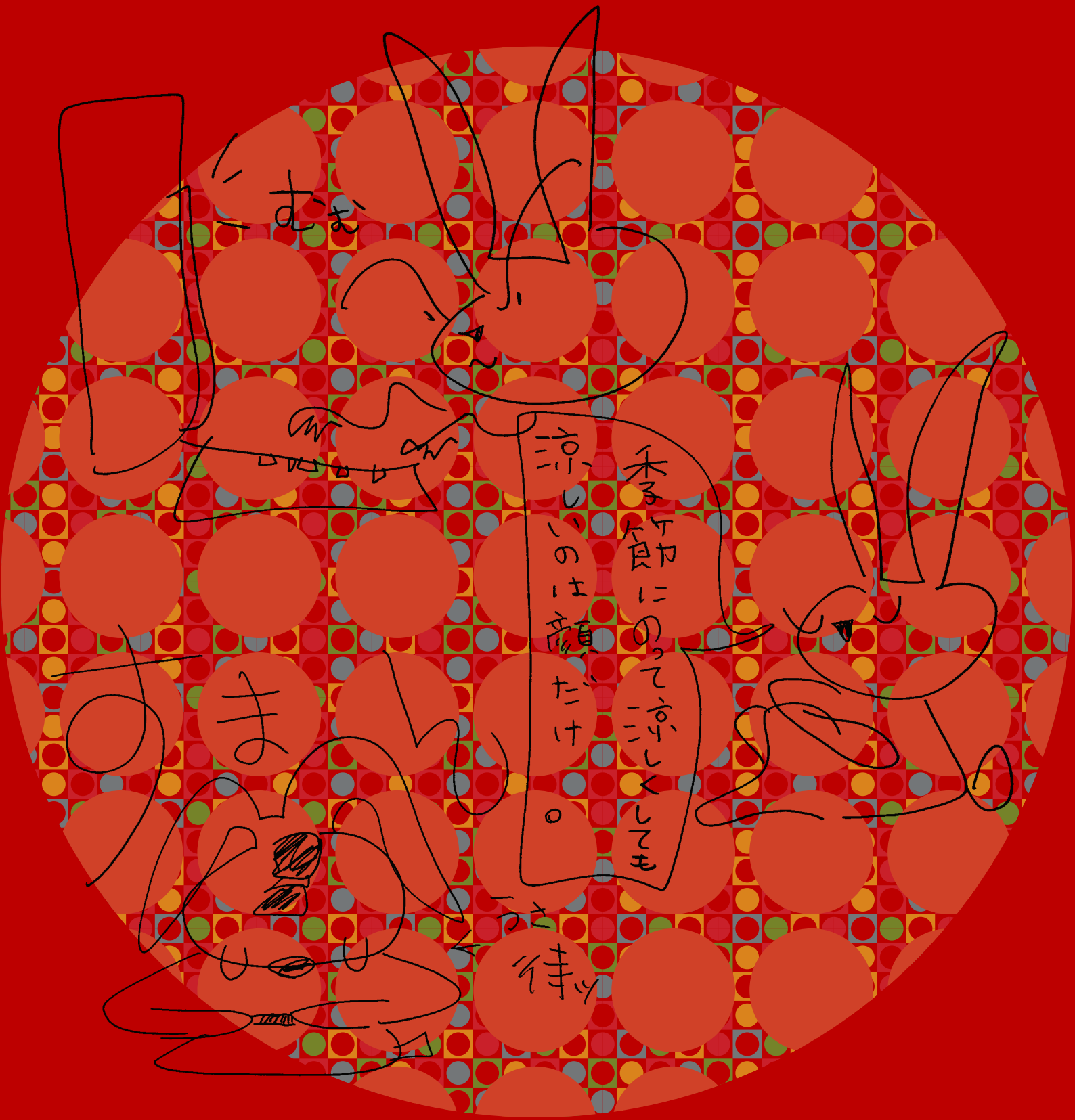
イーエム・ゼロ

[EM ZERO] 2008年12月 其の弐



EM ZERO Vol.2 コンテンツ

スクラム導入事例から学ぶプラクティス適用の極意

セントラルソフト株式会社
林　栄一
HAYASHI Eiichi

初めてのアジャイル

私の勤める会社のシステム開発部門は、準委任契約による要員提供と請負開発を行っています。責任をもって主体的に開発を行うための請負体制を発展させようというのが、部のスタンスです。今まで私の会社では、本格的なアジャイルプロセスを適用して開発を行ったことはありませんでした。私がスクラムマスター認定セミナーを受講したことをきっかけに、進行途中の開発プロジェクトにスクラムを適用することになりました。

ホテルのバイキングモデル

さて、私はとても食いしん坊なので、ホテルのバイキングでは目についたものを片端からお皿に取り、つい食べ過ぎてしまいます。自分の体調を考慮して食材を選び、和食か中華か洋食か方向性を決めて、適度な量をお皿に取ったほうがおいしく、体にも無理がない食事になるはずです。しかし、つい和食も中華も洋食もごちゃまぜのお皿を作ってしまうのです。

このようなことは、新しい手法を適用するときにも起きがちだと思います。アジャイルをはじめとする新しい手法では、多くのプラクティスが紹介されています。開発プロジェクトのお皿にすべてのプラクティスを乗せようとしてしまうことはないでしょうか。

目の前の状況に見合ったプラクティスを選択し、状況に合わせてカスタマイズしていく姿勢が大切です。いわゆる現物現場主義のようなものです。ほとんどの場合、同じチームで同じ難しさの同じようなプロジェクトを組むことは二度とありません。一度しかないものには、自分の頭で考えて対処していくほかはないのです。

ではこうした状況では、新しいプラクティスにどのように向き合っていったらいいのでしょうか。本記事では「スクラム」という手法の導入事例をテーマに考えます。

スクラムとは

スクラムは1990年代初頭に提唱された、反復プロセスを基調としたアジャイルプロセスの一種です。スクラムは、複雑なことを成し遂げるために逆説的にあえて細い管理をせず、チームによる自己組織的マネジメントの発生を促進します。そうすることで飛躍的なパフォーマンスを成し遂げるというコンセプトの、いわばチームマネジメントのフレームワークと言える方法論です。詳しい解説は参考文献をご参照ください。

スクラムの流れ

スクラムの流れを大まかに示すと**図1**のようになります。

スクラムのロール

スクラムのロールは次の通りです。

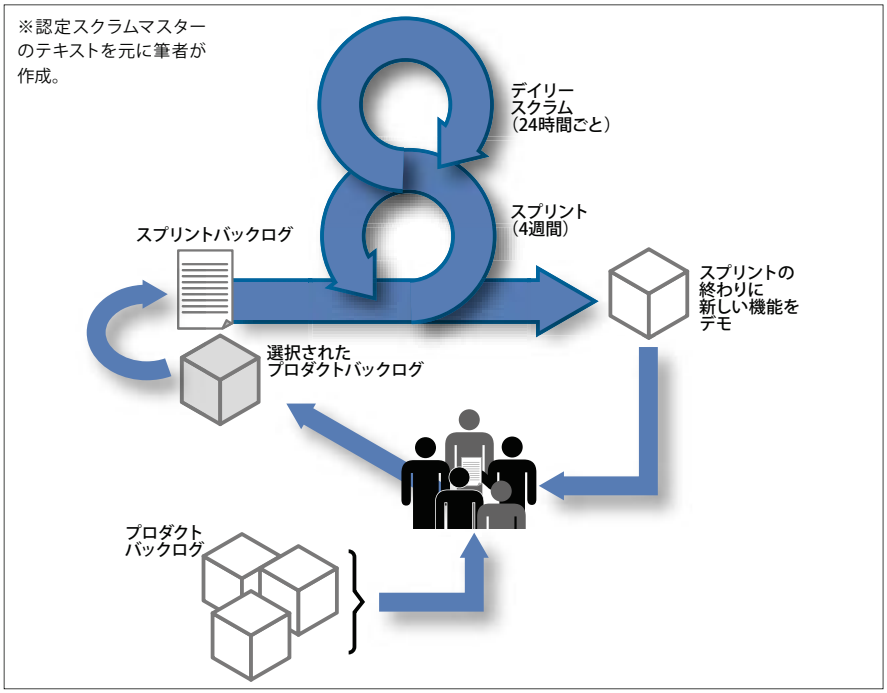
・スクラムマスター：外部との折衝を行う。チームを雑音から守る役割。ミーティングをファシリテートする、スクラム推進のキーパーソン。筆者が担当。
・プロダクトオーナー：業務の視点で、ROIからプロダクトバックログの優先順位を管理する。お客様との打ち合わせを中心的に行ったI君が担当。
・スクラムチーム：開発を行うメンバー。スプリントにコミットしている。5名ほどのスキルがバラバラなメンバーによるチーム。

