ゲーミフィケーションを用いた探索的テストの効果報告

根本 紀之
東京エレクトロン
noriyuki.nemoto@tel.com

1. 要旨

テスト仕様書ベースのテストが好きな開発者は少ない．もちろん例外はあるだろうが，一般的には少ないと言って過言ではないであろう．理由は創造的な活動ではない，テスト仕様書の作成に時間がかかる，など様々である．

本報告では，テスト仕様書ベースのテストの一部を，探索的テストに変え，さらにゲーミフィケーションを取り入れることで，バグを探すこと自体が面白くなり，通常用いているテスト仕様書ベースのテストより多くのバグを発見することができたという結果とその効果を報告する．さらにはテスト実行前に戦略立案，テスト実行後にふりかえりを行うことで，バグの効果的に見つける方法を共有する取り組みを紹介する．

2. 背景

『テストは単純作業で楽しくない．どちらかというと面倒で辛い作業』これが一般的な認識に近いであろう．筆者の経験でも開発者はテストフェーズに入ると，プログラミング時と比較して，モチベーションが下がるエンジニアが多いと感じていた．

図 1 は筆者が描いたテスト仕様書ベースのテストの問題構造図である．『テストフェーズは楽しくないので，モチベーションでダウンする』という問題が，『バグを積極的に見つけようとしない』マインドを引き起こし，最終的にリリース後のバグに繋がっているではないかという仮説に行きついた．

もう一つ着目したのは『自分が経験していないバグの出し方は良く分からない』という問題である．BTS を用いることでそのバグの現象や再現方法は共有できるが，筆者の開発経験の中ではバグを出す考え方は共有されることが少なかった．

これらの問題を解決することで，限られた時間の中で開発者が楽しく効率的にバグを見つけることができると考えた．

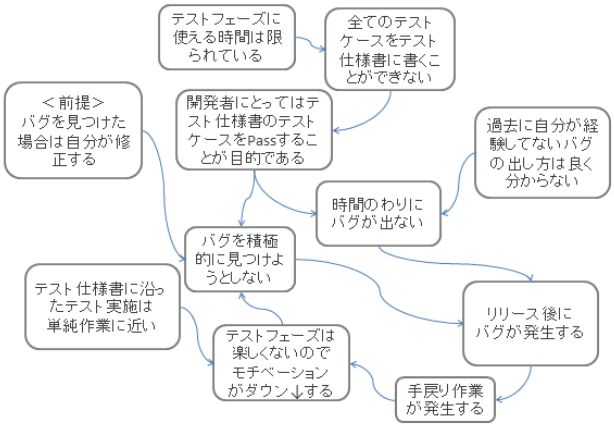

図 1 テスト仕様書ベースのテストの問題構造図

3. 目的

楽しくないと思われているテストにゲーミフィケーションを用いることでエンジニアの競争心を刺激し，モチベーションアップ，バグ検出効率向上に有効であることを確認すると共に，テスト戦略とバグの出し方を共有するふりかえりの効果を確認することを目的とする．

4. ゲーミフィケーションとは？

ゲーミフィケーションという言葉は「日常生活の様々な要素をゲームの形にする」という「ゲーム化（Gamefy）」から派生し，2010 年頃から使われはじめた[1] ．.ゲーム業界の中で独自に培われていたユーザを楽しませ，俗に言われる『ハマる』という状態を作りだすためのノウハウを教育や販促や人事評価など非ゲームのコンテキストで使用するのがゲーミフィケーションである．ゲーミフィケーションの基本的な要素は PBL と言われ，それぞれ P：ポイント，B：バッジ，L リーダーボードであり，最近は他の参加者とのチャットやチームでの取り組みなどソーシャ

ルな要素も重要視されている．ゲーミフィケーションという言葉自体は新しいものであるが，実は昔から身近な生活に取り入れられている．たとえば，ラジオ体操のスタンプカードなどもゲーミフィケーションの一つであるが，ラジオ体操自体の魅力は低くても，スタンプを押してもらうことに楽しみを見出し，夏休みの間，一回も休まず通った記憶を持つ方も多いのではないだろうか．

