためのカリキュラム Cloud Spiral の一環として，クラウド基礎 PBL という名称で実施されている．クラウド基礎 PBL では，アプリケーションを納期までに完成させることだけではなく，2.3 節で述べたような Scrum フレームワークへの理解やチームソフトウェア開発で守るべきルールの遵守，プロダクト開発スキルをできる限り多くの受講生に体験・理解してもらうことを目的としている．西日本の様々な大学から集まった受講生が 5〜6 人でチームを組み，この目的に従って，Web アプリケーションを 6 日間のコース（最終日は成果発表会）において開発する．2013 年度のクラウド基礎 PBL では，西日本 9 大学から 49 人の受講生が集まり，9 チームに分かれて開発を行った．以降では，2013 年度クラウド基礎 PBL を例に定量的評価基準に基づく SDPBL のカリキュラム設計について述べる．

4.1 開発プロセス

クラウド基礎 PBL の開発フローは図 1 に従う．ただし，開発期間が 5 日間のため，1Sprint を 1 日として 5Sprint 実施している．以下に Sprint のタイムテーブルと開発プロセスの設計について述べる．

S1 Sprint 期間は 1 日．受講生は Sprint 計画から振り返りまでを 9 時から 21 時の範囲内で行う．ここで，チームごとの開発時間の乖離や過度の残業を防ぐために，開発フェーズ開始時刻と Sprint レビュー開始時刻のみ教員から指定した．以下に典型的なタイムテーブルを示す．

9:00〜10:30： Sprint 計画
10:30〜17:00： 開発フェーズ
17:00〜17:30： Sprint レビュー
17:30〜19:00： 振り返り

S2 1 日を 1Sprint とし，5 日間開発を行う．なお，Sprint#1 のみ，チュートリアルとして開発環境の説明等を教員が行う．

S3 Scrum マスタは全員が交代で担当する．開発期間が 5 日間のため，6 人チームの場合は任意の開発日に Scrum マスタ 2 人で担当する．Scrum マスタが開発チームとして関与するかどうかは各チームの判断に委ねられる．

S4 プロダクトバックログ項目およびスプリントバックログ項目について，いくつかの DoD（Definition of Done）が教員によって前もって提示される．ここで，3.2 節で述べた QAD 基準に含まれる各メトリクスについても守るべきルールとして受講生に事前に提示される．**図 2** および下記に DoD の 1 例を提示する．この図は，java コンポーネントの開発を完了するためには，開発者は T_{COD}，T_{WUT}，T_{RV} を指定された順に実施しなければならないことを示している．

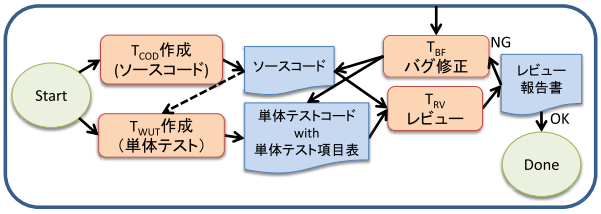


図 2 java コンポーネントの開発における DoD 例
Fig. 2 A sample workflow of a DoD for a java component.

- 同一コンポーネントの T_{COD} と T_{RV} を担当する開発メンバは異なっていなければならない．
- 開発メンバは T_{COD}，T_{WUT}，T_{RV} 終了するたびに，対応する成果物を版管理システムにコミットしなければならない．
- 版管理システムに保存されているソースコードはつねにビルドが正常に実行できる状態でなければならない．
- 均衡度 α を 0.2 とする．

S5 プロセス透明化のために，すべてのスプリントバックログ項目を受講生自身がチケットとしてチケットシステムに入力する．このとき，入力するスプリントバックログ項目の種別は教員が提示したものを利用する．チケットには，スプリントバックログ項目の概要と種別，開発対象コンポーネントおよび見積時間，開始時刻・終了時刻が入力されていなければならない．

S6 振り返りは KPT 法 [4] に基づいて実施する．ここで，K（Keep）は継続して実行すべきこと，P（Problem）は改善されるべき問題点，T（Try）は次回以降取り組むべきことを表す．Scrum マスタは KPT の項目を議事録としてまとめ，教員に提出する．

クラウド基礎 PBL 開始前に上記内容を全受講生に説明する．同時に，上記以外の内容については，すべて受講生に任されており，新規 DoD の追加や開発フェーズ開始・終了時刻以外のタイムテーブルの変更等自由に行ってかまわないものとしている．

4.2 開発環境

受講生は下記環境を利用して開発を行った．

- クライアント側開発環境
  - Eclipse IDE for Java Developers 4.2
  - Oracle JDK 7u25
  - Apache Tomcat 7.0.41
  - MongoDB 2.4.2

