Scrum Gathering Tokyo 2011 Day1 レポート

前編：変化を促すためのモデル

技術部　アジャイル開発センター
伊藤　宏幸

はじめに

去る２０１１年１０月１９日（水）に、野村コンファレンスプラザ日本橋にて開催された**「Scrum Gathering Tokyo 2011」**のカンファレンス（Day1）に参加してきました。

http://www.scrumgatheringtokyo.org/sgt2011/index.php?id=2

このカンファレンスで発表された内容、およびそこから私が学びとった点について、今月と来月に渡りレポートいたします。

今月号ではまず、Henrik Kniberg さんによる基調講演についてレポートいたします。

目次

基調講演『Everybody wants Change, but nobody likes to Be Changed』

基調講演は、『Scrum and XP from the Trenches』（邦題：塹壕より Scrum と XP）の著者である**Henrik Kniberg さん**による、**「アジャイル開発を導入するために、どのように上司・顧客・チームメンバーといったステークホルダーに働きかけていけばよいのか」**をテーマとした講演でした。

【Henrik さんのプロフィール】

スウェーデンの Crisp 社で、アジャイルおよびリーンのコーチをされています。

また、２００９年から２０１１年まで、Agile Alliance の理事をされていました。

以下、講演のポイントを整理します。

1. アジャイル開発導入の障壁としてのステークホルダー

アジャイル開発を「新たに」導入しようとする場合、次のステークホルダーを説得する必要があります。

- 上司（manager）　：　組織の決定権の保持者
- 顧客（customer）　：　プロダクト（製品・システム）の所有者・使用者
- チームメンバー（team）　：　実際に作業を行うメンバー

アジャイル開発の導入に際しては、このステークホルダーの人々がいわゆる「抵抗勢力」となります。したがって、このステークホルダーの人々をいかに説得できるかがポイントとなります。

2. 説得ではなく自発的変化を促す

それでは、アジャイル開発の導入のような「変化」を促すために、どのようにステークホルダーの人々を説得していけばよいのでしょうか？

Henrik さんによると、**「そもそも説得することは無理」**とのことです。説得するのではなく、変化を「自発的に」受け容れてもらえるように働きかけること、すなわち**自発的変化を促すこと**が重要とのことです。

具体的な例で考えてみましょう。もし６歳の子供に部屋の片付けをさせたい場合、どのようなアプローチが考えられるでしょうか？

まず考えられるのが、次のようなトップダウン型のアプローチです。

- 「部屋を片付けたら褒美をあげる」といって説得する
- 「部屋を片付けろ」と叱りつける

これらのようなトップダウン型のアプローチでは、子供に部屋を片付けさせることはできません。なぜならば、部屋を片付ける理由と動機付けが足りないためです。理由や動機がないのに強制だけをしても、子供は納得しません。(Henrik さんは、こうしたアプローチを**「父親モード」(father mode)** と称していました。)

上記のようなトップダウン型のアプローチではなく、次のようなアプローチで、子供に理由を示し、自分から部屋を片付ける気持ちを起こさせることが有効とのことです。(Henrik さんは、こうしたアプローチを**「コーチモード」(coach mode)** と称していました。)

- 部屋の主として、君には部屋を片付ける責任があるのだと説明する
- 部屋をきれいにし続けてくれると、親としてうれしいと言う
- 部屋を片付けることで、モノを見つけやすくなるといったメリットがあることを説明する

3. 変化を促すためのモデル

上記の６歳の子供の例は、変化の促し方を、これまでのありがちな「変化を強制する方法」から、「自発的変化を促す方法」へと変えています。ステークホルダーにアジャイル開発を受け容れてもらう場合も同様で、変化の促し方を自発的変化の方向へと変える必要があります。Henrik さんは、この自発的変化を促すための考え方として、次のような**「変化を促すためのモデル」(a model for change)** を紹介されていました。

