2013年度　第5回特別講義　レポート

日時	2013 年 10 月 11 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	「リーン開発と品質」
講師名・所属	天野　勝氏（株式会社　永和システムマネジメント）
司会	鷲崎　弘宜氏（早稲田大学准教授）
アジェンダ	1. リーンから、リーン(ソフトウェア)開発へ 2. リーン開発で重視する「品質」 3. リーン開発 4. リーン開発の原則 5. リーン開発はアジャイル開発か？ 6. リーン開発的なソフトウェア開発の分析 7. リーン開発を取り巻く世界
アブストラクト	トヨタ生産方式の研究から生まれた、「リーン」という概念が、ソフトウェア開発の現場でも活かされるようになってきている。しかし、リーン開発は原則集にしか過ぎないため、プロセスとしての形を持たず、理解、認識がしにくいという特徴がある。 今回は、書籍『リーン開発の本質』で語られている原則を、世の中の実例と対応させながら紹介していただく。また、リーン開発の原則の一つである「ムダをなくす」の一つである「タスク切り替えのムダ」について演習により体験して体験していただく。

司会者：鷲崎氏からの講演への着目点のご案内

「アジャイル開発とかリーン開発を適用したいんですがどうしたらいいのでしょうか」という質問をよく受けますが、「質問が違うのじゃないですか？」と答えます。開発方法を適用する事が目的ではなく、「あなたの中にどういう問題があって、それを解決するためにはどうすれば良いかを一緒に考えましょう」とアドバイスします。

本日の内容も、そのまま使ってみようというのではなく、根底の中にある考え方はどういうもので、それが皆様の中に抱える問題であるとか、仕事のやり方であるとか、マネジメントの仕方であるとかとどういう関係があるだろうかということを考えながら聞いていただければ幸いです。

＜講義の要約＞

【1】リーンから、リーン(ソフトウェア)開発へ

・リーンの語源

「リーン（LEAN）」の意味は、その直訳の通り「筋肉質で贅肉がない」とう意味ですが、筋肉だけという意味ではなく「無駄がない」という意味が当てはまるでしょう。機敏に動けて力も出せるというイメージです。

・リーンの起源
１９８０年代世界の景気が停滞する中,、ヨタだけは大きな伸びを示していました。この生産方式にMIT のジェームズ・P・ウォマック氏、ダニエル・T・ジョーズ氏らが着目し研究する中でトヨタにおける作業の仕方を表す言葉として提唱されたものです。この時代からトヨタ生産方式が世界に知れ渡っていた事は素晴らしいことです。

・TPS（トヨタ生産方式）の家
トヨタ生産方式を俯瞰するものとして、書物「ザ・トヨタウェイ」に示された TPS の家が有名です。後述するトヨタ生産方式のリーン思考の原則を家の構造に例えて図示したものです。
(http://jp.fujitsu.com/family/lsken/activity/work-group/09/abstract/pdf/2009abstract-11_tps.pdf)

・リーンソフトウェア開発
トヨタ生産方式に基づくリーンの思考をソフトウェア開発へつなげる考えをメアリー・ポッペンディーク氏らがまとめて、2003 年に「リーンソフトウェア開発」として出版されたのがソフトウェアへの考え方の導入の起源です。それ以来いくつかの書物が出版されてきましたが、開発現場からは遠い内容でした。再度現場への適用の考え方をまとめた「リーン開発の現場（カンバンによる大規模プロジェクトの運営）」がこの１０月に出版されます。

【2】リーン開発で重視する「品質」
・品質の定義
品質の定義は一般的には様々な規格または研究者によって定義されていますが、現在は「ユーザの満足を最終ゴール」とする事で概ね世界的に合意が得られています。そしてこの考え方は、石川馨氏他による「消費者志向」の考え方の影響が大きいのです。

・SQuBOK(ソフトウェア品質)とリーン開発との接点
この両者の接点は以下の事項の相性が良い事と考えられます。
「プロダクトの特性が顧客のニーズに応える事で満足を提供する」「品質は収益と結びつく」「不備（障害や誤り）から免れる」「よりコストがかからない」そしてこれらは TQM にもつながる事項ですし、品質がよければ儲かるという事にもつながります。