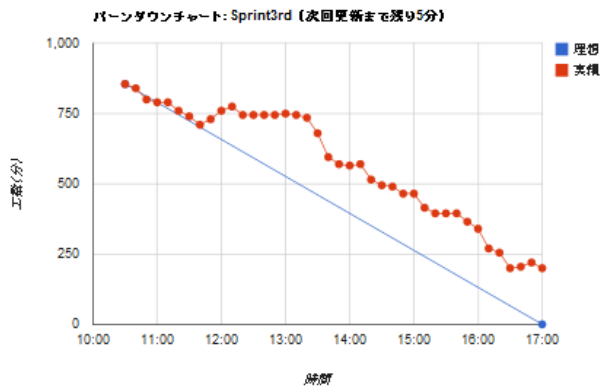


図 3 スプリントバーンダウンチャート例
Fig. 3 A screen image of sprint burndown chart.

- サーバ側開発支援環境
  - Subversion 1.8.0
  - Trac 0.12
  - Jenkins 1.525

全受講生はクライアント側開発環境を利用し，Web アプリケーションを開発する．開発支援環境には，版管理システムとして Subversion を，チケットシステムとして Trac0.12 を，継続的インテグレーション支援ツールとして Jenkins [*5] を導入した．Jenkins は開発者が版管理システムにコミットを行った際に，連動して自動的にビルドや単体テストを実行し，その結果を開発者に報告してくれるツールである．Jenkins を用いて，Q_{SB} を評価している．

上記以外のツールとして，チケットシステムに登録されたチケットを**図 3** のようにバーンダウンチャートの形式で可視化するツールを開発し，受講生に提供した．バーンダウンチャート [16], [17] は一般的な Scrum プロジェクトでも用いられているもので，未完了のスプリントバックログ項目群の作業見積時間を合計したものが，時間の推移とともに提示される．この図の場合，15:00 の時点で，残スプリントバックログ項目群の作業見積り時間合計が約 500 分であることを示しており，その値は理想的な作業推移と比較して，約 250 分遅れていることが確認できる．

受講生はこれらの開発環境，開発支援環境に加えて，ホワイトボードや付箋といった道具を用いて，チームのプロセスを透明化し，振り返りを行い，クラウド基礎 PBL 期間中を通じて，継続的に改善を続けていくことになる．

4.3 開発対象ソフトウェア

クラウド基礎 PBL において受講生は EventSpiral と名付けられたチケット予約システムの開発を行う．EventSpiral は DWR [*6] と呼ばれる Web アプリケーションフレームワークを用いて，HTML，JavaScript および Java で実装され，DB には MongoDB が使われる．受講生にはチケットの購入，イベントの登録，といった 13 のユースケース（ここではプロダクトバックログ項目と同じものとする）の詳細設計書が提示され，期間内に 1 つでも多くのプロダクトバックログ項目を完成させることを目指す．規模としては，全プロダクトバックログ項目の開発が完了した時点で SLOC が 5,000 行程度，その他単体テストコードが同程度の行数になる．ただし，初期状態でログイン・ログオフ，ユーザ登録の 3 つのプロダクトバックログ項目については完成している．また，詳細設計書として，ユースケース図，ロバストネス図，クラス図，シーケンス図，画面遷移図，UI 設計書およびクラス・メソッド仕様書が受講生に提示される．なお，どのプロダクトバックログ項目から順に実装するかは受講生に任されている．

*5 http://jenkins-ci.org/
*6 http://directwebremoting.org/dwr/index.html

5. SDPBL における QAD メトリクス計測

クラウド基礎 PBL において，我々教員は 4.2 節で述べた受講生が利用する開発環境に保存された情報に基づき，QAD の各メトリクス値を計測し，受講生にフィードバックを行った．本章では，各メトリクスの計測手法とフィードバック手法について詳述する．なお，すべてのメトリクス計測は受講生のサーバ側開発支援環境上（CentOS 6.5）のシェルスクリプトを利用して行われている．

5.1 Quality メトリクスの計測

Q_{SB}： Jenkins に保存されたログを利用して計測を行う．Jenkins はユーザが行うコミットに連動し，その時点におけるチームのソースコードのビルドを行う．そのため，Jenkins が行った全ビルドのうち，成功した回数を Jenkins ログより収集し，Q_{SB} を計測する．

Q_{CT}： チケットシステム（Trac）および版管理システム（Subversion）のログを利用する．Q_{CT} メトリクスの計測においては，3.2.1 項の Q_{CT} 項目に示したチケットが存在するコミットの条件に基づき，チームごとの Subversion リポジトリに含まれる各コミットが対応するチケットを持つか分析し，メトリクス計測を行った．