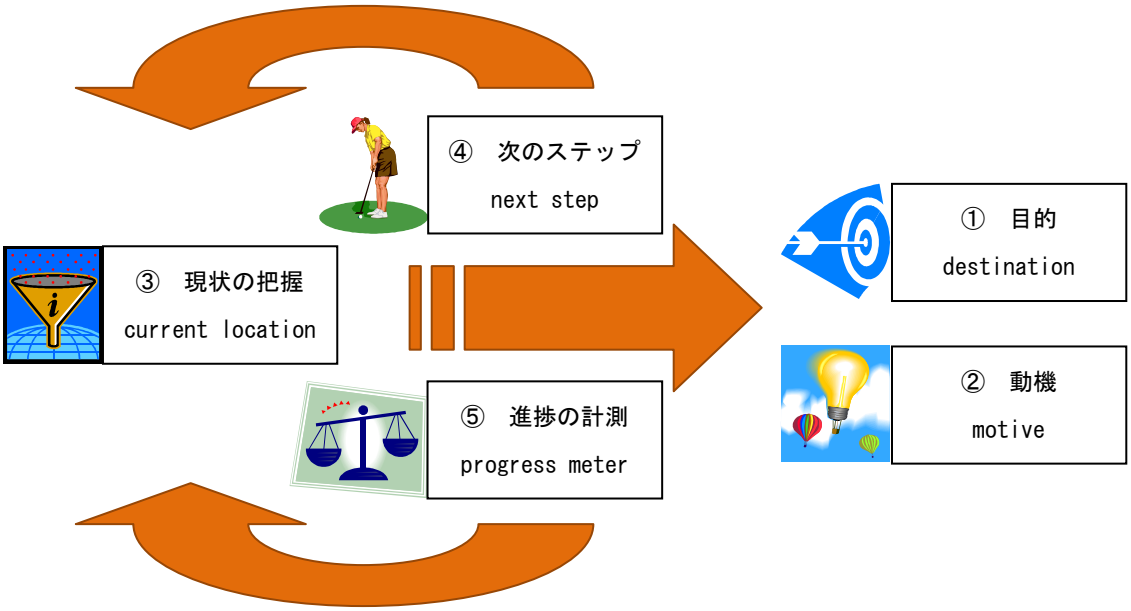

図１：変化を促すためのモデル

※Henrik さんの図を元に、伊藤が加筆・修正

このモデルの概要は、以下の通りです。

（１） まず最初に、変化のゴールとしての「目的」（図１の①）と、自発的変化を促すための「動機」（図１の②）を設定します

・人間は、目的・ゴールが明確であれば、そこに向かってがんばろうというエネルギーを自発的に生み出すことができます。

・人間は、他人から変化させられることを嫌います（自分は自分自身でコントロールしたいと思うものです）。したがって、自発的な変化を促していけるよう動機付けをすることが必要です。

（２） 次に、「現状の把握」（図１の③）を行い、「目的」に到達するための「次のステップ」（図１の④）を考え、実行します

・現時点の状況・組織・人間に合わない行動を取ろうとしても、抵抗を受けるなどして失敗してしまいます。したがって、まず現状を把握し、現状に即した行動を取ることが肝要です。

（３） しばらくしたら「進捗の計測」（図１の⑤）を行い、改善の必要があれば都度「次のステップ」を考え実行します

大目標を達成するために、小目標を設定し、進捗を確認しながら少しずつ改善・調整していく。これを繰り返し実行することで、徐々に大目標に近づくことができる（図２）。これが、このモデルのポイントです。

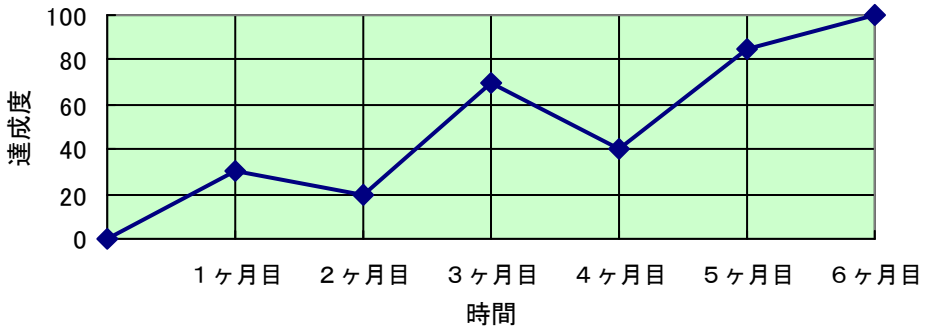

図２：「変化を促すためのモデル」に基づく変化の実現イメージ

このモデル（フレームワークやアーキテクチャと読み替えても良いかもしれません）を理解し、適切に利用することで、個人と組織に対して、アジャイル開発という自発的変化を促すことができます。

4. 変化を促すための戦略

Henrik さんは、上記の「変化を促すためのモデル」に加えて、変化を実行・促進・駆動するために、次のような**「戦略」(strategy)** を利用すると良いとのことでした。以下、各戦略について説明します。

4.1. 戦略１：抵抗を最小限に抑える方法を選択すべし

Henrik さんによると、アジャイル開発を導入する方法として、次の３つが考えられるとのことです。

- ビッグバン方式（Big Bang） ： 最初から一気にアジャイル開発へ転換する方法
- 増分方式（Incremental） ： 徐々にアジャイル開発へ転換していく方法
- パイロット方式（Pilot） ： パイロットプロジェクトで採用し、そこから全体に広めていく方法

これらのうち、最初のビッグバン方式は、変化の量などの影響が大きく、抵抗にあうリスクが大きいとのことです。したがって、**２番目の増分方式か３番目のパイロット方式が、抵抗が少なくて好ましいとのことです。**

また Henrik さんは、上記とは別に**「野火方式」(wildfire method)** という方法を提案されていました。この方法は、現場でアジャイル開発に興味のありそうな人を「焚きつけて」、現場レベルからの自発的変化を促すという方法です。

具体的には、以下のようなことを行います。

- 現場レベルで、アジャイル開発に興味のある人を見つける
- 興味のある人を見つけたら、
  実際にアジャイル開発をやるよう周りから「焚きつける」
- 「焚きつけられた人」がアジャイル開発をやり始めたら、
  アドバイスを送るなどして全面バックアップする