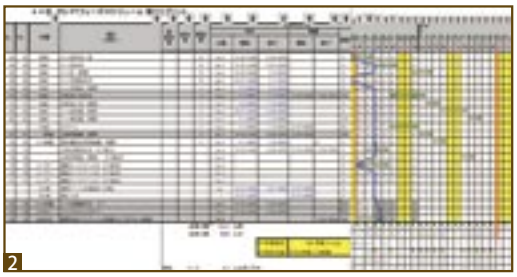
スクラムの主要なプラクティス

スクラムの主要なプラクティスは次の通りです。

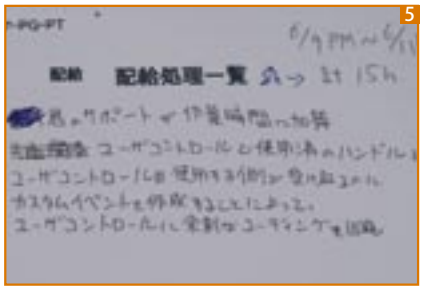
・スプリント：1ヵ月のイテレーション、

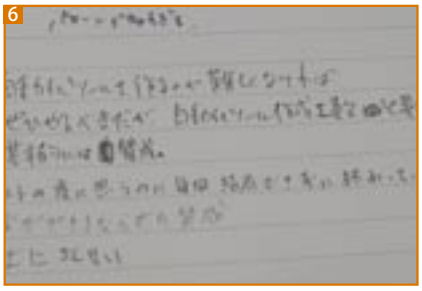
■図1　スクラムの流れ





■写真1　見積もりポーカーの様子
■写真2　見積もりカード
■図2　スプリントバックログ





■写真3　タスクボードによるタスク管理の様子
■写真4　バーンダウンチャート
■写真5　タスクカード
■写真6　555のカード

いつでも納品可能な状態を維持する
・スプリントプランニング：スプリントバックログを作成するミーティング
・デイリースクラム：毎日定時に行う10分ほどのミーティング
・スプリントレビュー：スプリントの終了時に行うふりかえりミーティング
・プロダクトバックログ：業務視点で優先順位付けされた機能リスト
・スプリントバックログ：スプリントで実現するタスクリスト

スクラムの適用

　スクラムをどうプロジェクトに適用していったのかを説明します。私が5月末にスクラムマスター認定セミナーを受講した時点で、全体の残工数を改めて見積もる必要がありました。そこで、スクラムの見積もり手法の1つである「見積もりポーカー」を試しました。これがきっかけで、スクラムの適用が始まりました。

プロジェクトの概要

●広告代理店の販売管理システム
●15人月ほどの規模
●2月にプロジェクト開始。すべての機能の詳細設計が完了した5月末からスクラムを適用

やったこと

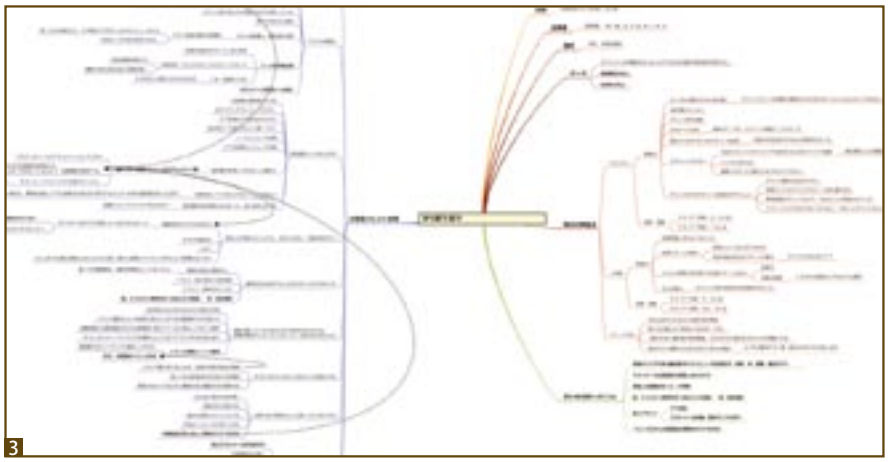
・見積もりポーカー：機能ごとに、基準との相対規模をカードで出し合います（**写真1**）。外部仕様を確認しながら「いっせーのせっ」で見積もりカードを出し合います(**写真2**)。楽しくてとても盛り上がりました。外から見たらほとんど遊んでいるようにしか見えなかったでしょう。
・スプリントバックログ：イナズマ線を書く既存のWBSをそのまま流用しました（**図2**）。スプリントごとにシートを分けて作成し、残工数をタスクごとの日々のマスに記入していきました。日付のところにそのタスクの残工数を記入して現在の残工数を把握します。この値をプロットするとバーンダウンチャートになります。
・タスクボードによるタスク管理：壁にカスクボードを取り付け、タスク管理に活用しました（**写真3**）。
・バーンダウンチャート：タスクごとの自己申告による残作業量合計でバーンダウンチャートを作成しました（**写真4**）。
・タスクカードによる管理：タスクカードにはA6サイズのカードを使いました（**写真5**）。記入事項は「タスクの開始日、終了日」「見積もりと実績」「見積もりの差異が出てきた理由（チームメンバーのK君から出たアイディア）」です。
・スプリントレビュー：ふりかえりを行いました。ニコニコカレンダーやバーンダウンチャートを使って時系列でふりかえります。課題管理表を見直して情報収集し、「555」（トリプル・ニッケルス）という手法で、対策を検討しました（**写真6**）。555は、みんなで意見をカードに書き、回しながら意見を追記していく手法です。声が大きい人の意見だけにならない良い手法です。『アジャイルレトロスペクティブ』から取り入れました。カードを読み上げながらブレインストーミングをし（**写真7**）、マインドマップを描きながら次のスプリントでのアクションを明確にしていきました（**図3**）。

お皿に乗せたプラクティス

　お皿に乗せたプラクティスは次の通りです。

・見積もりポーカー

■写真7　ブレインストーミングの様子
■図3　　マインドマップ



・ニコニコカレンダー
・555

自分で味付けしたプラクティス

自分で味付けしたプラクティスは次の通りです。

・プロダクトバックログ：優先順位は1回決めたら見直しを行いません。
・スプリントバックログ：通常のWBSに残工数の項目を追加しただけのものです。
・タスクボード：タスクカードには、そのタスク完了時に見積もりとの差異の理由や気づいたことを記入します。
・デイリースクラム：朝会と夕会に分けました。「夕会に進捗確認があったほうがモチベーションが上がりそう」というメンバーからの意見から、朝会はお互いの連絡、夕会は進捗というように役割を分けています。
・バーンダウンチャート：1つのタスクは単体テスト完了までとし、日々報告ベース残工数を集計してプロットしました。
・マインドマップを使ったスプリントレビュー