Q_{RV}： チケットシステム（Trac）のログを利用する．Trac にチケットとして記録されたすべてのタスクのうち，レビュー（T_{RV}）に関するものを抽出する．その後，抽出したレビュータスクそれぞれについて，レビュータスクの実施メンバ（レビュー者）および同一コンポーネントを対象として実施された直前の実装タスク（T_{COD} あるいは T_{WUT}）の実施メンバ（開発者）を特定する．その後，各レビュータスクについて，レビュー者と開発者が異なっているかを確認する．最終的に，全コンポーネントのうち，どの程度のコンポーネントについてレビュー者と開発者が異なっているかを Q_{RV} として算出する．

Q_{IT}： チケットシステム（Trac）のログを利用する．Trac にチケットとして記録されたすべてのタスクのうち，T_{EIT} すなわち結合テストの実施に関連するものを抽出する．各スプリント終了時に，そのスプリントにおいて開発が進められた各プロダクトバックログについて，結合テスト実施タスクが存在するかを特定することで，Q_{IT} を計測する．

5.2 Assignment メトリクスの計測

すべての Assignment に関するメトリクスはチケットシステム（Trac）のログを利用することで計測を行う．たとえば，A_{COD} の場合，ソースコードの実装（T_{COD}）に関連するタスクが記録されたチケットをすべて抽出し，その中からそのタスクを実施したメンバを特定する．どのメンバがソースコードの実装を何回実施したか算出し，A_{COD} を計測する．

5.3 Delivery メトリクスの計測

Delivery の各メトリクスはスプリントごとに実施されるスプリントレビュー時に計測される．4.1 節で述べたスプリントレビューでは，教員が受講生に対して，(1) そのスプリントで開発を予定していたプロダクトバックログ項目，(2) 開発が完了したプロダクトバックログ項目，の 2 点を確認する．その後，(2) を対象として仕様書に準拠した開発が行われているか，受講生によるツールのデモンストレーションを通じて教員が確認する．結果として，この 2 項目によって，D_{NOV} および D_{POA} を計測することが可能となる．

5.4 フィードバック

各メトリクスは Sprint が終了するたびに計測される．計測結果のフィードバックは最終 Sprint 以外の Sprint 終了時と最終 Sprint 終了時で異なった形式で行われる．最終 Sprint 以外の Sprint 終了時には，Quality および Assignment の一部のメトリクスとすべての Delivery メトリクスについて，教員が全チームの結果を提示する．このとき教員は，どのデータがどのチームのものであるかは示さず，メトリクス値がどのような傾向になっているかのみをデータに基づく考察という形で提示する．一部のメトリクスのみを対象としてチーム名を隠して提示する理由は，全メトリクスの具体的なデータを各 Sprint 終了時にチームごとに提示した場合，受講生は各自でメトリクス計測を行う必要がなくなってしまうためである．クラウド基礎 PBL では，QAD メトリクスの計測とチームプロジェクトの改善を受講生自身が行えるようになることをその学習目標の 1 つとして定めている．

最終 Sprint 終了時には，全チームの全メトリクス計測結果を受講生に提示し，講評としてフィードバックを行う．ここで教員はすべての計測結果をいっさい伏せずに受講生に対して提示する．

6. プロジェクト評価結果

クラウド基礎 PBL において我々が実施した QAD に基づくプロジェクト評価結果を示す．

6.1 Quality

図 4，**図 5** はそれぞれ全チームの Q_{SB}，Q_{CT} の推移を示しており，X 軸は Sprint を，Y 軸は各メトリクスの値（%）を表している．図 4 は Q_{SB} すなわちコミット時のビルド成功割合の推移を示している．ここでは，Sprint#1，#2 ではばらつきがあった成功割合が，#3 以降ほぼ全チームについて 90%以上に収束し，最終的に 7 チームが Sprint#5 におけるビルド成功率 100%に到達したことを示している．この結果より，ビルドが失敗するようなコミットを行わないというルールの重要性を受講生が理解し，遵守するようになったことが確認できた．

図 5 は全チームの Q_{CT} の推移，すなわち Sprint ごとのチケットに対応づくコミットの割合を示している．さらに，**図 6** および**図 7** において，Q_{CT} のチーム T3，T7 におけるメンバごとの値の推移を示している．図 7 によると，T7 は 5 人のメンバから構成されており，5 人中 4 人について，最終的に 80%以上のチケット–コミット対応付けが行われるようになっている．一方，T3 は 6 人から構成