上記のような人を徐々に増やしまとめていくことで、アジャイル開発の動きが燎原の炎のごとく組織全体に広まっていくだろう、ということです。

4.2. 戦略２：”Us and Them”問題を避けるべし

例えば、アジャイル開発について紹介すると、「確かにアジャイル開発にはメリットを感じるのですが、上司が嫌がってやらせてくれないんですよね…」という言い訳をする人がいます。

自分のためではなく、**「彼ら」のせいで**できない。

このように、他者に責任を転嫁することで、結果として変化を受容できないということ

があります。これを**“Us and Them”問題**と言います。

この“Us and Them”という考え方は、組織内外でのお互いへの無関心・対立・敵意を生むこともあるため、アジャイル開発の導入に限らず注意が必要です。（これは、来月号で紹介する Jeff Patton さんの講演のテーマでもありました。）

この“Us and Them”問題の解決方法として Henrik さんは、ステークホルダー全員を集めて話をする場を設けることを提案されていました。そこでは、how（どうやるか）ではなく what（ゴール・目的・何を成し遂げるべきか・何を作るべきか）を全員に説明し、どうすればその what を実現できるのかを一緒に考えてもらうようにします。具体的には、「なぜできないのか？」と他者を責める形から、「どうすればゴールに到達することができるか？」を一緒に考える形へ、発想の転換を促すように会話を進めます。このようにすることで、ステークホルダーの参加意識や関与を生み出すことができ、“Us and Them”の壁を取り払うことができます。

4.3. 戦略３：現状を「見える化」すべし

Henrik さんによると、現状を適切に把握するためには、現状を「見える化」することが重要とのことでした。具体的には以下の方法を紹介されていました。

（１） タスクボードの活用

タスクをホワイトボードなどの上に「見える化」する**「タスクボード」**の活用により、以下の効果が得られます。

- スプリント※1が機能していない状態（dead sprint）を検知できる
- 作業全体の流れを「見える化」できる
  （スプリント内の「カンバン」として機能する）
- プロジェクトの障害とそのエスカレーション状況を「見える化」できる
  （課題とその解決状況を把握できる）
- ボードを複数チームで共用することで、複数チームの作業状況を「見える化」できる
- 異なるタイプのタスクを「見える化」できる
  （同じ「タスク」であっても、機能・ユーザストーリー※2・バグなど、タスクの種類を識別しやすくできる）

※1 スクラムにおける反復のこと。１スプリントは２～４週間程度が推奨されている。

※2 アジャイル開発における要求管理手法の１つ。顧客の要求を自然言語で、カードなどに簡潔に記述する。見積もりや、後述するバックログのベースとなる。

・**「技術的負債」(technical debt)**[※3]を「見える化」できる

（２） ベロシティ込みの計画立案

「ベロシティ」(velocity)とは、１スプリントあたりに実現できるストーリーポイント数[※4]のことです。このベロシティを組み込んだリリース計画を立て、バーンダウンチャート[※5]などによりスプリントの進捗を確認することで、進捗の遅れなどの問題を「見える化」することができます。

（３） デスマーチの検知

次の２つの方法で、デスマーチを検知することができます。

・直感（gut feel）

各ユーザストーリー・機能を実現できそうか否かについて、メンバーに**「直感」**で点数をつけてもらうことで、実現が難しいものを見つけることができます。

例えば、確実にできると思うものを５点とし、１～５点の範囲でメンバーに直観で点数をつけてもらいます。集計の結果、特に点数が少ないユーザストーリー・機能は、メンバーが実現困難と考えていると言えるため、何らかの対策が必要ということになります。

ゲーム方式で直感的に行うことで、メンバーの本音を引き出しやすい方法です。

・ベロシティによる空約束や無理な確約の検知

上記（２）のベロシティ込みのリリース計画をもとに現状のベロシティを計測することで、メンバーの空約束や無理な確約(overcommitment)を検知することができます。

4.4. 戦略４：現状維持を含めた選択肢を提示して比較させるべし

例えば、ウォーターフォールを採用している組織でいきなりアジャイル開発手法を導入しましょうとだけ言っても、ステークホルダーは普通抵抗するでしょう。しかし、「現状維持」としてのウォーターフォールと、その「改善案」としてのアジャイル開発という形で、各々の選択肢を一緒に比較・評価できる形で提示すると、ステークホルダーの理解を得やすくなります。なぜならば、現状を元にすることで問題点や改善点について相対比較がしやすくなり、結果としてアジャイル開発のメリットが伝わりやすくなるからです。

※3 将来的に開発を妨げかねない技術的な問題のこと。具体的には、行き当たりばったりのコードや、手抜きなアーキテクチャなどを指す。

※4 ユーザストーリーを見積もるための架空の単位。人日などの絶対値ではなく、ユーザストーリー間の相対値で表現するため、より実情を反映した見積もりとしやすい。

※5 スクラムの進捗管理図法の１つで、スプリントでのバックログの残量をグラフにプロットしたもの。残作業が日に日に減っていくイメージから、「バーンダウン」と名付けられた。

