2013年度　第 6 回特別講義　レポート

日時	2013 年 11 月 8 日（金）　10:00～12:00
会場	日本科学技術連盟・東高円寺ビル　2階講堂
テーマ	「コンピュータゲーム開発と品質」
講師名・所属	長久　勝氏（国立情報学研究所）
司会	SQiP 研究会第 4 分科会主査　金山　豊浩氏（（株）ミツエーリンクス）
アジェンダ	1. 品質の観点からゲーム産業を俯瞰 2. コンピュータゲーム開発の現状について 3. コンピュータゲーム開発のこれからについて
アブストラクト	コンピュータゲームは、その多くが非常に短期間で消費されてしまうため、ユーザ体験が次回作の購買行動に強く影響を与えます。 ゲームを遊んで面白いことは最重要ですが、動作の不具合や内容の不整合など、論理的に検証可能な品質が十分でないことで、その製品の売り上げだけでなく、次回作以降の売り上げに悪影響が出ます。 企業や開発チームの持つブランドを守り、成長させていくためには、継続的に品質に対する取り組みが不可欠となっています。 今回は、ゲーム開発現場で現在行われている品質への取り組みから、コンピュータゲームの特性から考えられる今後の品質への取り組みまで、ご紹介します。

＜講義の要約＞

◆品質の観点からゲーム産業を俯瞰

ビジネス状況は、2013CESA ゲーム白書によると、国内メーカはハードウェアが好調で、売り上げは 1 兆 4,000 億円である。10 年くらいのトレンドを見ると、ハードウェアとソフトウェアを合わせて 1 兆円、ソフトウェアはその半分の 5,000 億円で、国内と海外が半々といった構造になっており、海外が無視できない状況にある。

上記のようなビジネス状況において、ゲーム業界では QCD がどのように考えられているのかについて紹介する。

「C：コスト」を占める順位は、人件費、広告宣伝費、運営費、開発環境整備費、管理費、製造費などである。いちばん掛かるのはソフトウェアエンジニア、シナリオライタ、デザイナなどを雇用する人件費で、次いではコンシューマに製品を通知するための広告宣伝費である。
次いで掛かるのは運営費で、ネットを使ったゲームが増えるにつれて、サーバを構え管理するケ

ースが増え、ユーザを飽きさせないように、新たなサービス、イベント、アイテムを追加する派生開発の費用が増えている。次いでは、開発環境整備費、管理費である。いちばん掛からないのは製造費で、ダウンロード版ではゼロである。

「D：納期」は基本的に短納期が求められる。コストの大半は人件費なので、コストを抑えるために、短期間で制作することが必要で、同じ内容（アイデア）のゲームであれば、出来るだけ早く販売したものが勝者になる。投資効果が見込める大規模タイトルは十分な投資と時間を掛けて制作することができるが、新ジャンルは出来るだけ短い期間で制作し出来るだけ早く販売することが求められる。

「Q：品質」の要求水準は、ユーザ視点（需要）と事業者視点（供給）で決まる。
ユーザ視点は、ゲームとして面白く、ソフトウェアとしてちゃんと動くことであり、事業者視点は、事業として儲かることである。
コンピュータゲームが評価されるのは、シナリオのよさで、ソフトウェアはシナリオを楽しめるレベルで動くことが求められる。取り残したバグがあった場合でも、統計的に発生頻度が小さく、ユーザへの影響が無視できるレベルにあれば、修正しないことがある。一方、ゲームのデータがセーブできなくなるハングアップなどの深刻な問題を防止するために、プラットフォーマ（プラットフォームを持っているメーカ）の厳しい品質チェックに合格しないと販売できない縛りがある。ソーシャルゲームの品質チェックはかなり緩い。
実際のユーザの視点を知ることは難しい。ゲームを制作した当事者の評価は偏った評価であり、コンシューマに受け入れられるものを評価しているとは限らない。ソーシャルゲームはゲームで遊んでいる間の操作がサーバに送られユーザ動向の調査に活用される。ゲームメーカはユーザ動向を分析し、よく使われている部分を拡張、使われていない部分を削除するなどして、ユーザがより魅力を感じる方向にゲームを進化させ続けることがトレンドになっている。
ユーザ視点の「面白い」を定量的に評価することは難しく、ユーザ動向の指標で代用している。最近では KPI を活用してソーシャルゲームを運用する取り組みがある。ユーザ動向とソフトウェア品質の相関はそれほど高くない。また、ユーザ動向の指標は時系列の増減のみ有意で拡大解釈は危険である。ひとつのメジャータイトルを時系列的に比較することは意味があるが、複数のメジャータイトル（別ジャンル）同士を KPI で比較することに意味はない。KPI で分かることは【儲かるか／儲からないか】なので、これに頼りすぎると、最近のソーシャルゲームのようにカードゲームばかりとなる状況に陥る。
コンシューマ【ゲーム専用機向けの(ROM で提供される)ゲーム】においては、昔は指標を取るすべがなかったので相対的にソフトウェア品質に留意した開発をしてきた。そして、出荷媒体は ROM しかなく ROM の出荷品質がクリティカルであった。今では最初に ROM で出荷しても後からインターネット経由でアップデートできるため、昔よりもクリティカルではなくなった。
ソーシャルゲームはサービス開始時の品質はクリティカルではない。ソーシャルゲームの新ジャンルは、アーリーアダプタの目に付く場所で遊んでもらいながらゲームを進化させ、儲けられるレベルに到達したら、広告宣伝費を掛けてコンシューマに伝えている。