お皿に乗せなかったプラクティス

お皿に乗せなかったプラクティスもあります。

・設計から納品までを1つのスプリントで回すこと：納品可能状態の維持まではやらずに、単体テスト完了までの工程を2つのスプリントに分けました。単体テストの後は結合テスト工程を実施しました。
・ユーザーストーリーごとの優先順位管理：あくまでも画面機能単位の管理としました。
・プロダクトオーナーによる優先順位のコントロール：外部設計が終わった段階からの適用でしたので、プロダクトオーナーは何もコントロールすることがありませんでした。
・タスクの完了単位での進捗管理：実行中のタスクは残りの工数を申告し集計しました。
・柔軟な契約：機能設計段階で一括の見積もりによる契約でした。

選んだプラクティスの味は？

実際に選んだプラクティスを見てみると、ほとんど何らかのカスタマイズを行っています。タスクの管理もスクラムの本来のやり方にはしませんでした。ウォーターフォールの枠組みは崩していないのです。それでもスクラムのエッセンスはプロジェクトメンバーで体験できましたし、製造工程を分割することで、単体テストレベルの品質ではありますがお客様にデモができました。アジャイルの経験がないメンバーに無理なく、それでもその本質を感じてもらうには適切な味付けであったと思います。次は無理なくアジャイル色を濃くすることが可能になるでしょう。

スクラムを適用して良かったこと

今回のように部分的にでもスクラムを適用して良かったことをまとめます。

・自己組織化が進んだ。
・チームメンバーからさまざまな運営上のアイディアが出てくるようになった。
・TDDをやりたいという意見がメンバーから出てきた。
・エンドユーザー様に、早期に動くアプリケーションを見ていただけた。

エンドユーザー様にちゃんと仕事が進んでいるということを実際の成果を見ることで確認していただき、安心感を提供することができました。元請の会社様からも初めての快挙だとおっしゃっていただきました。元来、開発途中のものを見せることは新たな仕様要求を喚起するため、あまり望ましいこととはされていません。元請のリーダーや担当者の方と議論を重ね、しっかり準備をした上でエンドユーザー様のためにやろうという同意を得たことで実現しました。

何よりも、チームメンバーから運営上のアイディアがよく出るようになった点が、部分的にでもスクラムをやって良かった点です。チームの仕事はチーム自身でマネジメントする自己組織化が進んだと言えます。

目の前にいるチームメンバーと何をするのか

私はかねてから銀の弾丸を求める姿勢には懐疑的でした。また、その反対に何も試そうとしない姿勢にも疑問を持っていました。特定の方法論に偏る危険性と、良いものはどんどん取り入れていくべきという思いを同時に感じていたのです。筆者はオブジェクト指向が大好きで、実は何でもかんでもオブジェクト指向でやろうとしたほうな

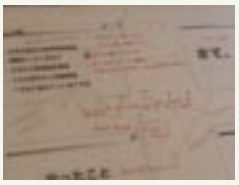
EM ZERO Column コラム

Agile Conference 2008でJeff McKenna氏にアドバイスをいただきました

EM ZERO編集部より、カナダのトロントで行われたAgile Conference 2008会場でスクラム創始者の一人Jeff McKenna氏と話す機会をいただきました。会見は約40分を2回。よく笑うとても楽しいおじさんで、すぐに大好きになりました。Jeff氏には私のスクラムの適用の実績を見てもらい、さまざまな助言をいただきました。

・タイムラインの危険性

Jeff氏「ニコニコカレンダーとバーンダウンチャートの突き合わせはあまり感心しないな。誰の責任で生産性が悪くなったという視点にならないように注意したほうがいいぞ。」



→インタビューの中で、Jeff氏は一貫して、次にどうするかにフォーカスしたほうがいいということを言っています。

・レビューの注意点

Jeff氏「見積もりとの差異をカードに書くのがうまくいっているならいいが、誰が悪いかという視点にならないように気をつけたほうがいいぞ。マイクロマネジメントになっていないか注意したまえ。」

→マイクロマネジメントとは、細かく監視して細かい指示を与える方法はうまくいかないことを揶揄して使う言葉です。

・見積もり、進捗管理手法

Jeff氏「相対見積もりを行って、合計を100%として進捗を管理する手法は良い方法だ。このほかに、プランニングやバーンダウンチャートとのとても有効な活用テクニックがあるぞ。この本（『Agile Estimating and Planning』）がお勧めだ。ぜひ読んでみたまえ。」

→この本は、永和システムマネジメントの安井力さんと角谷信太郎さんが翻訳中で、来年初頭に本屋さんに並ぶ予定とのことです。ぜひ買ってみてください。

・スプリントレビュー

Jeff氏「君のスプリントレビューは目的がはっきりしていないな、スプリントレビューは目的に合わせて、2つのパートに明確に分けなさい。1つは会社の誰でも参加できるもの、もう1つはプロジェクトチームだけが参加できるものだ。」

→スプリントの変わり目のミーティングは、「レビュー」と「ふりかえりとプランニング」の2つの目的にわけることを言っています。私たちはレビューとふりかえりを一緒にやっていたので、他の人がプロジェクトの内容を見る機会がなかったのです。

・プロダクトオーナーが不要の場合

Jeff氏「プロダクトオーナーが機能していないということだが、本当にその役割は必要だったのかな？ユーザーの仕様の安定度が高いときには、必ずしもプロダクトオーナーは必要ないぞ。ただし、本当に安定しているかどうかはちゃんと見極めたほうがいいぞ。」

→ここにも手法をそのまま使うのではなく、状況に合わせて役割を調整してよいという姿勢が伺えます。

・スクラムマスターの姿勢

Jeff氏「スクラムマスターには、技術としてのファシリテーションのスキルは必ずしも必要ないが、メンバーの関係を円滑にしたり、笑顔をもたらしたり、労をねぎらったり、積極的にメンバーの気持ちを聞いたりすることが大事だという君の意見には賛成するよ。何よりも、スクラムマスターはチームを信頼することが重要で、干渉しすぎず、ただ彼らが自分たちで解決することを助ける姿勢が重要だぞ。」

→なんだか泣けてきちゃいました。

のですが、実際のプロジェクトへの適用を進めるにつれ、過度のこだわりにも消極的な姿勢にも違和感を抱くようになっていったのです。

このことは、新しい手法を組織に適用するあらゆる場面に当てはまると思います。新しい手法を試し、検証し、適用する姿勢はとても大切ですが、それを一発で解決する「銀の弾丸」と思うのは望ましくありません。いつでも一番大切なのは、今目の前にいるチームメンバーと何をするのかということだと思うのです。

■参考文献

『スクラム入門ーアジャイルプロジェクトマネジメント』（Ken Schwaber著、株式会社テクノロジックアート訳、日経BPソフトプレス、ISBN-13：978-4891004408）

『アジャイルレトロスペクティブズー強いチームを育てる「ふりかえり」の手引き』（Esther Derby、Diana Larsen著、角征典訳、オーム社、ISBN-13: 978-4274066986）

『Agile Estimating and Planning』（Mike Cohn著、Prentice Hall PTR、ISBN-13: 978-0131479418）

Profile プロフィール

セントラルソフト株式会社
システム開発課課長
林　栄一
HAYASHI Eiichi

電子楽器メーカ→陶磁器の卸会社→鉄道関連の研究開発の会社を経てIT業界に入って12年。オブジェクト指向歴約15年。教育のプランニング、PM、SE、PGなんでも。Agile Conference 2008では日本から楽器を持っていってドラムサークルを開催。目下の興味の中心はドラムサークルを使ったチームビルディング。認定スクラムマスター。認定NLPプラクティショナー。

■謝辞

スクラムマスターセミナーを受講して1つのスプリントを行った結果をもとに、協力会社様対象にプライベートセミナーを行いました。Bas Vodde氏のスクラムマスター認定セミナーで同時通訳をしていたEmerson Mills氏にスクラムのフィロソフィーをレクチャーしていただき、私は適用状況の事例報告を行いました。60名ほどの方々に参加していただき、たくさんのQAやさまざまなフィードバックをいただきました。本稿は協力会社の方々からいただいた貴重なフィードバックを参考にして執筆しました。プライベートセミナーに参加していただいた協力会社の皆様には、改めて感謝の意を表したいと思います。また、Agile Conference 2008への参加を後押ししてくれた古賀社長と木村部長、そしてJeff McKenna氏との対談の機会を用意してくださったEM ZERO編集部に感謝します。