例えば、次のような選択肢を提示できると、ステークホルダーもアジャイル開発を選択しやすくなります。

現状維持 （ウォーターフォール）	改善案１ （厳密なアジャイル・スクラムの採用）	改善案２ （部分的なスクラムの採用）
仕様がなかなか決まらない。	・システムを少しずつ作り、動作を確認しながら仕様を明確化していくようにする。 ・顧客がプロジェクトチームに常駐し、いつでも仕様を確認し合えるようにする。	・システムを少しずつ作り、動作を確認しながら仕様を明確化していくようにする。 ・作成途中のシステムをいつでもさわれる環境を顧客に提供する。（→常駐しなくても良い。）
開発が所定の期間内に終わらない。	プロジェクトの全期間を１回 １ ヶ月のスプリントに分割し直し、毎月少しずつ成果を出していくようにする。	まず１ヶ月のスプリントを試し、できそうならば続けてみる。
手動や行き当たりばったりのテストが多く、システムのバグが多い。	**CI（Continuous Integration、システムの継続的統合）**と**TDD（Test-Driven Development、テスト駆動開発）**とを導入する。	まず xUnit で単体テストを整備するようにする。
今のままでは困る！	いきなりこれは難しい…	これならばできるかも…！

4.5. 戦略５：やり直せる実験をすべし

最初からアジャイル開発を全面導入すると抵抗が大きいですが、「失敗しても大丈夫な範囲」で「少しだけ」アジャイル開発を試してもらい、うまくいかなければ中止するという方法であれば、あまり抵抗なく試してもらえるのではないでしょうか。この方法を Henrik さんは、**「やり直せる実験」（reversible experiment）**と称していました。

例えば、１スプリントだけスクラムを試してもらい、続けられそうならばそのまま継続、ダメならば１スプリントで止めるという方法を採るのです。この方法ならば、失敗しても１スプリント（≒１ヶ月）だけで済むため、ステークホルダーも試しやすいでしょう。また、実際に試してみて、ステークホルダーがその価値を体感してくれれば、その後のスクラムの本格採用を受け容れてくれやすくなるでしょう。

このように、方法論を押し付けるのではなくて、ゲーム感覚にアジャイル開発手法を試

してもらい、ステークホルダーにその価値を体感してもらうようにすることで、徐々にアジャイル開発手法への移行を促すことができます。

人間は、一度試してみたことでなければなかなか受け容れられないものです。逆に言えば、経験したものであれば受け容れやすくなるということです。

4.6. 戦略６：例を示すべし

「論より証拠」、「百聞は一見に如かず」と言います。人間は、言葉で説得してもなかなか受け容れてくれないものですが、実際に効果のある事例を示されると受け容れてくれるものです。

例えば、何も知らない人にいきなり「CI サーバを入れましょう」と訴えても、「時間が足りなくて…」などと言われて断られるのがオチです。しかし、実際に CI サーバを動かして、バグ検知に役立つことを示せばどうでしょうか。そうすれば、ステークホルダーも理解を示してくれて、CI サーバの導入に同意してくれるでしょう。

例を示すことは大事です。

4.7. 戦略７：変化の費用対効果を示すべし

アジャイル開発の導入に実際に価値があることを費用対効果で示せると、ステークホルダーを説得しやすいでしょう。

例えば、もしチームメンバーが１ヶ所に集まって作業する「コロケーション」（co-location）を行えばチームのベロシティが５０%増加しリリースを２ヶ月早められるというようなことを示すことができれば、ステークホルダーはコロケーションの導入という変化を受け容れやすくなるでしょう。

4.8. 戦略８：メトリクスを利用すべし

Henrik さんから、メトリクス（metrics、測定基準）を利用することで作業状況が改善する例の紹介がありました。

まず、週毎のベロシティ、**“Feature burn-up chart”**（≒バーンダウンチャートの逆で、進捗を右肩上がりに示すチャート）およびリリース計画を示すようにしたことで、チームのベロシティが徐々に改善したとのことです。また、**「サイクル時間」（一連の作業に要する時間）**を示すようにしたことで、サイクル時間が徐々に改善したとのことです。

上記の『4.3. 戦略３：現状を「見える化」すべし』とも重なるところがありますが、**状況をメトリクスで「見える化」することで、メンバーのやる気や関与を引き出すことができます。**

4.9. 戦略９：「承認」より「赦し」を求めるべし

これは、事前にステークホルダーに承認を求めてから行動するのではなく、試してみてダメだったらその時に謝りましょうという意味です。つまり、常に承認をもらってから動く上意下達のチームではなく、必要なことを自主的に考えて行動する「自己組織化」されたチームとして行動するようにしていきましょうということです。

例えば、必要な情報を全てドキュメントにまとめる必要はあるでしょうか？もしかしたら、必要になった際の連絡先だけ示せば十分かもしれません。連絡先だけ示せばよいかどうかは試してみなければ分かりませんが、もしＯＫであれば大量のドキュメントを作成せずに済みます。もしドキュメントが欲しいと言われれば、その時に初めて作成すれば良いわけです。