このように、品質に関してはコンシューマ【ゲーム専用機向けの(ROM で提供される)ゲーム】とソーシャルゲームには温度差がある。

◆コンピュータゲーム開発の現状について

〇アーキテクチャ

ゲームの設計がすごいということはない。リアルタイムで動くメインの部分については、タスクシステムという枯れたアーキテクチャが存在し、アドベンチャー、シューティングについては、スクリプトでプログラムを制作する。このアーキテクチャは 10 年以上、基本は変わっていないので、作法を知っている人がきちんと制作すれば品質はそんなに悪くはならない。

最近ではタスクエンジンの論理的なアーキテクチャをパッケージ販売している。以前は非常に高価であったが手頃な価格となり、ライセンスも販売数がある数を超えてから発生するものがあるので、タスクエンジンを使って自分の望むゲームを制作できるようになった。タスクエンジンは知見のある高度な技術者が制作しているので、自分達がゼロから制作するよりも、タスクエンジンを採用する方が投資効果がよい。

〇開発の自動化(ツールの活用)

1)静的解析ツール

ゲーム業界の大手はソフトを外注化して自社ブランドで販売することがある。そのような場合、外注から上がってきたソースをレビューしなければならないが、レビューアを長時間確保することは難しいので、静的解析ツールを活用してレビュー効率を高めている。

2)CI(継続的インテグレーション)

ゲーム開発は、ソフトウェアエンジニア、シナリオライタ、デザイナなどの協業で制作を進めるため、ゲームのシナリオやデザインに何らかの変更があった場合に、ソフトウェアエンジニアがいなくても、ビルドが実行できるように、Jenkins などの CI ツールを活用してビルドを自動化し、最新版で結合試験できる環境を実現している。

〇開発プロセス

開発プロセスを効率化して品質担保の取り組み実施するために Agile を導入している。ゲーム開発は納期が決まっているので、開発プロセスはウォータフォールのように見えるが、ゲームが面白いのかを考えながら制作するので、必然的に反復開発をしてきた。そのような文化と Agile は相性がよく、多くのゲームメーカが Agile を広めたいと考えている。

Agile の取り組みのほとんどは、大規模システムの開発においてプロセスの縛りが厳しい所を緩めれば周りがよくなるということであるが、元々プロセスの縛りが緩い所に Agile を導入して引き締めることは、考えの方向が逆となので、そのことを理解しているかいないかで、導入が成功するか失敗するかが決まる。

Agile が有効か有効でないのかを数学的に考えるために、ソフトウェア開発は問題解決であるとする。そうすると、問題と解答はある空間の座標で表現され、その座標に対する点数を示す評価関数で表現できる。空間を 2 次元だとすると評価関数を f( )として解答=f(問題)となる。

f( )が時間とともに変化しないで固定されている場合は、同じドメインで仕事すればするほど、今まで見えていなかった f( )がはっきりと見えるようになり、どこがピークになるのかを特定できるようになる。このような場合は、Agile を導入する必要ななく、ピークを狙った計画を立案し、実施すればよい。同じ IP で 10 年間やっていて、ユーザもある程度決まっていて、有名なデザイナがやっている場合がこれに当たる。
これに対して、f( )が時間とともに変化する場合は、どこがピークになるのかを特定できない。このような場合は Agile を導入する価値がある。前はシューティングゲームを制作したが、次はロールプレイングを制作するような場合、ソーシャルゲームの新ジャンルの場合などがこれに当たる。

〇ゲームとしての品質
ゲームとして面白いかどうかは、昔はよく分からないことだったが、最近はユーザ動向の統計処理が手軽にできるようになり、統計処理の観点からゲームの面白さを評価する取り組みがある。また、認知科学からの観点からゲームが面白いかどうかについての取り組みがある。
複数ユーザがマップの中でゲームをする場合、ユーザはどの位置で何をしたのかをサーバに吸い上げ、それをヒートマップとして視覚化したものを使って、現状のゲームが設計通りに遊ばれているのかを分析し、マップを改善することに活かしている。