[連続企画] 萩本順三氏インタビュー【第1回】

技術顧問という仕事

コタツモデルで作る元気な会社

聞き手：EM ZERO編集部 豆パン君

萩本順三さんは、2008年4月からリコーソフトウエア株式会社（以下、リコーソフト）の技術顧問として活動されています。萩本さんは、IT企業がもっと価値を出すためには技術者の意識改革と、IT企業としての変革が必要だと日ごろからおっしゃっています。

そこで今回から何回かに分けて、エンジニアとして、どのような意識改革が必要で、IT企業としてはどのような変革が今必要とされているのか、萩本さんにインタビューしていきます。

まずは技術顧問としてどのような活動をされているのか、本誌のイメージキャラクター（?）である豆パン君にインタビューをしてもらいましょう。

技術顧問という仕事とは

豆パン君：そもそもの技術顧問の目的はどのようなことだったアルカ～？

萩本さん：リコーソフトの掲げる新しいソフトウェア企業を目指して、要求開発の普及と、「元気のある会社」というコンセプトをもとにした企業改革と、社員の意識改革、そして楽しめるエンジニアリングを身につけてもらうことです。

まだ半年しか経っていませんが、ようやくその成果が少しずつ出てきました。

豆パン君：技術顧問のきっかけは何だったアルカ～？

萩本さん：それは2007年に開催された、日経BP社主催のXDev（http://itpro.nikkeibp.co.jp/ev/xdev/）を見たリコーソフトの方から依頼が来て同年、リコーソフトが主催する技術イベント「ソリューション フォーラム」で同社高田社長と対談したのがきっかけとなりました。

僕は、以前から要求開発のコンサルティングをやってきましたが、要求開発の本質は企業改革や社員の意識改革にあることを実感し、もっと腰を据えて企業に深く入り込み要求開発の実践ができる企業を探していました。リコーソフトはまさにぴったりな企業に見えたので、担当者の方を通して高田社長にお願いし、実現に至りました。

豆パン君：どうしてリコーソフトがぴったりな企業に見えたアルカ～？

萩本さん：それは高田社長が、お客様の価値をもたらすエンジニアリングをしっかりと身につけさせたい、そのためには要求開発のような考え方がぜひとも必要ということをよくわかっていらしたことです。また、社員こそが企業の財産であることを理解されており、本物の教育を受けさせたいという気持ちが強かったからです。

また、要求開発では、ユーザ企業と開発企業が一緒になってチームを作りビジネス改善・見える化・システム開発につなげていくというのをモットーにしています。このことを、「コタツモデル」というメタファで表現し要求開発では大切にしています。株式会社リコーとリコーソフトの関係は、このコタツモデルを実現しやすい関係にあると思ったのです。

豆パン君：なるほど、コタツモデルというのはコタツに入って、ビジネスのこと、IT化のことをいっしょに考える場なのアルネ～。

「コタツモデル」とは

萩本さん：はい。実際のコタツモデルは「トップ」「業務担当」「IT担当」という役割（ロール）の関係を示しています。僕が要求開発のコンサルティングを実施する際に最も大切にしているのがコタツモデルなんですよ。要求開発方法論（Openthology：http://www.openthology.org/）はあくまで調味料です（笑）。

実は、顧問活動もこのコタツモデルをそれぞれの課題に策定して進めています。

豆パン君：実際には、どのような活動をなさっているアルカ～？

萩本さん：技術顧問としての活動、要求開発のコタツモデル（タスクフォース）は次の4つのサブタスクフォースで構成されています。

1. 教育サブタスクフォース

マネージャークラスに対する要求開発の基礎教育、若手社員のモチベーション向上を図る教育などを萩本が担当し、それ以外のキャリアプランに応じた教育プランについてもサブタスク

EM ZERO Column コラム

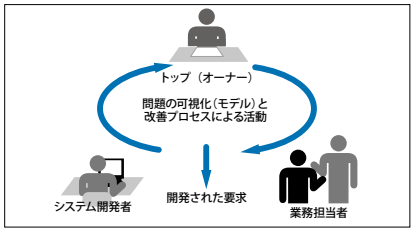
コタツモデルについて

コタツモデルとは、「ビジネスを見える化」し、改革・改善を加えてITに結び付ける道のりの中で、ビジネス要求とシステム要求を獲得していくチームのことを言います。このチームは、活動規模に応じて「タスクフォース、プロジェクト、組織」といった形態で構成されることになります。

コタツモデルは「トップ（事業オーナー）」「業務担当」「IT担当」という役割を持つ人たちが集まる場所を作り、その中で3つの視点から出てくるビジネスとITの要求を獲得し方向性を整理していきながら、その中でビジネス要求とシステム要求を開発するという活動を行っていくことを象徴するメタファなのです。

実際の活動はコアメンバーを作って会のプランを立て、週1〜2回、規模が大きいものは毎日活動するなど様々であり、トップも月1回は必ず参加してもらいます。

要求開発には、このコタツモデルが必須であり、最も大切なものとして考えています。



として確立する。

2. プロセス・ルネッサンス・サブタスクフォース

CMMレベル4を取得しているリコーソフトの開発プロセスに対して、要求開発を導入するための計画および改善活動。

3. モチベーションアップ・サブタスクフォース

社員のモチベーションを高め、楽しめるエンジニアリングを実現するための様々な活動。社員の持つ「私の課題」について技術顧問を交えてデスカッションを行う。また、テクニカルラウンドテーブルと名付けた、納得感のあるソフトウェア開発プロセスを確立するために、開発プロセス、要件定義、ユースケースモデル、アーキテクチャ設計などといった技術について深くデスカッションを行う。

4. 会社を元気にするサブタスクフォース

これはリコーソフトの掲げるドキュメントソリューションというコンサルサービスの充実を図るために、要求開発の考え方を導入し、優れたコンサルタントを創出することを目標に計画を進めている。

それぞれが、コタツモデルとなっていて、活動のプランニングと実施・評価を行っていきます。

新人との活動

豆パン君：様々な活動をおやりになっているアルネ〜。

萩本さん：はい。それだけに僕自身も楽しめています。まさに現場感覚が磨かれるのですよ。今までに要求開発の講演は24回、若手社員のモチベーション向上を図る教育も5回程度完了しました。鹿児島、北海道北見、札幌、秋田などの事業所を回ってセミナーと技術交流も行っています。

また、既に進行中のプロジェクトのアドバイザリーなども行っていますし、新人にビジネスを提案させそれを具現化していくという面白い活動（**写真**）も始まりましたから、いよいよこれからスピードアップできるのではないでしょうか。そのためにコタツモデルの中に高田社長を含めたコアチームを結成。チームメンバーの一人は、要求開発のような会社全体のパワーアップに積極的に関わるマネージャーですので、ほとんど僕と行動を共にしています。

元気のある会社を作りたい

豆パン君：やはり要求開発を実践する思いの強さが、成功につながるのでしょうね。それにしても元気のある会社というコンセプトは面白いアルネ〜。

萩本さん：面白いでしょう。逆を言うと、エンジニアたちが元気をなくしているということです。それを隠さず、

■写真　新人との活動

その問題に立ち向かう企業姿勢が僕は立派だと思います。

実際に、IT企業は3Kと呼ばれたりしていますね。そのベースには、社員であるエンジニアが元気をなくしていることがあるのではないでしょうか？

エンジニアリングはもっとクリエイティブな仕事のはずなのに、管理職の方々や、エンジニアたちがみんなで仕事をつまらなくしている。企業も、規則で縛り付けて、新たなことにチャレンジさせようと思っていない。