ソフトウェアテストに関連するゲーミフィケーションの論文としては Microsoft の Language Quality Game という事例が存在している．これは多言語対応されている Microsoft のダイアログボックスの誤記や意味の通じないメッセージを見つける作業をゲーム化した事例であり，参加者 4500 人，チェックしたダイアログボックスは 50 万以上，報告されたバクが 6700 件という結果が報告されている[2]．

5. 探索的テストとは？

James bachによると，「探索的テストは，学習，テスト設計，テスト実行を並行して実施するものである．」とされ [3]，従来のテスト仕様書ベースのテストであるスクリプトテストと区別される．探索的テストはアジャイル開発との親和性も高いことから，北米を中心とする海外ではメジャーなテスト手法となってきている[4]．

探索的テストはドキュメントの作成とメンテナンスのコストが少なくて済み，またテスト実施中に対象ソフトウェアの動作をフィードバックすることで怪しいところを重点的に攻めることができることから，一般的には費用対効果が高いと言われている．一方で，バグを出す考え方やノウハウは属人化する傾向が高く，探索的テストの課題の一つとなっている．

6. 手法

ゲーミフィケーションを用いた探索的テストの対象となる機能は，探索しやすいようにGUIを含み，厚くテストを実施する必要があるとチームで合意した機能とした．またこの機能は実験用の機能ではなく，実際に製品として開発した機能である．参加したエンジニアは 5 年目～15 年目の中堅エンジニア～ベテランエンジニアが中心であり，それぞれ設計，実装，テストと一通りの開発経験がある．テストは期間を置いて2回実施し，1回目は個人戦，2 回目はチーム戦である．それぞれの特徴を表 1 に示す．

個人戦では，テスト実施の時間は 1 時間のタイムボックスとし，テスト対象は参加者全員同じ機能を対象とした．同じ時間，同じ機能をテストすることで，公平性が生まれる．バグは見つけた開発者がその場で現象を共有ファイルに書き込んでいく．バグは重要度の高い順にAランク，Bランク，Cランクとポイントを規定し，テスト後のふりかえりで，ランク付けを行い，個人のポイントを決定した．また，ふりかえり時にはバグの現象だけではなく，そのバグをどのように出したかという暗黙知を共有するための質問を積極的に実施した．ふりかえりの時間は 1 時間程度とし，最後に個人 TOP を表彰した．

チーム戦も同じ時間に同じ対象についてテストを実施するという基本のルールは個人戦と同じである．チーム編成では，公平を期すため少なくともチームに一人はその機能の知見者を配置した．チーム戦で新たに付け加えた施策は，テスト前の戦略立案とリーダーボードの二つである．戦略立案は今回のイテレーションで開発された案件や過去のバグを確認し，テストで狙うべき箇所とチームでの役割分担を計画する．このアプローチは探索的テストから派生したセッションベースドテスト[5]に近いと考えている．

Web アプリであるリーダーボード（図 2）はテスト実施中に他のチームが発見したバグの件数がリアルタイムで分かるようになっている．この情報はあくまで件数であり，ランクのポイントが考慮された現在の順位は分からない仕掛けにしている．

表1 個人戦とチーム戦の違い

タイプ	個人戦	チーム戦
単位	1 人	3 人 1 チーム
参加人数	5 名	9 名 (3 チーム)
テスト対象機能	AAA 機能	BBB 機能
テスト前の戦略立案	なし	あり
テスト実施時間	1 時間	
テスト実施中の リーダーボード	なし	あり
テスト後のふりかえり	あり	
ポイント配分	A ランク ：5 ポイント B ランク ：3 ポイント C ランク ：1 ポイント	
賞	個人 TOP	個人 TOP チーム TOP