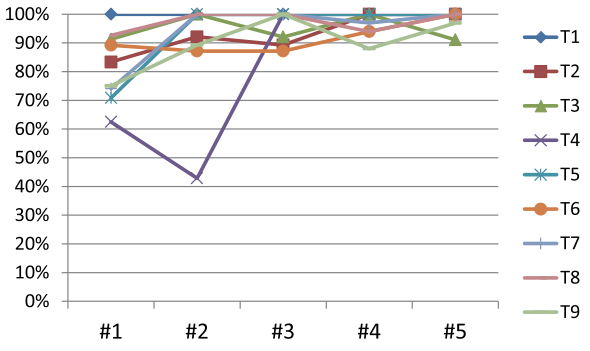


図 4 Q_{SB} の結果
Fig. 4 The result of Q_{SB} for all teams.

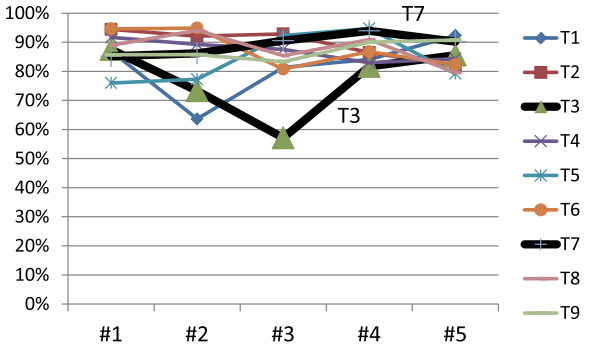


図 5 Q_{CT} の結果
Fig. 5 The result of Q_{CT} for all teams.

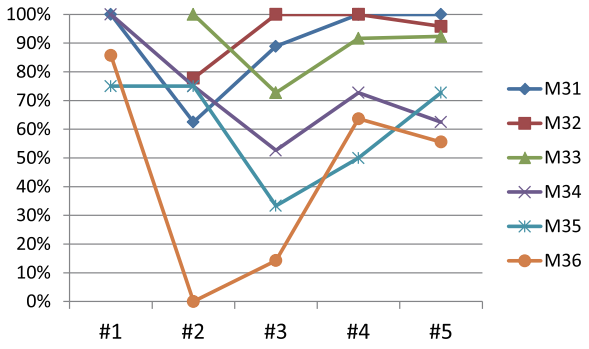


図 6 T3 の Q_{CT} 結果
Fig. 6 The result of Q_{CT} for T3.

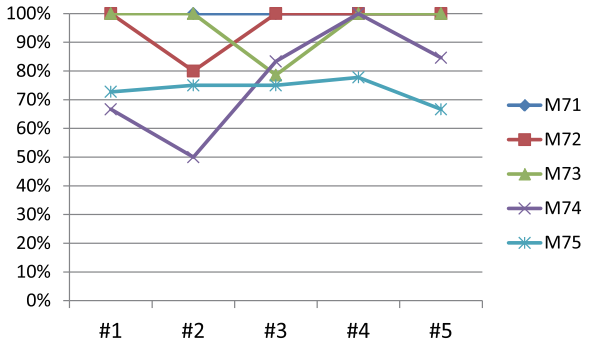


図 7 T7 の Q_{CT} 結果
Fig. 7 The result of Q_{CT} for T7.

されており，Sprint ごとに徐々に改善は見られるものの，80%を超えたメンバは 3 人にとどまっていた．また，最終 Sprint で値が低下したメンバが T3 では 2 人，T7 では 3 人存在した．最終 Sprint で値が低下した理由について受講生にインタビュを行ったところ，コミット–チケットの対応自体は意識するようになっていたものの，最終 Sprint で時間の余裕がなく，漏れが発生してしまった，とのことであった．なお，登録されたチケットを確認したところ，ほとんどの場合において，実施すべきスプリントバックログ項目そのものが漏れているわけではなくチケットへの開始/終了時刻の記入が漏れていた．以上のように，Q_{CT} については，コミットとチケットの対応付けをどの程度意識するかには個人差はあるが，ほぼ全員において最終 Sprint を除いて改善傾向が確認された．

6.2 Assignment

図 8 に全チームの A_{COD} 推移を示す．なお，ここでの A_{COD} の値は実装に関するスプリントバックログ項目の累積値を対象として計測している．たとえば，#5 であれば，Sprint#5 までのすべてのスプリントバックログ項目を対象として，誰がどの程度分担したかを計測し，A_{COD} を算出している．この値については，9 チームのうち 7 チームまでが，#5 に向けて割当て状況の改善が見られる．