・納品後のソフトウェアの機能の利用度
2002 年のアジャイルカンファレンスのジム・ジョンソン氏の報告によると、納品後の機能の利用度はせっかく苦労して開発し納品しても約半数の機能は全く使われていないし、実際に使われるのは全体の１／３、よく利用されるのは１／５です。つまり多くの無駄な機能を納品しているわけで

す。ただし、受託して作る側から言うと使われようが使われまいが作ることにより収益を得られるので、表面上は満足しているように見えますが、「役に立たないものを作っていた、ゴミを作っていた」と考えると何とも辛い思いとなります。これは、契約形態が人月いくらという形態である事が問題でありビジネス構造の重要な問題です。

・システムは陳腐化する
従来、開発においては要求を早く固定する事を良しとしてきました。しかし昨今の要求事項の価値の変化速度の激しい時代では、開発しているうちにすでに提供価値の陳腐化が進みます。つまり要求を早い時期に固定すると、その製品の完成時期にはすでに価値の多くが劣化してしまうことになります。
これを防ぐためには、要求の固定を遅らせる事が必要になります。しかしながらこれでは開発期間が短くなり現実的では有りません。この相反する条件を満たす手段として「保守開発」という考え方があります。つまり、保守しながら小刻みで要求を反映し続ける事で提供価値を維持し続ける事が可能となります。これを実現する手段としてリーン開発の考え方が必要となります。

【3】リーン開発
・リーン開発とは
しばしば開発手法と間違われますが、手法ではなくトヨタ生産方式を手本としたより良いソフトウェア開発を
考える思考ツールです。書物「リーンソフトウェア開発」においては、7つの原則と22の思考ツールが紹介されています。主旨は、プル生産方式の考え方とそれの現場への適用のし方です。
また、書物「リーン開発の本質」では、現場への適用の考え方の中で、「アイデアをいかに早くお金に換えるか」という視点で紹介されています。

・前提とするビジネス環境
リーン開発が注目されだした理由は、ビジネス環境の変化の影響が大きいのです。前述したように、市場に供給したものが市場ニーズに合致する事が成功条件でありますし、そのためにはニーズの変化にいち早く対応する事が必要となります。つまり、要求の固定を遅らせ短周期での開発を実施する必要があります。そして更に難しい条件として「自らが供給したシステムによって利用者のニーズそのものも変化を受ける」という複雑な連鎖もあります。このようにシステム開発には多様なリスクが潜んでいます。

・ニーズに合う商品を提供するために
これらニーズに関する影響を最小に抑えるためには、市場ニーズを把握してからそれを実現する周期の短縮を図る必要があります。そのためには、時間軸上のムダを省いてニーズの獲得から商品提供までを如何に素早く行うかが命題となります。これの実現のためにリーンの考え方が必要となるわけです。

【4】リーン開発の原則

リーン開発を語る上で難しいのは、そもそも思考なので抽象論である事です。よって、自身の経験を付け加えて説明します。
・リーン思考の原則（リーン開発の本質）
リーン思考の原則は、以下の7つです。これら全体に共通する考え方は「良い状態を保つための思考フレームワーク」です。
・原則1：ムダをなくす、・原則2：品質を作りこむ、・原則3：知識を作り出す、・原則4：決定を遅らせる、
・原則5：速く提供する、・原則6：人を尊重する、・原則7：全体を最適化する
以下に各原則を説明します。
・原則1：ムダをなくす
生産工程での7つのムダが定義されています。その中で代表的なムダは以下です。
在庫のムダ、作りすぎの無駄：使われない機能のモジュールはこれに相当します。
動作のムダ：掛け持ちで複数作業を実施する場合、タスク切り替えにムダが生じます。
では、ムダの定義は何でしょう。ここで大切なのは、顧客視点のムダと開発側視点のムダは異なる部分がある事です。お客様視点のムダとは、「お客様にとって価値のないもの」であり、例えば開発側では必須であると思い込んでいる品質確保の手段としてのデバッグは、お客様視点ではムダと見なせます。こう考えると目指すのは、「開発側のムダを無くし」、「お客様視点のムダを如何に減らすか」ということになります。そしてこのムダ取りにより開側のムダをゼロにし、その分をお客様の付加価値に回せば大きな生産性向上が望めるのです。
ここで開発側のムダを排除する方法を紹介しましょう。この比較評価にはバリューストリームマップが有効です。製品の要求分析からお客様へのリリースまでの各工程を時間軸で評価します。お客様目線で見た場合の価値とムダを工程ごとに時間軸で書き出だしてみます。ここに紹介した例では、工程トータルで価値は2時間40分、ムダは6週間と4時間になりました。さて、全工程に要した時間とお客様の価値を比較して、それを効率とすると何と1%と見なせます。つまりお客様視点から評価すると99%はムダなのです。