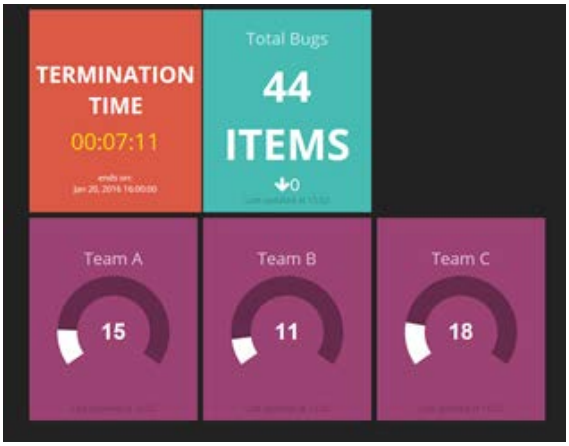

図 2 テスト実施中のリーダーボード

7. 結果

7.1. モチベーションアップ

テスト実施後に，「テスト仕様書に沿ったテストと比べて楽しかったですか？」というアンケートを取った．個人戦のアンケート結果を図 3 に，チーム戦のアンケート結果を図 4 に示す．チーム戦後のアンケートでは変わらないという人もいるが，平均的には楽しかった人が多くなっているのが見てとれる．

ゲーミフィケーションを用いた探索的テストの仕組みにより，モチベーションの低下を抑制できたと考えてよいであろう．筆者自身も実際にテスト実施の間は，他の参加者に負けたくない！高得点に繋がるバグを見つけたい！という思いでテストを実施していた．

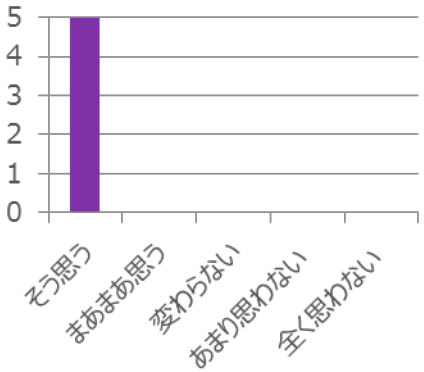

図 3 テスト仕様書に沿ったテストと比べて楽しかったですか？のアンケート結果（個人戦）

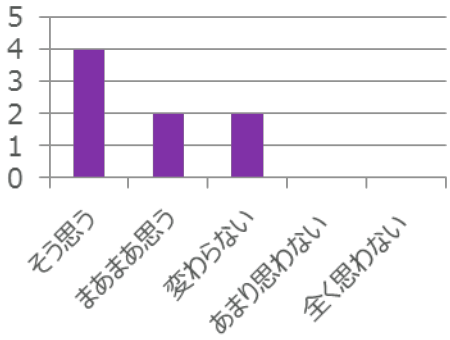

図 4 テスト仕様書に沿ったテストと比べて楽しかったですか？のアンケート結果（チーム戦）

7.2. バグ検出効率の向上

個人戦におけるバグ検出のデータを表 2 に，チーム戦におけるバグ検出データを表 3 へ示す．単位時間あたり個人戦では 5 件，チーム戦では 6.4 件のバグを検出した．

発見されたバグには既知のバグとの重複や，同じ対象をそれぞれの参加者がテストするため，参加者同士でのバグの重複がある．

表 2 バグ検出データ（個人戦）

総発見バグ数	25
一人当たり平均	5
有効発見バグ数	25
A ランク	1
B ランク	5
C ランク	17
確認待ち	2

表 3 バグ検出データ（チーム戦）

総発見バグ数	58
一人当たり平均	6.4
有効発見バグ数	22
A ランク	0
B ランク	5
C ランク	16
確認待ち	1

7.3. バグの効果的な出し方の共有

「ふりかえりの共有のときに新しい発見がありましたか？」というアンケートを取った．個人戦のアンケート結果を図 5 へ，チーム戦のアンケート結果を図 6 へ示す．

このアンケート結果からは，ふりかえりでは何らかの発見があったことが読み取れる．実際のふりかえりの中では，バグをどうやって出したか，現状の仕様がどうやってできあがったかという経緯の説明，それぞれの開発者が思っているバグの傾向など通常共有されない暗黙知が次々と出ていた．