僕は、要求開発を通して、エンジニアはクリエイティブであるべきだということを、みなさんと一緒に作りあげています。

そしてクリエイティブなエンジニアになるためには、どのような考え方が必要なのか、知識をどのように蓄積すべきなのか、それぞれの技術をどのように捉えるべきなのか、といった本質について議論しているのです。そうしているうちに、参加メンバーの意見として、クリエイティブなエンジニアは、カッコいいし、イケてるエンジニアだよという意見が出てきて、カッコいいエンジニアの行動パターンを研究しようという動きも出てきています。

豆パン君：それはどんな行動アルカ～？

萩本さん：たとえば、自分のスタイルを持っているとか、お客様だけではなく、チームのメンバーにもサービス精神豊かで、その人がいるだけでパ～ッと周りが明るくなるとか、服装だって気をつけようとか…。

豆パン君：そういえば、萩本さんはいつもにこにこアルネ～（笑）。

萩本さん：そうですね。現場のみなさんとの交流の中から、新たなエンジニアのスタイルを確立し、それを広げることで元気な企業作りを進めていますので、単なる技術向上だけではないのです。

新会社“匠Lab”で より良い社会を

豆パン君：そして萩本さん、新しい会社を作られたアルカ～？

萩本さん：そうアルヨ～。新会社はリコーソフトの技術顧問として体験したことをメソッドとして確立します。それだけ僕としても大きな成果を得ることができました。これからは豆蔵もプロフェッショナルフェローとなりましたし、リコーソフトの技術顧問、そして政府内閣官房IT室のGPMO(ガバメント・プログラム・マネジメント・オフィス)補佐官、このような仕事を通して、クリエイティブで元気なエンジニアを生み出していくことでIT企業を変革することが僕の天職なのかもしれません。

豆パン君：最後に、萩本さんは何を目指しているアルカ～？

萩本さん：それは、先ほどから申していますように、IT企業がユーザ企業の価値を高めるようにビジネスモデルや開発モデルを変革することです。これが新会社“匠Lab”（http://www.takumi-lab.co.jp/）を起業した目的です。その中で、クリエイティブで元気なエンジニアがたくさん育ち、いずれはそのようなエンジニアたちがITを活用した匠の技で、より良い社会を形成してもらいたいという願いがあります。ですから匠Labのロゴは、匠の漢字の中の「斤（まさかりつくり）」を「IT」に変えたものにしました。

豆パン君：おお～、なかなかお洒落アルネ～。萩本さん、とても面白いお話、ありがとうアルネ～。

次回は、実際のリコーソフトでの活動について豆パン君にお話を聞いてもらいます。

Plofile プロフィール

株式会社匠Lab
代表取締役
萩本順三
HAGIMOTO Junzo

最近、IT業界に価値のある変革を起こしたくて会社を作りました。みなさん、これからもよろしくお願いします。

RICOH

リコーソフトウエア 株式会社

http://www.ricoh-soft.co.jp/genki

IT技術者はみんな プレゼンやり放題!

リーダーへの報告、メンバーへの指示、顧客との相談、ユーザーへの説明、みーんなプレゼンプレゼン♪お仕事終わっても、ブログに、カンファ、勉強会に交流会、プレゼンばっかりでイヤになっちゃう★そんなキミにとっておきのアドバイス！ボクらが知らないあの分野から、プレゼンテクを盗んじゃおう！！

意外!それは「ギャル」ッ!

さて、今回お勉強させてもらうのは『小悪魔ageha』（インフォレスト）。われわれ男性技術者には縁のない※ギャル系ファッション雑誌です。この雑誌、パッと開くと誌面のテンションに圧倒されてしまいますが、よくよく読み取ってみるとわれわれ男性技術者が使えるテクニックがたくさんあります。特にページ内で主張したいことを一目で読者にアピールするテクニックが大変強力です。その中からすぐに使えるものをピックアップして、解説していきます。

目立つ!タイトル☆ （図1-1、2）

ページを開いて最初に目を引くのは、なんといってもタイトルです。最初に目が行く上端や右端・左端に配置するのは当然ですが、ページ中で最も大きい文字と目立つ色を適用しています。文字を白抜きにしてページのメインカラーで縁取りを行ったり、逆にページのメインカラーで文字を描いて縁取りに白を使ったりして、思い切り強調しています。ただし、PowerPointやOpenOffice.org Impressなどのプレゼンツールでは、書式設定やデザインの機能上本物そっくりにはできませんので、テキストボックスを2つ重ねるといった工夫で乗り切りましょう。

※雑誌に載っているお姉さんにはお世話になっている人はたくさんいます。夜の街的な意味で。

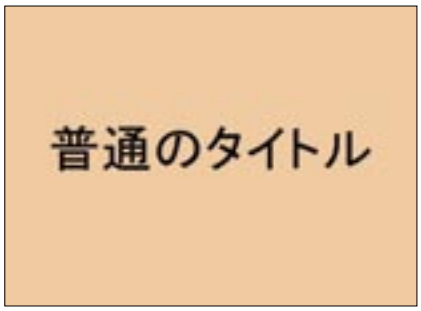
分かる!タイトル・サブタイトル♪ （図2-1、2）

普段は何も考えずに本文より少し大きくする程度の強調で済ませがちですが、本来一番見てもらいたいところです。一番目立つように工夫しましょう。

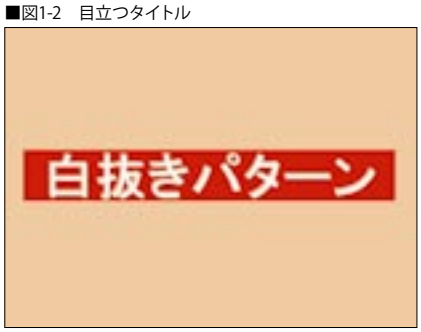
タイトルとサブタイトルでは、ページ中で説明したいことのアピールを行います。例えば、何らかの手法について解説するページの場合、その手法を使うと得られるメリットを含めてタイトルとサブタイトルで表現します。

タイトルやサブタイトルの重要性は分かっていながら、「簡潔にしないといけない」という固定観念にとらわれて

■図1-1　目立たないタイトル



■図1-2　目立つタイトル



■図2-1　簡単すぎるタイトル

■図2-2　メリットを表現したタイトル

しまい、結果的に簡潔過ぎてよくわからないものになりがちです。タイトルやサブタイトルを一見するだけで概要をつかめるように工夫しましょう。

ビジュアル見れば たちまち夢の中♪（図3-1、2）

図表での表現は、プレゼンテーションにおけるベストプラクティスの1つです。特に手順を説明する場合は、各ステップを1つ1つビジュアルを多用して表現します。手順自体を分かりやすくするという効果だけではなく、手順を適用している状況を明確にイメージさせらせます。

プレゼンテーションだけでなくディスカッションを行う場合でも、説明する自分と説明させる相手という「自分vs.相手」の対立ではなく、自分＋相手の「私たちの関係」に持っていくことが重要となります。手順を利用している姿をイメージさせることによって相手を説明の世界の中に引き込み、「私たちの関係」の成立を補強しましょう。

文章1つにテンションぶっ込み☆

口頭でのコミュニケーションではすべてのセンテンスに話し手のテンションがこもります。文章でのコミュニケーションでは、句点の代わりに記号を利用することで書き手のテンションを文字情報によってアピールします。

■図3-1　文字だけのタイトル

■図3-2　ビジュアルの目立つタイトル

・記号があるとき

Excelは、APIとして意外と使いやすいです♪
私のExcel慣れ補正もありますが、非破壊で計算してくれたり、プラグインによって機能拡張ができたりといった機能も持っています★
データ構造としては、ブック、シート、レンジの階層構造に加えて、レンジ自体が行と列による二次元配列的な構造を持っています◎
しかも依存する別のレンジへの参照も持ち合わせているという、なかなか粋な構造となっています♥