【タスク切り替えのムダを体験するワーク】
開発組織のムダを体験するために、タスク切り替えのムダをワークで体験してみましょう。手元にお配りした表に倍数を書き込んでいくワークです。1回目は上の表は2，4・・・と記入していき30秒経ったら下の表に9，18・・・と30秒間記入していきます。2回目は1分間で上の表と下の表を交互に倍数を記入していきます。さて、1回目と2回目の比較でどちらが多くの数字を記入できたでしょうか。多くの方は1回目の方が多くの数字を書き込めたと思います。つなり、このような単純な作業でも上下の表に書くというタスクを切り替える事で効率が落ちる、つまりタスク切り替えのムダが判る訳です。

・原則2：品質を作りこむ
ここで大切なことは、作って最後で直すのではなく直さないように作ることです。例えば、各工程の中で小刻みなチェックを入れる事で大きな手戻りを無くすのもこれに相当します。また、ムダなコー

ドを無くすには、「テスト駆動開発（TDD）」の適用も効果的です。テストコードをまず書く事で実行可能なユニットテストを作りながらのプログラミングが可能となります。早い段階でチェックするという意味では、ピアレビューもこれに相当します。チームの中の身近な者にまずチェックしてもらう事で早い段階で欠陥を直し手戻りの最小化が見込めます。

・原則3：知識を作り出す
通常の開発プロジェクトですと当該プロジェクトが完了時点で振り返りを実施し、何を継続すべきか、何を改善すべきか、そして今後のチャレンジすべき事は何かについて次のプロジェクトへの申し送りする場合が多いのですが、これですと学習サイクルが大きいために学びのタイミングも遅くなります。そこで、日々の作業の中でこのサ
イクルを小刻みに回しながら進めれば、開発完了時期を待たずに学びながら進めることができます。学習のみではなく、例えば、PDCA を1週から4週程度の期間で回せば、つまりアジャイル的なイテレーションですが、例え失敗しても早く小さな段階で失敗を発見できアクションを実施できるわけです。

・原則4：決定を遅らせる
大胆な考え方です。従来、開発プロセスの常識として仕様は開発早期に決定すべきといわれてきました。ところが、先に述べたように取り巻く環境の動きの速さが顕著になってきた昨今では、早く決定すると出荷時にはすでに陳腐化してしまうのです。よって、決定のデッドラインを決めた上でぎりぎりまで決定を遅らせることにより盛り込める知識が増え、ぎりぎりまで磨けることとなります。
これを助ける手段として、「継続的インテグレーション」という言葉があります。開発中のソフトウェアを常に出荷可能な状態に保った上で新たな機能を追加していくやり方です。これによりビッグバン結合試験が可能となり、テスト期間の短縮によりコスト削減にもつながります。

・原則5：速く提供する
前にも述べましたが、短期間での開発が可能となれば、要件の決定時期を極限まで遅らせる事ができ、相対的に要件の実現を早くできたのと同じこととなります。これにより、より多くの付加価値を付ける事が可能となるわけです。

・原則6：人を尊重する
トヨタ生産方式において常に触れられる事に「人を尊重する」という言葉があります。これは、「人は機械ではないのだから各人が自発的に動く事が大事である」という思想に基づくのです。これの手助けをするのがリーダーです。逆に、全てリーダーが指示し管理する組織を作ってしまうと、「言われたから行動する」とか、逆に「言われないから行動しない」といった事になり、リーダーの責任問題となります。つまり、リーダーがボトルネックになってしまいます。こうなるとだれもリーダーには成りたくないといった消極的な組織となります。逆に、メンバーが自発的に行動し、メンバー間も自立的に協働するという仕組みを作るのがリーダーの仕事です。
次に、「見える化」についても重要なキーワードです。異常を見えるようにし、自然にアクションをとるような仕組みです。決して管理のための道具にしてはいけません。例えば、個人の査定に使う