自発的変化を促すためには、そのための土壌が必要です。

4.10. 上記のいずれの方法でもうまくいかなかった場合

GTD（Getting Things Done）※6の提唱者である David Allen さんによると、人生には次の２つの問題しかないとのことです。

- やりたいことは分かっているけれども、どうやれば良いのかが分からない
- そもそも自分が何をやりたいのかが分からない

もし上記の戦略のいずれを試してもうまくいかなかったのであれば、個々の方法論以前に、「そもそも自分が何をやりたいのか」が分からなくなっているのかもしれません。したがって、もう一度次のことを自身に問いただすと良いとのことです。

（１） なぜ自分は変化を欲したのか？
　　変化の目的が何であったのか、もう一度自身に問いただしてみましょう。

（２） 自身がその変化を望んだ動機は何だったのか？
　　同様に、変化を望んだ動機（≒上記（１）の目的）を、もう一度自身に問いただしてみましょう。

また、『The Psychology of Computer Programming』（邦題：プログラミングの心理学）などの著者として有名な Gerald Weinberg さんによると、問題は理想と現実との間のギャップから生じるとのことです。したがって、以下のようなアプローチも有効とのことでした。

（１） 現実を見直してみる
　　実は十分状況が改善してきているにも関わらず、自分自身がそのことに気付いて

※6 仕事の整理術の１つ。頭の中にある作業を紙などにリストアップ・整理・優先度付けし、できることから順次片付けていく方法。複雑な作業を同時に行う知的労働者に向いている。

いないだけなのかもしれません。もう一度、現実を見直してみましょう。

（２） 理想を見直してみる

もしかすると、自分自身で理想像を高く設定しすぎていて、必要以上の変化を要求しているだけなのかもしれません。不必要な完璧主義などに陥っていないか、もう一度自身の理想を見直してみましょう。

（３） 現実と理想のギャップを埋める

現実と理想とを見直してみて、それでもやはりギャップがあることが分かれば、その時に初めてギャップを埋める作業をすれば良いのです。現実と理想とを適切に把握している分、ギャップを埋める方法もより妥当なものになるでしょう。

もしうまくいかなければ、そこから「去れば」良いのです **(Law of 2 feet)** [※7]。努力してダメだったならば、やり直せば良いのです。

5. 実践例

それでは、上述の「変化を促すためのモデル」や各種の戦略は、実際にはどのように利用していけば良いのでしょうか？Henrik さんから、次のような自らの実践例の紹介がありました。

5.1. スタート～最初の対応

Henrik さんは、「旅行が好きなのだけれども、家族と離れ離れにはなりたくない」、「可能であれば、家族と一緒に半年かけて世界一周旅行をしたい」と考えられたそうです。

では、どうすればこれを実現できるか？Henrik さんは、「変化を促すためのモデル」に従い、まず**「仕事を減らす」**という目的を設定されたそうです（図３の①）。またそれと同時に、以下の３つの動機を設定されたそうです（図３の②）。

・家族と過ごす時間をもっと増やしたい
・旅行を計画する時間をもっと増やしたい
・（旅行の柔軟性を高めるために）ネットに繋がなくても良いようにしたい

次に、変化のために具体的に何をすれば良いか？「変化を促すためのモデル」に従い現状の把握を行ったところ（図３の③）、以下のような状況だったそうです。

※7 Harrison Owen さんが提唱されている、グループによる問題解決方法である Open Space Technology（OST）の原則の１つで、もし自分がそこに何も得られるものも貢献できるものもなければ、どんなタイミング・状況であってもそこから去り、もっと好きなところへ行きなさい、ということ。

・仕事が非常に多い
・週に１０～１５時間、メールに時間を割いている
・月に１２～１６日、客先を訪問している

そこで、「客先訪問を月に２～３日にする」という小目標を新たに目的に追加し（図３の①）、最初の対応として、「全てのプロジェクト・案件を一覧にする」というステップを実行されたそうです。（図３の④）