・記号がないとき

Excelは、APIとして意外と使いやすいです。
私のExcel慣れ補正もありますが、非破壊で計算してくれたり、プラグインによって機能拡張ができたりといった機能も持っています。
データ構造としては、ブック、シート、レンジの階層構造に加えて、レンジ自体が行と列による二次元配列的な構造を持っています。
しかも依存する別のレンジへの参照も持ち合わせているという、なかなか粋な構造となっています。

記号の有無によって、まったく感じ方が変わってきます。もし記号の利用にどうしても違和感がある場合は、顔文字で代替するのもよいでしょう。その場合、海外式の顔文字を利用すると、シンプルにまとまります:-)。なお、素材がそろっている場合は、携帯電話やインスタントメッセンジャーでおなじみの絵文字を利用するとさらに効果的です。

■参考資料

『小悪魔ageha 10月号』（インフォレスト株式会社、2008年）
『小悪魔エンジニア』（渋川よしき、日本XPユーザグループ主催イベント「XP祭り2008—LT五輪」ライトニングトークス、2008年、http://sky.ap.teacup.com/shibu/57.html）

目指すは俺たちage野郎！

冒頭で述べた通り、IT技術者の仕事はプレゼンテーション漬けという状況です。それにもかかわらず、読み手に対しての表現方法を仕事上で学ぶ機会はほとんどありません。そこでプロの仕業から良い表現の要素を抽出・再構成することで、表現の幅を広げていきましょう。その際には普段接していないジャンルの本を利用すると新鮮な刺激を受けられます。それは今後のエンジニアライフの改善にとどまらず、人としての興味の広がりを得ることにもつながります。その第一歩として、まずはコンビニや書店で『小悪魔ageha』を手に取り、そのクオリティとテイストを明日に活かしていきましょう。

Profile　プロフィール

しば
SHIBA

顔文字そっくり顔の自宅料理員兼ExcelとRubyをこよなく愛するソフトウェア技術者。現場メンバーの1人1人が自律的に活動することを理想として日々活動中。CCPM勉強会「CTR」メンバー。ソロPF&ソロXP&ソロ管理推進担当。東京~~Excel~~Ruby会議01スピーカー。

Scalaの勧め、DSLのゆめ、モノとコトのMixin

株式会社豆蔵
羽生田栄一
HANYUDA Eiichi

1　はじめに

ちりとてちん。皆さん、お初にお目にかかります。世間では、熊さんがオブジェクト指向ラブなんぞと言えば、八さんがくしゃみをしながら、ふぁふぁぁファンクション関数型だ、と喧しい限りですな。ここは喧嘩両成敗ということで、ではひとつ、熊八取らずに終わらないようにわかりやすく、純粋オブジェクト指向の関数型言語「Scala」（**図1**）の特徴をしっかり理解していただこうと考えています。まずはいくつかの事実から。

事実1　ScalaからすべてのJavaクラスを簡単に利用でき、JavaからScalaクラスを呼び出すことも自由です。つまり、膨大なJava、J2EE、Java ME CLDCの資産がすべて、より合理的でコンパクトな形で利用可能になります（現在は提供されないが、過去に.Net上のScalaが存在した。将来復活の予定あり）。

事実2　ScalaはJVM上で実行され、.classファイルが生成され、その実行性能はJavaコードとほぼ同等です。結果として、ほとんどのスクリプト言語より1桁速いと言えます。

事実3　Scalaは純粋なオブジェクト指向言語でありながら、その枠組みの中で本格的な関数型言語を提供します。Scalaのデータはすべてオブジェクトであり、関数もオブジェクトなので、データとして（つまりクロージャとして）操作できます。

事実4　ScalaはJavacやJava Genericsの開発貢献者でもあるMartin Odersky教授率いる強力な開発体制のもとで早いペースでリリースがなされ、ドキュメントも充実し、今後が期待できる実用的な汎用プログラミング言語（スクリプト言語的にも使えますが）です。Scalaは2008年11月の時点でScala2.7.2のRCが最新版です。

■図1　ScalaのWebページ（http://www.scala-lang.org/）

2　まずは対話型インタプリタで

Scalaのインストールは簡単です（特に全プラットフォーム用IzPack Installerが悩まずお勧め）。上記のサイトからダウンロードしたファイルを展開するだけです。後はどこからでも起動できるようPATHをbinに通しておきましょう。コマンドラインから“scala”と打ち込むと対話型インタプリタが立ち上がります（ただし、日本語利用のためには、“scala -Xnojline”として立ち上げてください）。複数行にわたる入力も文法的に判断して受け付けます（**リスト1**）。

3　Javaとの文法上の相違点

ここで簡単にJavaとの文法上の違いをまとめておきましょう。

・型の宣言は、“型名 変数 = 値”ではなくて“変数:型名 = 値”で指定。ただし、与えられた値から型推論できる場合には型名は省略可。
・代入不可の変数はvalで、通常の変数はvarで宣言する。また任意のデータに

EM ZERO Column コラム

Scalaの情報源

【やさしいチュートリアル】「ITPro連載：刺激を求める技術者に捧げるScala講座」
http://itpro.nikkeibp.co.jp/article/COLUMN/20080613/308019/?ST=develop
【本格的なチュートリアル】「DeveloperWorks：多忙なJava開発者のためのScalaガイド」
http://www.ibm.com/developerworks/jp/java/library/j-scala01228.html
【上司の説得用】「InfoQ：なぜScalaなのか？」
http://www.infoq.com/jp/news/2008/01/why-scala
【書籍（Scala 2.7.2対応、日本語訳の予定あり）】“Programming in Scala: A comprehensive step-by-step guide” Martin Odersky, Lex Spoon, and Bill Venners
http://www.artima.com/shop/forsale/

```
scala> val msg = "こんにちは"      <= valで定数の宣言。型省略。リテラルに日本語OK
msg: java.lang.String = こんにちは <= msgがStringだと型推論されている
scala> msg size                   <= msg.sizeやmsg.size()、msg.length()としても同じ
res2: Int = 5                     <= 結果の値はres<N>に束縛され後で利用可。正しく5文字！
scala> (1 to res2) foreach print  <= 1.to(res2)の略記。Intのメソッド toでRangeを生成。
res0: Range.Inclusive = Range(1, 2, 3, 4, 5)
scala> (1 to res2) foreach print  <= (1 to 5).foreach(i => print(i))の略記。無名の関数を渡す
12345                             <= 1〜5を関数printに順番に適用した結果。
scala> msg = "さよなら"           <= valで定義したmsgは定数なので代入不可
<console>:2: error: assignment to non-variable  <= エラー：定数への代入
scala> var msg = "挨拶"           <= varで変数を宣言。型指定するとvar msg: String = "挨拶"
msg: java.lang.String = 挨拶      <= 型指定しなくてもStringと推論。
scala> msg = "やぁ"               <= 変数なので新規に代入可能
msg: java.lang.String = やぁ
scala> msg.foreach(println)       <= msgの各Charに関数printlnを適用
や
ぁ
scala> msg (0)                    <= Stringなどのコレクション系のクラスに対し、ゼロ始まりの(i)で要素iにアクセス
res3: Char = や                   <= msg (i)はmsg.apply(i)の省略形。更新はupdate(i, a)だが文字列には適用不可
scala> for (c <- 'あ' to 'ん')    <= toでRangeを指定。<-は要素∈集合の記号を意味する
     |     print(c)               <= for文の途中であえて改行。インタプリタ中で縦棒で継続可能。
あぃいぅうぇえぉおかがきぎくぐけげこごさざしじすずせぜそぞただちぢっつづてでとどなにぬねのはばぱひびぴふぶぷへべぺほぼぽまみむめもゃやゅゆょよらりるれろゎわゐゑをん
```

■リスト1　Scalaの対話型インタプリタ

defを用いて名前を付けることができる（valの代わりにもdefを使える）。関数やメソッドの宣言にはdefを用いる。

・文の区切り目の";"はオプション。通常は改行で表す。

・一連の複文は";"で区切ってブロック{ }でまとめる。単文はブロックにしなくてよい。先ほどのforループ中のprint(c)は{ print(c) }としてもしなくても問題ない。