のは最悪で、こうすると嘘の情報があがるようになるので破綻してしまいます。
見える化のサイクルは現場のチーム内で回さないとうまくいきません。つまり、可視化した事によりチーム内の担当者が問題に気付き、チームで改善策を検討し実施する事が大切です。もしもリーダーが先に問題に気付いてしまうと、リーダーの改善策が現場に伝わらないか、もしくはやらされ感に陥ってしまうのです。
見える化の道具として「タスクボード」があります。タスクの状態を未実施（ToDo）・実施中（Doing）完了（Done）の3つの状態に分けて配置します。各タスクが今どの状態にあるのかを明示する事で各人の状況が常に見えます。ここで大切なのは「だれかが困っているのか」が見えれば「即座に助け合う風土をつくる」事です。このタスクボードを導入しただけで生産性が30%も向上した事例もあります。
また、バグの見える化ツールとして「バグレゴ」があります。バグが出るとレゴブロックで自由に形を作り並べるのです。レゴにはバグのチケット番号とバグ概要を付箋紙に書き込み貼っておきます。例えば、影響度大のバグであればお手上げの人形を載せたりし、一目で状況が見えるようにするのです。この施策によりバグを隠さなくなったとの事例があります。もしもこのボードがバグで埋まってしまって置く所がなくなれば、一旦試験を止めてバグ解決に集中しようと宣言します。この取り組みは、ゲーミフィケーションの考え方が入っているとも思われます。つまり、普段の活動の中に小さい目標管理とか小さい報酬を入れて仕掛けを継続していこうもしくは自発的に行動する仕組みを維持しましょうという考え方です。

・原則7：全体を最適化する
各工程をつなぐプロセスが有る場合、先のプロセスに塊（仕掛品）ができてしまう場合があります。つまり滞留が生じてるわけです。この滞留は無駄なわけで、これをスムースに流れるようにしましょうという考え方です。まさに TOC（Theory of Constraints ： 制約理論）の考え方です。
最近、この状態をタスクボードに入れて分析する動きがあります。制限付きのタスクボードと言いますが、各工程の仕掛品の量に制限を設けることにより、制限値以上に滞留をさせないという考え方です。タスクの状態を未実施（ToDo）・実施中（Doing）完了（Done）の3つの状態に分けて配置したものです。後工程のタスクが完了しない限りいくら前工程からタスクを送り込んでも、在庫（仕掛品）Ready のタスクが増えるばかりで完了タスクは増えないので、全体としては価値がない事となります。では、先の工程のタスクが制限値に達した場合、自分の工程
では何をやるかと言うと、他のネックとなっている工程のタスクを手伝いに回るというものです。実際には、自分の担当以外のタスクも実施できることが前提にはなりますが、これができると全体としての生産性はあがります。つまり、各担当が自分の担当のみ頑張るのではなく、全体を見渡してうまく流れている事を確認しながら進めるのが大事です。例えば、仕掛のタスクがある状態で前工程のタスクについて変更要望があった場合、その仕掛タスクはやり直しになる場合も考えられます。まさにムダです。