進捗の計測については、以下の３つの方法で実施することにされたそうです。（図３の⑤）
・自動的に顧客訪問時間を計測するようにする
・毎週振り返りを行うようにする

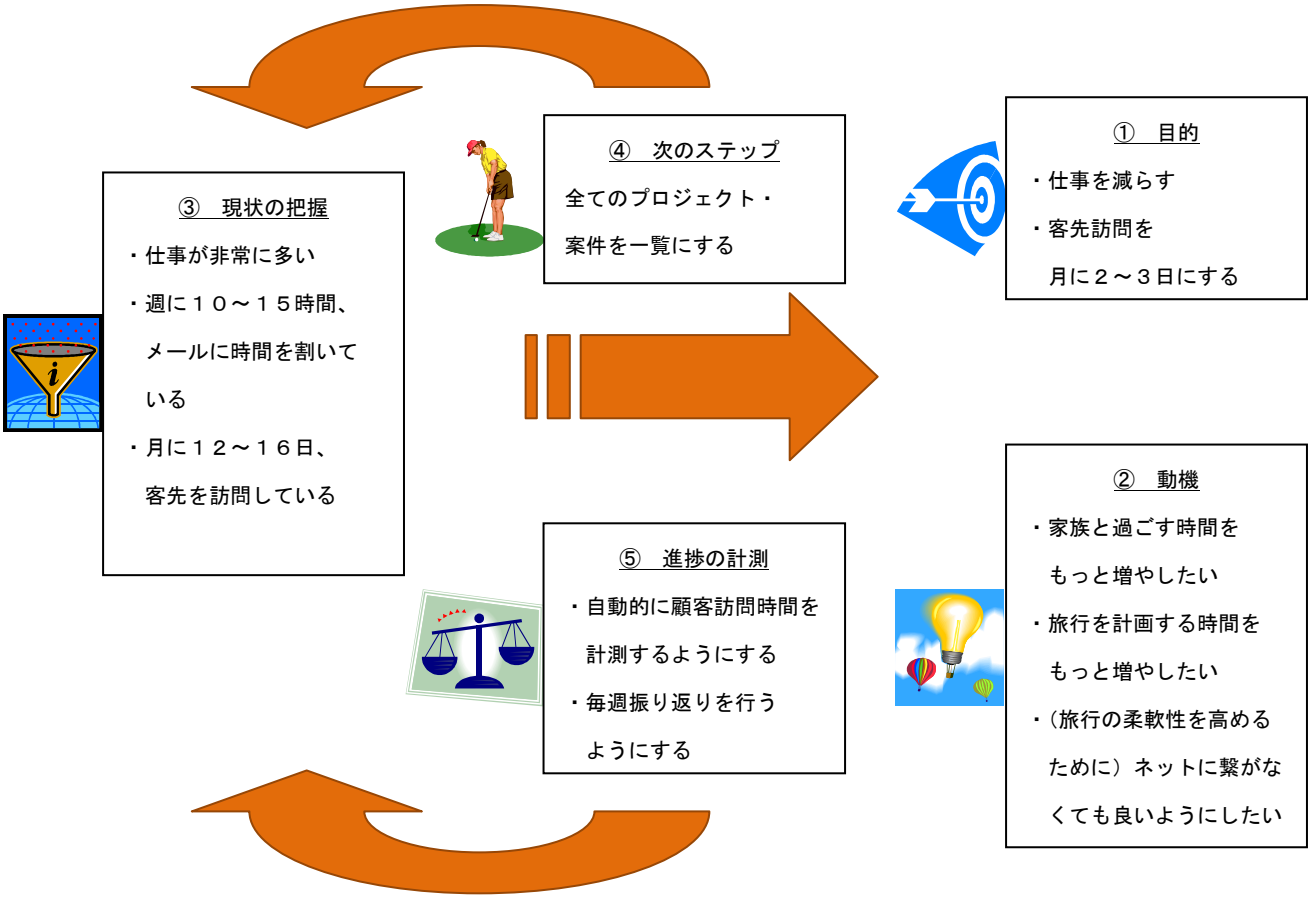

図３：最初の対応時

5.2. 次の対応－仕事の仕分け・削減

上記の「全てのプロジェクト・案件を一覧にする」というステップを実行後、次の対応として、リストアップしたプロジェクト・案件の中から、８０～９０%を削減できるものを見つけるというステップを実行されたそうです。(図４の④)

具体的には、以下のような優先度付けフィルタを用いて、リストアップしたプロジェクト・案件を以下のように仕分けされたそうです。

【優先度付けフィルタ】

・負荷が予測できるもの
・義務的なもの
・楽しくて面白いもの
・旅行の予定にあうもの
・市場の価値や利益を満たすもの

【仕分け後のプロジェクト・案件】

・終了もしくはペンディング　：　対応不要
・外注または委譲できるもの　：　肩代わりしてもらえる
・自動的に期日までに終わるもの　：　あまり意識しなくて良い
・継続実施していくもの　：　対応要
・新しく受けたもの　：　対応要