・数や文字列、配列も含めすべてのデータはオブジェクト。int、double、boolなども含めすべてのデータは特定の型を表わすScalaクラスに属する。

・voidはUnitクラスとして扱い、Unit型の唯一のインスタンスは()。

・配列のインデックスはarray[i]ではなくarray(i)とアクセスする。配列の参照array(i)および更新array(i)= x もarray.apply(i)およびarray.update(i, x)という通常のメソッド適用と見なされる。

・[ ]は型パラメータの指定に使われ、"type IList = List[Int]"のように別名を宣言できる。型パラメータを使えばほとんど無用だが、型Tへのcastはは asInstanceOf[T]メソッドを使う。

・forループは文法ではなく、syntax sugarとして定義され、map、filterなどに変換される。

・staticという概念はあえて捨てられ、Singletonパターンで代用される。classではなくobjectを"object Singleton extends Any { val data: Int }"などとして定義し、その特性にstaticメンバ代わりにアクセスできる。

・import文でパッケージやクラスをインポートできるが、"*"ではなく"_"を利用する。"import javax.swing.JFrame; import javax.swing.JFrame._"などとする。"import javax.swing.{JFrame=>MyWindow}"などと名前を付け替えられる。

4　Scalaにおける関数の定義

Scalaでの関数の定義を見てみましょう。基本は、名前付きの関数です。defを用いて、関数名、引数リスト、戻り型を宣言し、"=" の右に関数本体を指定します。関数の仮引数の型は絶対に省略できませんが、戻り型は通常省略可です。ただし、関数定義本体が再帰的な場合には戻り型を省略できません。Scalaでは、関数のシグニチャは、「関数名: (引数型リスト)戻り型」として表示されます（**リスト2**）。

次に無名関数の定義（いわゆるラムダ式）を見てみましょう。引数と結果の式を "=>" で直感的につなげばよいのでわかりやすいと思います（**リスト3**）。

今度は同じ無名関数をオブジェクトとして扱い、それにval宣言で名前を与えます。するとその名前に引数リストを渡せば、関数の適用が行えます（**リ**

■リスト2　関数の定義

```
scala> def ##(str: String) = str.size <= 文字数カウント。引数はString。戻り型は省略している。
$hash$hash: (String)Int               <= 戻り型Intがメソッドsizeから型推論された。関数の型の表示。
scala> ##("Scala大好き")              <= ##といった未使用記号も自由に名前に使える。
res4: Int = 8
scala> def fact(n: BigInt): BigInt =  <= 再帰定義なので戻り型を宣言。Scalaの無限長整数BigInt。
     |   if (n == 0) 1 else n * fact(n - 1)
fact: (BigInt)BigInt                  <= 関数factのシグニチャ表示。
scala> fact(100)                      <= 整数リテラル100からの暗黙の型変換でBigIntに。
res5:BigInt=(933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000)
```

```
scala> (n: Int) => n * 2          <= 2倍するという名なし関数λn:Int.(n * 2)を表わす
res6: (Int) => Int = <function>  <= (Int)=>Intという型の関数インスタンス
scala> res6(10)                   <= インタプリタの付けた仮名res6で関数を参照し整数10に適用
res7: Int = 20                    <= 結果は整数20
```

■リスト3　無名関数の定義

```
scala> val double = (n: Int)=> n * 2    <= 無名関数にvalでdoubleという名前を与える
double: (Int) => Int = <function>       <= doubleという名前のデータとその型の表示
scala> double (100)                     <= 関数のバインドされた変数doubleに引数リスト(100)を適用
res4: Int = 200                         <= 結果は整数200
scala> List(1, 2, 3, 4, 5) map double   <= リスト(1, 2, 3, 4, 5)の各要素に関数doubleを適用
res14: List[Int] = List(2, 4, 6, 8, 10) <= 結果は2倍の値のリスト
```

■リスト4　関数の適用

```
object double extends Function1[Int, Int] { //2つの型パラメータは第1引数と戻り型を示す
   def apply(n: Int): Int = n * 2           //引数リストへの関数本体の適用の仕方をメソッド化
}
```

■リスト5　関数doubleの定義

```
class Person(na: String, ag: Int) {
  def name() = na
  def age() = ag
}
scala> val tanaka = new Person("田中", 25)
tanaka: Person = Person@13c2797
```

■リスト6　クラスMixinの定義

```
import javax.swing._
val mameWin = new JFrame("豆窓 EM ZERO個体広告")
mameWin setSize(250, 150)
mameWin getContentPane() add(new JLabel("これからはScalaですから"))
mameWin setVisible(true)
```

■リスト7　Scalaの対話環境

■図2　リスト7の出力

スト4)。

実は関数は、“Function0, Function1, Function2, …”といった引数の数を末尾に持つクラスのインスタンスです。たとえば、先の関数doubleは**リスト5**のように定義されたことになります。

このほかScalaは関数型言語の特性を活かし、関数やクロージャもデータや引数として加工・操作・合成が自由にできます。

5　ScalaにおけるクラスMixinの定義

Scalaのクラス定義で特徴的なのは、クラス宣言が基本コンストラクタの定義も兼ねていることです。そしてメソッドの定義も関数定義のやり方がそのまま使われます（**リスト6**）。

なお、Scalaでは自由にかつ簡単にJavaクラスやJavaインターフェースを利用できます。次の例は、JavaのSwingライブラリの利用例です。Scala対話環境で実験確認しながらSwingを使えるのでとっても便利ですよ（**リスト7、図2**）。

6　Javaインターフェースより強力なTraitによるMixin

Scalaには、単一継承しか認められない通常クラスとは別に、Javaで言うInterfaceをより強力にし属性や操作も持たせられるTraitという概念がありMixin的に多重継承が行えます。Rubyのモジュールに相当するものと言えるでしょう。しかもインスタンス生成時にwith節を使ってMixinすることもできるので非常に柔軟に型やコードの再利用が可能です（**リスト8**）。また、なんとインスタンス生成時に個別にTraitの性質を与えることもできます（**リスト9**）。

7　パターンマッチング機能

次は関数型言語の表現力の源泉であるパターンマッチング機能です。Scalaでは、パターンマッチの対象をcase classとして宣言してやります。こうしておくと、そのクラスのインスタンス生成時のコンストラクタ呼び出し時にnewを省略して、“new Var("x1")”を単に“Var("x1")”とできるようになり便利です。caseクラスを利用してたとえば**リスト10**のような抽象データ型を簡単に定義できます。この型の長い式もコンパクトに例えば、“Fun("x", Fun("y", App(Var("x"), Var("y"))))”と定義できます。

そしてmatch関数で“case パターン => 対処法”という具合にパターンマッチング適用すれば、与えられた構造を簡単にパーズできます。以下は項（Term）を受け取ってλ式風（/¥x./¥y. (x y)など）に印字する関数を定義しています（**リスト11**）。“/¥”で“λ”のつもりです。

パターンマッチでは、条件や構造（値だけでなく型宣言を含むパターン）の一致判断とその構造の一部を取り出して変数にバインドする（コンストラクタ引数まで取り出せます）こととが同時に行われますので、コードがコンパクトになりますし、その値をすぐにその後のプログラムで利用でき非常に便利です。

```
trait Teacher extends Person {
  def teach = ... //バーチャルなメソッドであってもよいし
}
trait PianoPlayer extends Person {
  def playPiano = ... //実装を伴う具象メソッドであってもよい
}
class PianoplayingTeacher extends Person with Teacher with
PianoPlayer <= クラスとして継承
```

■リスト8　TraitによるMixin

```
val tanakaTaro = new Person with
Teacher with PianoPlayer <= 田中太郎に
個別に性質を付与！
```