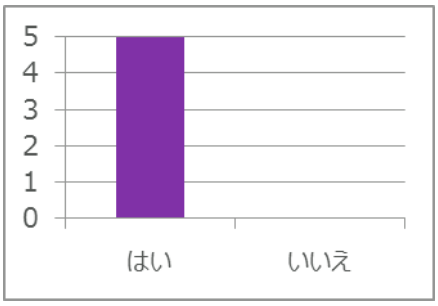

図 5 ふりかえりのときに新しい発見がありましたか？のアンケート結果（個人戦）

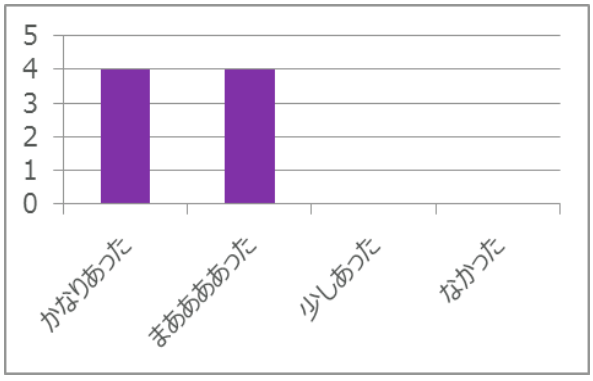

図 6 ふりかえりのときに新しい発見がありましたか？のアンケート結果（チーム戦）

8. 考察

8.1. 課題と解決策

それぞれの課題に対して効果が高かった解決策をマッピングしたものを図 7 に示す．今回は探索的テスト，ゲーミフィケーションの 2 つの施策が上手く絡まりあって，テスト実行のモチベーションアップ，短時間での大量のバグを検知できたという良い結果に結びついたと考える．

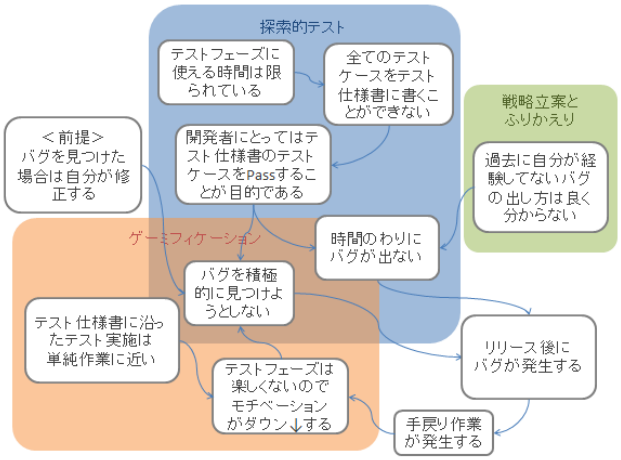

図 7 問題点と解決策のマッピング

8.2. ゲーミフィケーションについて

今回のゲーミフィケーションを用いた探索的テストではどのようなことが起こっていたのかを考察する．テスト仕様書に沿ったテストに比べ，時間が経つのが早かったですか？という，アンケートの個人戦の結果を図 8，チーム戦の結果を図 9 に示す．全員ではないが，半数以上はそう思う，まあまあ思うと回答している．この結果から，時間が早いと感じた開発者は集中した状態であり，ゲーミフィケーションの特徴の一つである『ハマる』という状態，つまり一般的に言われるフロー状態に入っていたのではないかと考えられる．

前述したとおりテスト実行の時間は 1 時間であるが，終了時には一気に疲れがくるとアンケートの感想に書かれていた．この感想もテスト仕様書ベースのテストより集中してテスト実行していたことを裏付けている．