上記の仕分けによって、仕事量を減らすことができたそうです。

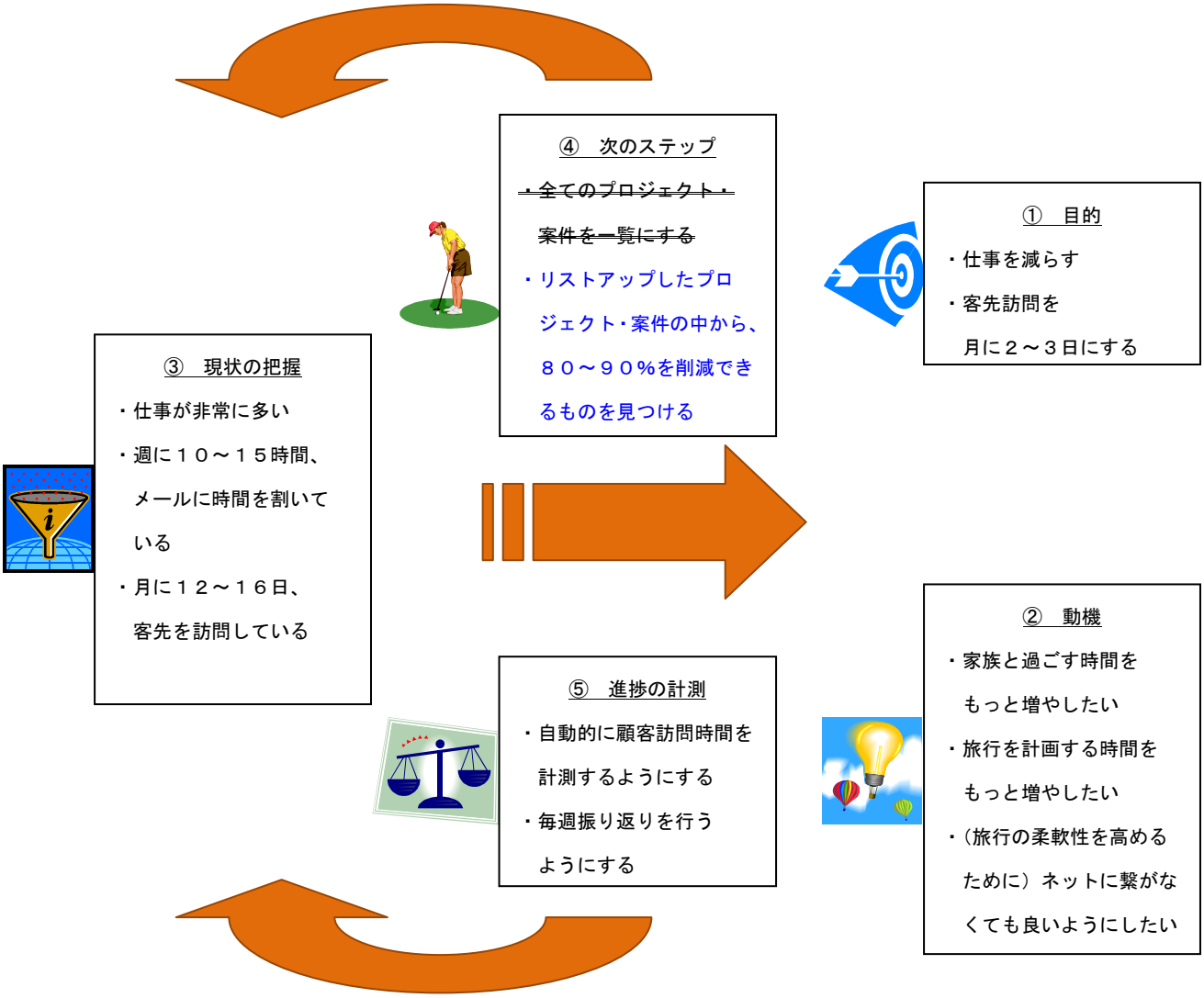

図４：次の対応時（仕事の仕分け・削減）

5.3.　次の対応－メールに関わる仕事量の削減

上記の仕事の仕分け後、次の対応として、メールに関わる仕事量を削減するというステップを実行されたそうです。（図５の④）

具体的には、「メールに割く時間を週１～２時間とする」という小目標を新たに目的に追加し（図５の①）、メール作業時間を自動計測・チェックし（図５の⑤）、メールに関わる仕事量を徐々に減らすよう改善を繰り返されたそうです。