図 9，図 10 は T3 および T7 のメンバごとの累積担当数推移を示している．これらの図で分かるように，T3 で

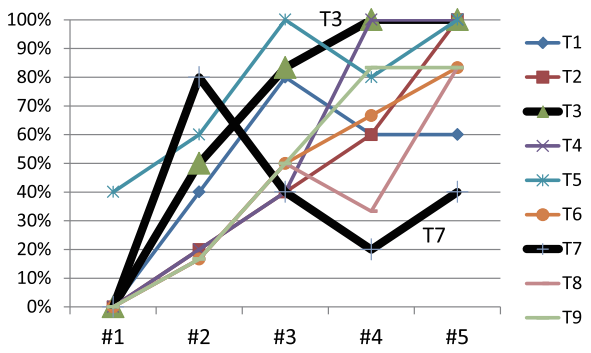


図 8 A_{COD} の結果
Fig. 8 The result of A_{COD} for all teams.

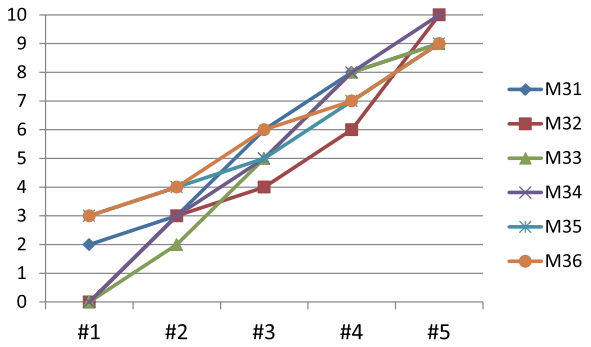


図 9 T3 のメンバごと T_{COD} 累積担当数推移
Fig. 9 Transition of accumulated number of T_{COD} tasks for each member in T3.

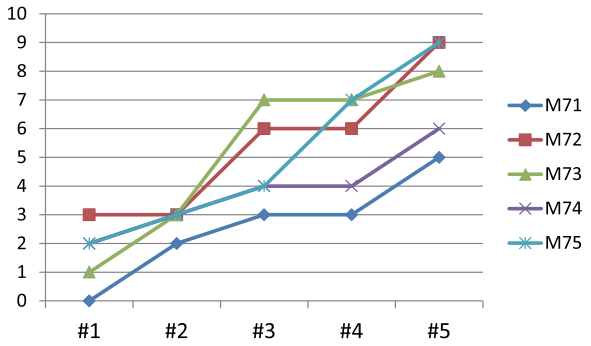


図 10 T7 のメンバごと T_{COD} 累積担当数推移
Fig. 10 Transition of accumulated number of T_{COD} tasks for each member in T7.

は当初よりメンバ間で均等になるよう意識してタスク割当てを行っているのが見て取れる．一方 T7 では，#5 の時点での平均担当数が 7.4 で $\alpha = 0.2$ なため，担当数が 8.9 を超えている M72 および M75 と 5.9 以下の M71 が割当て範囲から外れてしまっている．図を見る限り T3 では，#4 以降のタスク割当てについて，十分に考慮できなかったことが確認できる．このようにチームごとにタスク割当ての際にどの程度配慮できているかは大きく異なる．ただし，全チームにおいて，期間を通じて特定の種別のスプリントバックログ項目をまったく担当しないといった極端な偏りは発生しておらず，受講生に多くの経験を積んでもらうという Assignment 基準の目的はある程度達成できたと考え

表 2 Sprint#5 終了時の QAD メトリクス
Table 2 QAD metrics values after the Sprint #5.

	Q_{SB}	Q_{CT}	Q_{RV}	Q_{IT}	A_{COD}	A_{WUT}	A_{RV}	A_{WIT}	A_{EIT}	D_{NOU}	D_{POA}
T1	100%	92%	84%	75%	60%	40%	40%	100%	100%	7	100%
T2	95%	82%	83%	100%	100%	80%	100%	100%	100%	5	83%
T3	95%	86%	60%	63%	100%	67%	100%	67%	83%	5	56%
T4	87%	84%	88%	83%	100%	100%	100%	100%	100%	4	80%
T5	96%	79%	87%	83%	100%	40%	80%	100%	100%	4	80%
T6	92%	82%	81%	50%	83%	100%	100%	100%	100%	4	57%
T7	96%	90%	76%	100%	40%	40%	40%	100%	100%	3	60%
T8	98%	80%	50%	40%	83%	67%	100%	67%	100%	3	60%
T9	93%	91%	72%	57%	83%	33%	67%	100%	83%	3	50%