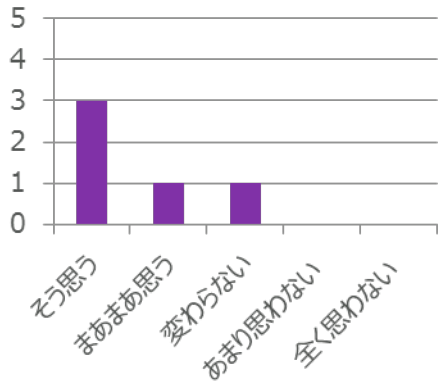

図 8 テスト仕様書に沿ったテストに比べて，時間が経つのが早かったですか？のアンケート結果（個人戦）

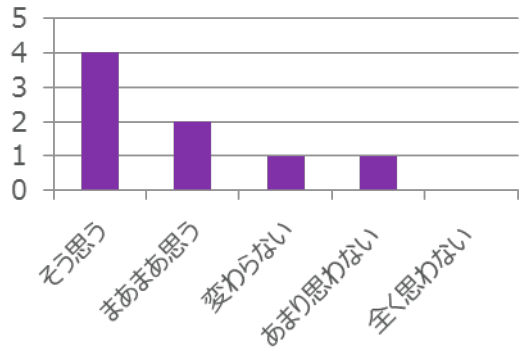

図 9 テスト仕様書に沿ったテストに比べて，時間が経つのが早かったですか？のアンケート結果（チーム戦）

次に『ハマる』状態を作り出す主要因として，3 つの要素を考えた．一つは，同じ時間，同じ機能を対象として，条件を公平にしたことで，純粋にバグを出す能力に焦点を当てたことである．仮に別の機能で同じようなことを実施しても，バグが多い機能に当たったからポイントを取ったという見方もできるであろう．

もう一つは開発中の機能であるため，適度なバグが存在したということである．もちろんプロダクトとしては褒められる状態ではない．しかし自分でバグをどうやったら出せるか考え，その操作することで，実際にバグを発見し，ポイントが手に入るという成功体験を繰り返すことが，楽しいという感覚に繋がっていくと考えられる．仮に，対象となる機能の品質が高く，1 時間で 1 件もバグを見つけることができない場合には，楽しかったという結果にはつながらない可能性が高い．

最後の一つはバグを見つけると褒められる仕組みにしたことである．本来，社内でバグを検出するということは市場にバグを流出させる前に止めることであり，喜ばしいことである．しかし，リリース直前などでは，「なんで今バグを出すんだ！」などと言われた経験がある人も多いのではないだろうか．今回はランクが高いバグを出すことでポイントを手に入れることができるというルールにしたため，バグを出すことは良いことであるという共通認識になり，積極的にバグを見つけるモチベーションにつながったと考えられる．

8.3. 戦略立案とふりかえりについて

チーム戦では戦略立案を取り入れたが，この戦略には，簡易ではあるがテスト分析，テスト設計の要素が含まれている．対象機能の知見を持っている開発者を必ず一人以上チームに配置することにより，その機能の特徴や過去のバグなどを共有することができた．ただし，戦略に沿ってテストをするだけになってしまうと，ソフトウェアを動かしたときの違和感などのフィードバックを見逃す可能性があるので，その点は気を付けて進める必要がある．

ふりかえりでは，バグの現象とその発見に至った経緯の共有に重点を置いた．同じ機能に対してテストしたからこそ，自分には無いテスト観点や，考え方の違いを受け入れやすかったと考えられる．異なる機能をテストしていた場合ではコンテキストが共有されていないため，詳しく説明されてもイメージが湧かない可能性が高い．

したがって，この戦略立案とふりかえりの両方の取り組みが，暗黙知を共有する良い仕組みであったと考えられる．この活動は探索的テストではなくとも使えるため，テスト仕様書ベースのテストでも実施することは可能である．