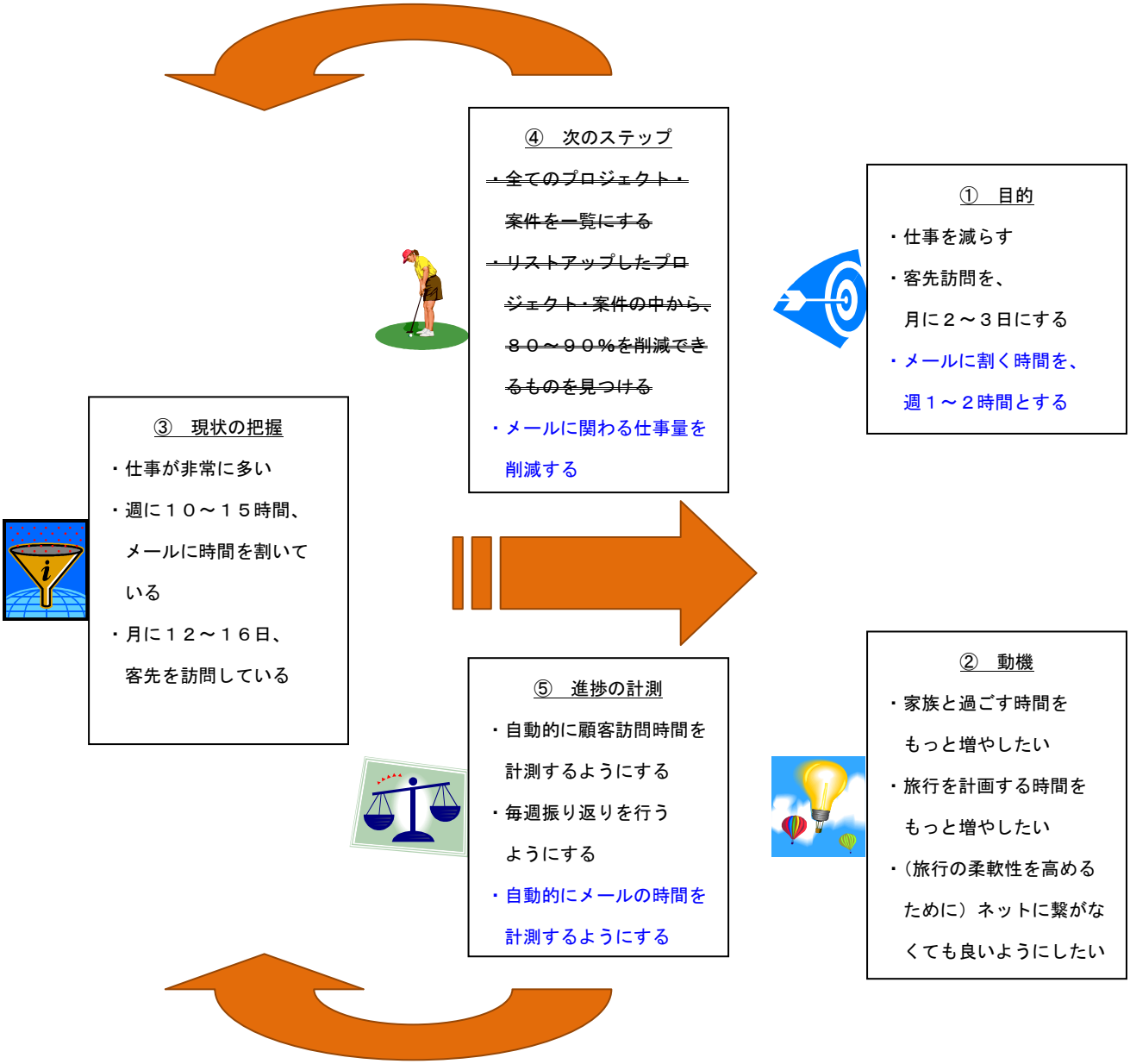

図５：次の対応時（メールに関わる仕事量の削減）

5.4. 一連の対応後

仕事の削減や効率化をある程度達成し、その後も色々と発生する障害を何とか片付けていかれていたそうですが、それでも全ての仕事を予定していた期日までに終わらせることはできなかったそうです。そこでやり方を変えて、**家族を連れて世界中を旅行しながら仕事をすることにされたそうです。**

確かにこの方法ならば、仕事が残ってしまっていても、家族と一緒に世界中を旅行することができますね。

6. 結論

最後に Henrik さんは、ステークホルダーに変化を促すためには以下のことが重要であることを、結論として示されていました。

（１） 外部から無理に変えようとするのではなく、自発的に変わっていくように動機付けすることこそが大事

具体的には、以下の手順が役に立つとのことです。

・「見える化」を進めて、変化する理由を示す
・変化する方法を、少しずつはっきりと示す
・変化に対して、サポート・激励・フィードバックを与える

このように、**懸念や不安を与えずに動機付けをすること（Motivate anybody without fear）**が必要とのことでした。

（２） 「見える化」は、全てに適用しなくても良い

「見える化」は、プロジェクトにとって重要なポイント、またはメンバーの動機付けに役立つところにのみ集中して適用すれば良いとのことでした。

（３） 問題と不安とを混同しないこと

問題と不安とを混同することで、真の問題にたどり着けないことがあります。そのため、まずメンバーの不安を取り除いて、本来の問題に集中できるようにします。その後に、本来の問題を解決できるようにメンバーを導いていくようにすると良いとのことでした。

7. 所感

変化を促すためには、強要するのではなく自発的変化を促すことが必要であること、そしてそのためには「変化を促すためのモデル」および各種の戦略を用いると良いこと。Henrik さんの論旨は、非常に明快で力強く、そして示唆に富むものだと感じました。

私自身も過去のソフトウェア開発プロジェクトで、変化の強要が役に立たないことを何度も経験し、どう変化を促せばよいのかについて悩んできました。悩んだ結果、マズローの欲求段階を参考に、説得する相手を承認し、変化を納得してもらうようにするのが良いのかなと思うに至り、自分なりに実践していましたが、その方法がある程度正しいのだという裏づけが得られたのは自信になりました。

また、“Us and Them”問題の解決や「見える化」の推進など、いかにチーム内の協力を阻む障害を発見して取り除くかについても非常に細かく考察・整理されていて、学ぶところが多かったです。

変化を促すためには、正しい考え方を理解し、適切に用いること。Henrik さんから教えていただいたことは、現在の私自身の仕事を改善していく上での大きな軸となっています。

8. 参考資料

Henrik さんの資料は、下記に公開されています。

http://dl.dropbox.com/u/1018963/projects/2011-10-19%20Japan%20Scrum%20Gathering/Change.pdf

前編のおわりに

来月号では、Jeff Patton さんによる“Us and Them”問題の解決方法、および Yahoo!JAPAN 社をはじめとする各社の現場レベルでのアジャイル開発の取り組みについてレポートいたします。