られる．

6.3 Delivery

表 2 は全 Sprint 終了時の全メトリクスの値をチームごとに示したものである．Delivery については，T1 が最も多くのプロダクトバックログ項目を開発しており（D_{NOU}），#5 における見積りにおいても，予定していた開発対象すべてを完成させていることが確認できる（D_{POA}）．今回実施したクラウド基礎 PBL では，完成したプロダクトバックログ項目の数に 3〜7 と大きな差が発生した．完成数が少ないチームの中には，ほぼ完成していたが，最後の結合テストが間に合わなかったチームも存在する．

7. 議論

本論文において我々は QAD 基準を下記の目的のために策定した．

O1： 受講生に守るべきソフトウェア開発プロセスおよび関連ルールの重要性を理解したうえで Scrum フレームワークに基づくチームによるソフトウェア開発を遂行してもらう．

O2： 実装，単体テスト，レビューといったプロダクト開発に関連するスキルを均等に分担し，できる限り多くの受講生に，多くの経験を積んでもらう．

O3： O1，O2 が実施されていることを受講生および教員の両方が定量的に確認できる手段を提供する．

結果として，表 2 に示したプロジェクト終了時のすべての QAD メトリクスにより，どのチームが O1〜O3 の目的について，どの程度考慮していたかを評価することが可能となった．この表から T2 および T4 は Quality についてはすべてが 80%以上で，Assignment については T4 は全項目において 100%，T2 も 80%以上と非常に高い数値となっており，総じて O1，O2 を考慮して開発を進めていたと考えられる．一方で T1 については開発したプロダクトバックログ項目数自体は多いものの，Assignment の各メトリクスについては低い値となっていた．すなわち，タスク割当てにおける均衡度合いを部分的に考慮せずに，開発を進めることを優先したことが分かる．同様に T3 についても，開発したプロダクトバックログ項目数は多いが，Q_{RV} および Q_{IT} の値が低くなっている．すなわち，レビューや結合テストが適切に行われていないコンポーネントやプロダクトバックログ項目が存在することを示している．

以上のように QAD 基準を用いることで，受講生のプロジェクトごとに納期以外の多様な点について，定量的に評価し，フィードバックすることが可能となった．フィードバック後の受講生の感想として，Assignment 基準に関するものを以下に示す．

> “Assignment のおかげでいろいろな種類のタスクを満遍なくできた．チームを助けるシステムを作らないと回らないため自分も助かった．”
>
> “合宿に Assignment 制約があったのが，プログラム未経験者としてはとてもありがたかったです．おかげで，未経験の状態から最後には人に教えられるようになるところまでなりました．”
>
> “Java にも JavaScript にもいままでほとんど触れたことがなかったので不安だったのですが，Assignment 制約があったおかげで，合宿の終わりにはどちらもできるようになったのがうれしかったです．”

Assignment 基準の目的は，多様な種類のタスクをできる限り多くの受講生に経験してもらうことであったが，結果として未経験のタスクを実施する際の助け合いといった副次的な効果が確認できた．また上記のコメントのほかに，Assignment の基準を意識した結果，レビューやテストを適正に実施するといった Quality 基準についても意識するようになったという意見も得られた．Quality，Assignment，Delivery という本論文で述べた各基準は相互に影響しあっている．Assignment 基準を満たすためには，そもそも単体テストやレビューを実施しなければならない．また，開発者は不得意なタスクも実施する必要があるため，開発により多くの時間を要することになり，Delivery に影響を与

える．そのため，受講生はQAD基準を守るために，不得意なタスクや複雑なコンポーネントのレビュー・テストといった時間のかかるタスクをいかに効率良く行えるようにするかをつねに考えながら開発を進めていかなければならない．結果として，SDPBLをO1〜O3に基づいて実施することが可能となったと考えられる．

提案手法の限界として，チケットシステムに受講生が入力するスプリントバックログ項目の記入漏れや記入誤りが存在した場合に，メトリクスの値が不正確になることが考えられる．クラウド基礎PBLでは，教員が定期的に記入漏れ等をチェックし，フィードバックを行うようにしているが，今後は入力誤りや記入漏れを自動推定し，フィードバックを行うシステムの構築を目指したい．

8. おわりに

我々はQuality，Assignment，Deliveryの基準に基づくScrumプロジェクトを前提としたSDPBLを設計し，実際に適用した．結果として，チームソフトウェア開発経験のほとんどない受講生を対象としたSDPBLにおいて，我々が提案したQAD基準がソフトウェア開発プロセスやタスク割当てを受講生に意識させることが可能であることが確認できた．今後は初期教育の次の段階として，より発展的な内容を前提としたScrumプロジェクトのための定量評価基準やSDPBL設計手法について検討していきたい．