【5】リーン開発はアジャイル開発か？
・アジャイル開発とは

ソフトウェア要件の理解からリリースまでのサイクルを素早く回すための開発手法や技術の総称です。

・代表的なアジャイル開発手法
アジャイルという言葉が生まれたきっかけは、2001 年 2 月に Kent Beck らにより宣言された「アジャイルソフトウェア開発宣言」によります。この内容は、当時軽量級と呼ばれていた開発手法でもちゃんとした結果を出している事を宣言したもので、各開発手法の共通点をまとめてアジャイル宣言としたものです。つまり、個々の手法を明確に示すものではなく、活動といった方が当てはまります。
広義のアジャイルと狭義のアジャイルの定義がありますが、狭義では前述した宣言文に署名しているかどうかです。リーン開発の著者である Bob Charette 氏、Mary & Tom Poppendieck 夫妻はこの宣言に署名をしていませんので、そういう意味では、リーン開発はアジャイル開発ではないと言えます。ただし、個々の取り組みを見ると類似のものがありますので、広義にはアジャイル開発といえます。これらの議論はあまり意味がないのですが、各人が現場に帰った際に実際何が役に立つかを見極めて対応してほしいです。
注意すべきは、われわれは開発手法を議論するのが目的ではないので、アジャイルかそうでないかは議論の的ではありません。「開発現場で本当に役立つやり方は何だろう」という議論が大切で、それを議論する中でそれぞれの手法を参考にして考える事が大切です。そしてその中で、「収益を上げよう」とか「組織をどう考えよう」という場合にリーンの考え方にヒントが多いし、「チーム運営をどうしたら良いか」という場合にはスクラムの考え方が役立ちます。また XP は「テスト駆動開発」とか「継続的インテグレーション」とか「ペアプログラミング」といった、実際の現場での技術的手法として参考になります。
大切なのは「自分たちが使いこなせる事」であり、皆さんが「現場で役立つやり方は何か」を考えてほしいのです。これら手法は提唱者の関心事でまとめたものに過ぎないからです。

・アジャイル開発とリーン開発の関係
Developers Summit 2013 において、川口恭伸氏が発表した「Agile and Lean」の資料の紹介です。
（ http://sssslide.com/speakerdeck.com/kawaguti/agile-and-lean-from-altitude-12-000-feets ）
アジャイルと日本の製造業とリーンの関係が示されており、リーンは TPS の影響を受けており、アジャイルに対しては影響を与えているという図です。他の手法もリーンで表現しようとすれば可能な事が多いのです。つまり様々な提唱がされているが、考え方では共通するものが多いのです。

【バリューストリーム分析と7つの原則の理解を深める演習の紹介】
・バリューストリーム分析
自分の仕事をバリューストリーム分析で実際に分析してみようというものです。開発の各工程を時間軸で並べてみて、その各工程に実際の価値のある作業はどれ位で、どれだけの無駄な時間があるか、待たされている作業は無いかを明示するものです。この図を描こうとすると一人では描けません。このため社外も含めた関連者皆で話をしなければならない、これにより様々なムダが見え

てきます。簡易に試すのであれば、自分の前後の仕事を描いてみるのも良いでしょう。

・7つの原則の理解を深める
そもそも原則なので日常作業の至る所に眠っている事項です。皆で「うまく言っているところ」、「ここは何かおかしいな」という気付きが大切です。例えばレビューのやり方です。「従来なぜ最後にまとめてレビューしていたのだろう」という気付きが得られました。日次ミーティング・朝会、振り返りの会とかでどんな原則が含まれているかを議論してみるのが良いでしょう。これらの説明は IPA から出ているものがあります。

【6】リーン開発的なソフトウェア開発の分析
今回は説明は省きましが、下記事項について議論してみるのが良いでしょう。
・バリューストリーム分析、7つの原則、日次ミーティング、ふりかえり、リファクタリング

【7】リーン開発を取り巻く世界
・「リーン思考」を身に付けると待ち行列が見えてくる！
仮説を説明してきたがこれを考えていますと、何が動いていて何が溜まっているかが見えてきます。例えばレビューの実施を考えると、レビュー結果により何度も何度も修正する事で、その前後に待ち行列が増加している事が見えてきます。これはレビュー?修正のサイクルが過度に回りすぎている可能性があるからです。その場合、やはり作る段階で対策を打つ必要性が見えてきます。

・タスクボードの例（人毎の作業を把握）
ToDo に個人単位にタスクを固定し助け合いが無いと、タスクが溜まる者と空きの者とが出てきます。そこで人に固定しない共通タスクのカテゴリを設けると、空きができた者は共通タスクを実施する事で全体が最適化されます。この場合、まずはだれでもできるタスクを切り出すのですが、育成も利用して徐々にだれでもできるタスクを増やして行くことが改善につながります。