8.4. マイクロソフトの取り組みとの差異

マイクロソフトのテスト対象は膨大にあるダイアログボックスである．作業としては単純作業であり，そこにゲーミフィケーションを使うことで大人数のエンジニアが自発的に参加し，大量のバグを検出することができた．

一方，本報告で取り組んだテスト対象は特定の機能であり，実施した探索的テストも単純作業ではない．しかし，今回の取り組みとアンケートの結果から創造的な探索的テストにもゲーム性を持たせることは可能であることが分かった．テストの時間や対象を同じにするなど，公平な条件にすることで知的なゲームとなり得るであろう．

9. 課題

9.1. ゲームバランスの調整

ゲームバランスの調整として，二つの問題がある．まずはランクによるポイントの変更を考える必要がある．最終的な目的は A ランクである重要バグを検出することであるため，A ランク発見時のポイントを引き上げる必要がある．ランク毎の数の比率も考慮に入れると，ゲーム的にも一発逆転が可能な 20 ポイント程度が適切だと考えている．

もう一つは既知の不具合に対するポイントの検討である．例えば既知の不具合に対するポイントを 0 ポイントにした場合は，知っている不具合でポイントを稼げてしまうというチートを防ぐと共に，高得点を出すためには過去のバグを確認するという行動を誘発することができる．

9.2. ゲーム性を高めるUIの構築

今回は各チームの状況を見ることができる Web アプリのリーダーボードを用意した．しかし，さらにゲーム性を高めるアイディアを実装することで，エンジニアは楽しくバグを探すことができる．まだアイディアレベルであるが，以下の施策を候補として考えている．

・A ランクのバグを見つけた場合は，他の参加者へ通知する

・時間 10 分前になるとデータが見えなくなる

9.3. バグ報告の簡易化

バグ報告には現象と共に画面キャプチャーなどが必要である．その作業が簡単にでき，バグを見つけ出す思考が止まらないような仕組みを作る必要がある．最終的には検出したバグを精査後に，BTS として使用している Redmine に登録するが，この作業も開発者の負担となっているため，ワンクリックで Redmine に登録できる仕組みも考える必要がある．

9.4. 重複バグへの対策

時間と対象を同一にして複数人でバグ検出をしているため，複数の重複したバグが発見される可能性がある．その点に関してはプロダクトの成熟度合いなどを考慮に入れて，ゲーム実施の人数を絞ったり，テスト対象を変えたりする必要がでてくるであろう．ただし，テスト対象を変えると公平さが失われる可能性がある．

9.5. 通常の開発プロセスへの適応

時間と対象を同一にすることがゲームの公平性を保つために重要であるが，この状況を通常の開発プロセスに適応するのは前述した費用対効果の面からも難しいことが予想される．

アイディアレベルであるが，以下の施策を実施することで，一定の公平性を保ちながら開発プロセスへ適応することが可能になると考えている．

・対象をある開発フェーズで入ったすべての機能とし，どの機能を探索するかも戦略の一つとする．

・チームがどの機能を探索するかを抽選とする．

10. 謝辞

今回の報告において，初めての手法に面白そうだと理解を示し，一緒に取り組んでくれたチームメンバーである的川建史様，藤村浩様，大谷陽介様，小川裕亮様に感謝いたします．またリーダーボードを作成してくれた平井有希様に感謝いたします．また忙しい中，レビューを実施してくれた小楠聡美様に感謝いたします．

参考文献

[1] Wikipedia https://ja.wikipedia.org/wiki/ゲーミフィケーション

[2] Kevin Werbach, 渡部典子訳, 2013, ”ウォートン・スクール ゲーミフィケーション集中講義”, CCC メディアハウス

[3] James bach, Exploratory Testing Explained v.1.3 4/16/03 , http://www.satisfice.com/articles/et-article.pdf

[4] State of Testing Survey http://www.practitest.com/pdf/State_of_Testing_Survey_2013_Japanese.pdf

[5] James bach, Session-Based Test Management, http://www.satisfice.com/sbtm/