なお，本論文は文献[9]の発表に，各メトリクスの詳細定義と，我々が実施したクラウド基礎PBLにおいて実際に適用したメトリクス計測手法を追記し，さらにPBLへの適用結果についての詳細な考察を実施し，まとめたものである．

謝辞 本研究はenPiT（分野・地域を超えた実践的情報教育協働ネットワーク）事業の一部として提供されている教育カリキュラム「Cloud Spiral」を対象としている．また，本研究はJSPS科研費24700030の助成を受けている．

参考文献

[1] Batatia, H., Ayache, A. and Markkanen, H.: Netpro: An Innovative Approach to Network Project Based Learning, *Proc. International Conference on Computers in Education* (*ICCE*), pp.382–386 (2002).
[2] Cheng, W. and Warren, M.: Making a difference: Using peers to assess individual students' contributions to a group project, *Teaching in Higher Education*, Vol.5, No.2, pp.243–255 (2000).
[3] Clark, N.: Evaluating student teams developing unique industry projects, *Proc. 7th Australasian Conference on Computing Education* (*ACE2005*), Vol.42, pp.21–30 (2005).
[4] Cockburn, A.: *Agile Software Development*, Addison-Wesley (2003).
[5] dos Santos, S. and Soares, F.: Authentic Assessment in Software Engineering Education Based on PBL Principles A Case Study in the Telecom Market, *Proc. 35th International Conference on Software Engineering* (*ICSE2013*), San Francisco, USA, pp.1055–1062 (2013).
[6] Fukuyasu, N., Saiki, S., Igaki, H. and Manabe, Y.: Experimental Report of the Exercise Environment for Software Development PBL, *Proc. 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp.482–487 (2012).
[7] Button, G. and Sharrock, W.: Project Work: The Organisation of Collaborative Design and Development in Software Engineering, *Computer Supported Cooperative Work* (*CSCW*): *The Journal of Collaborative Computing*, Vol.5, pp.369–386 (1996).
[8] Hosono, S.: A DevOps framework to shorten delivery time for cloud applications, *International Journal of Computational Science and Engineering*, Vol.7, pp.329–344 (2012).
[9] Igaki, H., Fukuyasu, N., Saiki, S., Matsumoto, S. and Kusumoto, S.: Quantitative Assessment with Using Ticket Driven Development for Teaching Scrum Framework, *Proc. ICSE Companion 2014 Companion Proc. 36th International Conference on Software Engineering*, pp.372–381 (2014).
[10] Mahnic, V.: A Capstone Course on Agile Software Development Using Scrum, *IEEE Trans. Education*, Vol.55, pp.99–106 (2012).
[11] Paasivaara, M., Lassenius, C., Damian, D., Raty, P. and Schroter, A.: Teaching students global software engineering skills using distributed scrum, *Proc. 2013 International Conference on Software Engineering* (*ICSE '13*), pp.1128–1137 (2013).
[12] Persson, M., Kruzela, I., Allder, K., Johansson, O. and Johansson, P.: On the Use of Scrum in Project Driven Higher Education, *Proc. World Congress in Computer Science, Computer Engineering and Applied Computing* (*2011*), Las Vegas/Nevada (2011).
[13] Qt Project: Qt Commit Policy, available from ⟨http://qt-project.org/wiki/Commit_Policy⟩ (2014).
[14] Rocha, F. and Stroulia, E.: Understanding individual contribution and collaboration in student software teams, *Proc. 2013 IEEE 26th Conference on Software Engineering Education and Training* (*CSEE&T*), pp.51–60 (2013).
[15] Scharff, C.: Guiding global software development projects using Scrum and Agile with quality assurance, *Proc. 24th IEEE-CS Conference on Software Engineering Education and Training* (*CSEE&T*) (2011).
[16] Scharff, C. and Verma, R.: Scrum to support mobile application development projects in a just-in-time learning context, *Proc. 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering* (*CHASE '10*), NY, USA, pp.25–31 (2010).
[17] Schwaber, K. and Sutherland, J.: The Scrum Guide, available from ⟨https://www.scrum.org/Scrum-Guides⟩ (2013).
[18] Sommerville, I.: *Software Engineering*, Addison (2010).
[19] Tai, G.X.-L. and Yuen, M.C.: Authentic assessment strategies in problem based learning, *Proc. Australasian Society for Computers in Learning in Tertiary Education* (*ASCILITE 2007*), Singapore, pp.983–993 (2007).
[20] Kilamo, T., Hammouda, I. and Chatti, M.A.: Teaching collaborative software development: A case study, *Proc. 2012 International Conference on Software Engineering*, pp.1165–1174 (2012).
[21] VERSIONONE: 7th Annual state of Agile Development