◆コンピュータゲーム開発のこれからについて
〇ゲームソフトウェアの特徴
枯れた論理的なアーキテクチャを使ったゲームエンジンの上に、ゲームを制作する DSL(ドメイン固有言語)が組み込まれており、効率的な制作が行えるようになっている。

〇DSL のうれしさ
DSL を使う利点は、文法の規模が小さいため、DSL のツールを用意しやすいことにある。また、DSL でビジネスロジックを書くことができるので、ゲームの実装(画面を表示してキャラクタを動かすのかといった処理を C や JAVA で書くこと)から離れて、ゲームの内容を形式的に扱うことができる。そして、シナリオの矛盾点を DSL のツールを使って DSL レベルで検証することができる。

DSL を使ってシナリオの矛盾点を見つけるデモ
アドベンチチャゲームのシナリオ進行をモデル検査する例
簡単なアドベンチャーゲームを、LTSA というモデル検査ツールで矛盾点を探すデモが実施された。
一つ目:
プログラムコード中に LTSA を活用するためのモデル式、変換式を出力する検査コードを組み込み、検査コードの出力結果を LTSA に読み込ませると、シナリオ進行が状態遷移図で表示される。これと設計したシナリオ進行の状態遷移図を比較し、状態遷移図からシナリオの矛盾点を特定することができる。
二つ目:
シナリオの透過性(例えば、面ごとに鍵を見つけてドアを開けるといったシナリオにおいて、1面目

で鍵を見つけると、2 面目は鍵を見つけなくてもドアが開けられてしまうといった矛盾）は、あるべき姿（正しい姿）をテストプロセスとして検査コードを組み込めば、テストプロセスとシナリオの出力結果を使い、LTSA で矛盾点を特定することができる。
三つ目：
キャラクタの関係性（例えば、A キャラと B キャラはお互いに知らないシナリオにおいて、A キャラと B キャラがお互いに知っているといった矛盾）も、あるべき姿（正しい姿）をテストプロセスとして検査コードを組み込めば、テストプロセスとシナリオの出力結果を使い、LTSA で矛盾点を特定することができる。

◆質疑応答
Q：日本はゲーム産業が強いと思っていたが、海外で作られたシステマティックなゲームエンジンを導入して仕事を分業している状況に驚いた。昔はソフトウェアエンジニア、シナリオライタ、デザイナが、ひとつのチームひとつの部屋に集まって擦り合わせながら開発していくスタイルであったが、今は会社の中で求められるスキルセットが変わって自分のスキルを活かせていない人がいるのではないかと思った。そして、そんなこんなことをしているうちにゲーム産業が衰退してしまうのではないかと心配になったが、その辺りの状況を教えてほしい。

A：欧米ではサブプライムの時に多くのスタジオがリストラを実施した。リストラされた彼らの多くは、他の仕事に就くのではなく、数人が集まってインディゲームを制作するようになった。
一方、日本ではレイオフされることがシビアではない。会社の仕事は大規模タイトルの一部しかやらせてもらえない状況かもしれないが、昔に比べて家に帰れるようになったため、家に帰ってから自分の好きなインディゲームを制作し、コミケとかプレイストアとかアップルストアとかで販売している人達がいる。こういったインディの中からヒーローが出てきてほしいと期待している。

Q：LTSA などのモデル検査の中で、例えば、最初に戻るとか、もう一回やってみるといったことは、テスト観点でいうとテストデザインに相当すると思うが、シナリオからどこをテストするのか、どのような検査コードで、どのような条件で組み上げていくのか、ということを、どのように設計しているのか教えてほしい。

A：モデル検査に限ったことではなく、ユニットテスト（テスティング FW）の上で自動検査コードを出す取り組みと似ている。
最後まで行ったらフラグに矛盾があるといった経験的に積み上げられるテストケースは、今のツールにアドインしていくことできる。
これに対し、キャラクタ間の関係性の矛盾を検出するようなテストケースは、シナリオごとに用意しなければならない。シナリオライタが頭の中でやっていることを知見化できれば、LTSA のようなモデル検査ツールに組み込むための補助ツールを準備できるかもしれない。

＜講義の感想＞
コンピュータゲーム開発における QCD の考え方、アーキテクチャ、開発の自動化、開発プロセスに

ついて、とても興味深く聞くことができました。
特に Agile の適用のポイントを数学的なモデルでご説明頂き、適用が成功するか／失敗するかを決める前提条件を明確に理解することができました。現場に適用する時の方針が見えてきました。
また、ゲームエンジンの上に、ゲームを制作するDSL（ドメイン固有言語）が組み込まれているアーキテクチャは、ソフトウェア開発の未来を見ているようで、エンタプライズ系、組み込み系においても、注目される技術になることを予感できました。今後の展開が楽しみです。
司会者の金山様、貴重な講演をしてくださいました長久様に深く感謝いたします。