・ファーストフード店の座席稼働率の問題
座席が込んでいる場合、注文をとる前に座席を確保する方が居ます。これが増えると食事をする前から座席の占有時間が増えてしまい本来の座席の稼働率が落ちます。改善するには注文数に制限を設け、座席の空きを確保しておく、つまり後工程の処理能力を基準にして前工程の処理能力を落とすことで座席の稼動率が向上し最適化されます。

・リーンスタートアップと学習サイクル
予算が少ない中で、いかに身軽に素早く導入を進めるかです。簡易なプロダクトをターゲットにして回してみてその結果から学びを得てふくらまして行くやり方です。ここでプロダクトを作る段階でアジャイルの考え方が注目されています。リスクを考えればスモールスタートでやるのが良いがその場合にもアジャイルの考え方が役に立ちます。

◆質疑応答

Q1. バリューストリーム MAP についてですが、この MAP を作って、価値を生む仕事がどの位で実際どの位時間がかかるかは考える場合、MAP の一番下の欄の待ち時間がありますが、これは悪であると言う視点だと思います。それがリーン開発の視点と考えてよろしいでしょうか？
A1. 要は早くリリースしたいと言うことです。早くリリースするためにはどうするかと考える事が大前提です。そのために待ち時間が合ったら遅くなります。そこでまずは待ち時間を削減する事からはじめれば楽だと言う事です。TOC と同じでボトルネックを探してやり易い所から取り組んで行くのが良いでしょう。

Q2. 原則4の「決定を遅らせる」についてですが、部下とか回りのメンバーに影響する仕事をしている場合に、自分の決定を遅らせると回りに影響が出てくるし、出荷間際に作りこんだものは回りにも影響が出てしまって品質も保てないのではないでしょうか？そこで解釈は以下で宜しいでしょうか。方針の決定、及び主要な機能の改造追加はしておき、出荷に向けたチューニングは出荷間際に実施する。という事でしょうか？
A2. 方針も遅らせるほうが良いです。リリースに間に合えば良いのでそのためにどこまで遅らせられるかが大事です。ゴールできる範囲で如何に決定を遅らせるかということです。

Q3. タスクボードを試したがうまくいきませんでした。人の常として自分のノルマを守ろうという考えが強いからだと思いました。最終的に関連者全体がうまく回るという事を伝えて普及させるにはどうしたら良いでしょうか。
A3. タスクボードの使い方のみを学んでもうまくいきません。タスクボードに貼るタスクを皆で考えることが大切だからです。見積もり、プランニングについて関連者で議論し考え方を共有する事が大事なのです。
自分はこう思うが他人は考えが違う場合でも「こうするとうまくいく」というコミュニケーションが図れる事が良いのです。タスクボードを監視のツールにしてはいけません。どう知らしめていくかと言うと、タスクボードの使い方が若干甘かったからかも知れません。叱る役を置くのも良いでしょう。母親のように成長につながるように叱るのがコツです。

Q4. バリューストリームマップの書き方で「価値」の行、「ムダ」の行の書き方について教えてほしい。
A4. 特に決まった書き方は無いのですが、「正味の作業」と「付帯作業、待ち時間」を表せば良いです。
それにより価値とムダを明示できることが目的ですので。

◆感想
まず、冒頭で本日の講演内容についてご案内いただきました鷲崎氏の発言に強く同調しました。ソフトウェア開発に関わる思想・手法につきましては、キーワードを耳にすればまずは紐解かないと気が済みませんし、書物を読めば自らの課題はさておき試して見たくなるのが人情ですし、そこで間違いが起こる事を再認識いたしました。
開発の思想、手法については様々な場で議論もしましたし、書物も手にしたのですがどうも「腹に

落ちない」状態が長く続いておりました。本日の講演のキーワードはリーン開発なのですが、トヨタ生産方式をはじめとし、アジャイル開発等、昨今のキーワードまで含めてその経歴・関連について紐解いていただき、霧が晴れた思いです。特に「思想」と「手法」の違いを明示していただいた事で、現場の課題と適用すべき思想、方針が見えてきました。
司会者であられます鷲崎様、貴重な講演をしてくださいました天野様に深く感謝いたします。