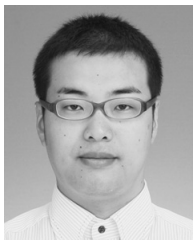
Survey, available from ⟨http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf⟩ (2014).
[22] Webb, N.M.: Group Collaboration in Assessment: Multiple Objectives, Processes, and Outcomes, *Educational Evaluation and Policy Analysis*, Vol.17, No.2, pp.239–261 (1995).
[23] Yamin, S. and Masek, A.: Ducation for sustainable development through problem based learning: A review of the monitoring and assessment strategy, *Proc. International Conference on Education for Sustainable Development in Technical and Vocational education and Training*, Manila, Philippines, pp.171–178 (2010).
[24] 木崎 悟，田原康之，大須賀昭彦：Scrum に基づきコミュニケーションを重視したソフトウェア開発 PBL の実践，ソフトウェアエンジニアリングシンポジウム 2013 論文集，pp.1–6 (2013).
[25] 井垣 宏，福安直樹，佐伯幸郎，柗本真佑：ぺた語義：ソフトウェア開発 PBL の定量的評価—クラウドコンピューティングの活用，情報処理，Vol.54, No.12, pp.1266–1269 (2013).
[26] 沢田篤史，小林隆志，金子伸幸，中道 上，大久保弘崇，山本晋一郎：飛行船制御を題材としたプロジェクト型ソフトウェア開発実習，情報処理学会論文誌，Vol.50, No.11, pp.2677–2689 (2009).
[27] 松澤芳昭，杉浦 学，大岩 元：産学協同の PBL における顧客と開発者の協創環境の構築と人材育成効果，情報処理学会論文誌，Vol.49, No.2, pp.944–957 (2008).
[28] 小川明彦，阪井 誠：チケット駆動開発，翔泳社 (2012).

井垣 宏 （正会員）

平成 12 年神戸大学工学部電気電子工学科卒業．平 14 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．平成 17 年同大学院博士後期課程修了．同年同大学院特任助手．平成 18 年南山大学数理情報学部講師．平成 19 年神戸大学大学院工学研究科特命助教．平成 22 年東京工科大学コンピュータサイエンス学部助教．平成 23 年大阪大学大学院情報科学研究科特任准教授．博士（工学）．ソフトウェア工学教育，サービス指向アーキテクチャ，ソフトウェアプロセス等の研究に従事．電子情報通信学会，日本教育工学会，日本ソフトウェア科学会，IEEE 各会員．

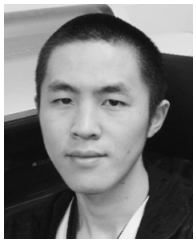
福安 直樹 （正会員）

平成 8 年名古屋大学工学部情報工学科卒業．平成 12 年同大学大学院工学研究科情報工学専攻博士後期課程修了．博士（工学）．同年和歌山大学システム工学部助手．平成 19 年同助教．平成 26 年同准教授．ソフトウェア開発環境，ウェブ工学，ソフトウェア工学教育に関する研究に従事．日本ソフトウェア科学会，教育システム情報学会各会員．



佐伯 幸郎

平成 16 年高知工科大学工学部情報システム工学科卒業．平成 21 年同大学大学院工学研究科基盤工学専攻博士後期過程修了．同年同大学助手．平成 22 年同大学助教．平成 25 年神戸大学システム情報学研究科特命助教．博士（工学）．ディジタル信号処理，ソフトウェア工学教育に関する研究に従事．電子情報通信学会，IEEE 各会員．



柗本 真佑 （正会員）

平成 18 年京都産業大学理学部卒業．平成 21 年日本学術振興会特別研究員（DC2）．平成 22 年奈良先端科学技術大学院大学博士後期課程修了．同年神戸大学大学院システム情報学研究科特命助教．博士（工学）．マイニングソフトウェアリポジトリ，クラウドコンピューティングの研究に従事．

楠本 真二 （正会員）

昭和 63 年大阪大学基礎工学部卒業．平成 3 年同大学大学院博士課程中退．同年同大学基礎工学部助手．平成 8 年同講師．平成 11 年同助教授．平成 14 年同大学大学院情報科学研究科助教授．平成 17 年同教授．博士（工学）．ソフトウェアの生産性や品質の定量的評価に関する研究に従事．電子情報通信学会，IEEE，JFPUG，プロジェクトマネジメント学会，日本ソフトウェア科学会各会員．