■リスト9　オブジェクト個別ににTraitの性質を付与

```
abstract class Term
case class Var(name: String) extends Term            <= 抽象クラスのサブクラスとして
case class Fun(arg: String, body: Term) extends Term <= 各Termの具体ケースのクラスを
case class App(f: Term, v: Term) extends Term        <= コンストラクタ兼用で宣言
```

■リスト10　パターンマッチング機能のためのcaseクラス定義(この4行で抽象データ型Termが宣言できる)

```
def print(term: Term): Unit = term match { // パターンマッチ対象termに対しmatchを適用
  case Var(n) => print(n)                  // マッチ時の対応は個別にcase以下で定義
  case Fun(x, b) => print("/¥" + x + "." ); print(b)
  case App(f, v) => print("("); print(f); print(" "); print(v); print(")")
}
scala> val t1: Term = Fun("x", Fun("y", App(Var("x"), Var("y"))))
t1: Term = Fun(x,Fun(y,App(Var(x),Var(y))))
scala> print(t1)
/¥x. /¥y. (x y)
```

■リスト11　パターンマッチング

```
def isIdFun(term: Term): Boolean = term match {
  case Fun(x, Var(y)) if x == y => true // case中のif節で等値テスト
  case _ => false                       // 残りすべてのケース「 _ 」
}
scala> isIdFun( Fun("a", Var("a") ))  <= 項/¥a.aは明らかに恒等関数Id
res33: Boolean = true
scala> isIdentityFun(t1)              <= 項/¥x. /¥y. (x y)は恒等関数ではない
res32: Boolean = false
```

■リスト12　意味解析(各ケース内のif節でガードを付加できる)

さらに次のようにcase中のif節でガードを書くことで簡単な意味解析が実現できます。**リスト12**は、恒等関数（Id）かどうかをテストしています。

8　さいごに

Scalaは言語として見たとき、Smalltalkのように単純で統一性のある文法を目指して定義されています。さまざまな機能は言語として取り込むのではなく、あくまでもライブラリとして提供することで、言語自体は軽くして、ただし性能は意識してかなり最適化がきっちりなされた処理系を提供しています（Groovy ／ Grailsも見た目は違いますし動的な言語ですが、大きな方向性としては類似した思想を感じます。詳しくは当号の山田さん記事を）。

また、Scala上には、いくつかのWebフレームワークが公開されています。Lift（Scala＝階段に対してエレベータという意味）という開発進行中の強力なフルスタックのWebアプリ開発フレームワークが存在します（http://liftweb.net/）。SmalltalkベースのSeasideやDjangoそしてRuby on Railsを意識した作りで、MVC、ORマッパー、Ajax ／ jQuery、AMQPなど機能は充実しています。また日本の西本圭佑氏が「Better CGI, Better PHP!」を目標として開発中の非常にシンプルで使いやすいWeb Flavorというフレームワークが公開されています（http://webflavor.sourceforge.net/）。

Scalaは強い静的型付けがされ、動的な性質が弱い分だけ柔軟性に欠けると予想されがちですが、型推論や高階関数とクロージャ、パターンマッチングなど強力な型理論の背景にも助けられて、表現力においても記述のコンパクトさにおいても十分な実用言語に仕上がっています。ぜひ多くの皆さんに試してみることを強くお勧めします。

Profile　プロフィール

株式会社豆蔵
Coラボ所長、
取締役フェロー
羽生田栄一
HANYUDA Eiichi

オブジェクト指向やソフトウェア工学に関するコンサルティング・教育を自ら実践するとともに、それをテーマとした専門会社「豆蔵」の経営に参画しています。豆蔵Coラボにご期待ください！

Groovyの「動的って動よ」

有限会社メタボリックス
山田正樹
YAMADA Masaki

動的って何？

ソフトウェアの世界ではよく「動的」とか「ダイナミック」という言葉を使います。一見格好良さそうですが、実はよくわからない言葉かもしれませんね。「動的」とはソフトウェアにとって本当のところ、どういう意味を持つのでしょうか。ここではJava VM上で動く、Java互換の動的なプログラミング言語Groovyを例にしながら、ソフトウェアが動的であるとはどういうことかを探っていきたいと思います。

ここで言う「動的」とは、「決定を先延ばしにできるアーキテクチャ」という意味です。考えてみれば、ソフトウェア自身も「動的なハードウェア」と見なすことができるかもしれません。例えば入力A、Bに対してそのAND（論理和）をCに出力するようなハードウェアを考えてみましょう。このような回路を実際に作るには、例えば74XXというようなロジックIC（あるいはトランジスタやダイオード、真空管！のような能動素子）を用意して、プリント基板にハンダ付けすることになります（まぁ、実験ならばブレッドボードに差せばいいのですが）。「しまった、ANDじゃなくてORだった」となったら、ハンダ付けのやり直しです※1。

それに対して「ソフトウェアな人」はこう考えます。回路はめちゃくちゃ汎用的（チューリングマシン）にしておこう。したいことが変わっても回路の作り直しは必要ないように、何がしたいかは後から決められるようにしよう。その「何がしたいか」をプログラムと呼ぶことにしよう、と…。つまりソフトウェアを使うことによって、あるゲートがANDかORかは回路設計の段階ではなく、実行直前にプログラムをロードするときまでに決めればよくなったわけです。「boolean in a, in b, out c; c = a || b」などと書けばいいだけですから。この考え方が後々の世界にどれだけの柔軟性と発展性をもたらしたかは皆さんご存じの通りです。

※1）もっともハードウェアも進化して、今では「動作中に！」回路を組み替えることができるデバイスもあります。ソフトウェアとハードウェアの境界はあいまいになりつつありますね。

動的な型付け

Javaは静的で強い型付けを行う言語です。GroovyもほぼJava互換ですから値はちゃんと型を持っていますし、静的で強い型付けを行えます。interfaceもgenericもあります。Groovyではそれに加えて動的な型付けも可能になっています。「動的な型付け」とは、「変数に型を指定する必要がない」ということですね。例えば、「def destination」とすれば、変数destinationにはどんな型の値でも入れられます（Groovyではintやbooleanなど、いわゆる原始的な型もオブジェクトです）。静的な型と動的な型のどちらを選ぶかはそれぞれに利点があり、きわめて感情的、宗教的な問題です。なので、Groovyの提供する「どっちもあり」ソリューションは（使い方を間違えさえしなければ）現実的で強力ですね。つまり、Groovyでは、「型を決めるのを後回しにできる」だけでなく「どんな型付けを行うかも後回しにでき、かつ、場面に応じて使い分けできる」わけです※2。

静的な型付けの利点については多くの皆さんがよくご存じでしょう。では動的な型付けの利点は何でしょうか。「どんな型の値が入るかをプログラミング時に決める必要がない」ことです。これは従来の「メソッド呼び出し」のパラダイムからすると弱点と言えるかもしれません。相手が持っていないデータにアクセスしたり、メソッドを呼び出したりすると確かにまずいことが起こりそうですよね。しかし、オブジェクト指向本来の「メッセージ送信」を考えれば「とりあえずメッセージを送るから、後はよろしく」ということになります。メッセージに基づいてどのメソッドを呼び出すかは、メッセージを受け取った側の責任ですから問題はありません。というよりも、それがオブジェクトの自律性を高め、全体としての柔軟性を高める仕組みなのです。

そう、実はGroovyではある値の型が決まったからと言って、その値がどんなメッセージを受け付けるかは決めら

れません！これについては後でお話ししましょう。

※2）今のGroovyの仕様・実装では、先ほどの例はJavaで、「Object destination」と書いたのとほぼ同じです。これはもしかしたらこの先変わるかもしれません。例えば、Java7で型推論のメカニズムが導入されればそれが利用されるかもしれません。実際、簡単な型推論を行って編集作業を支援するIDEもあります。

動的なメソッド定義—クロージャ

Javaではクラス・メンバ（クラスの構成要素）としてフィールド（変数ですね）とメソッドがあります。しかし、メソッドはフィールドとは異なり、実行中に内容を変更したり、内容をよそに持ち出したりできません。つまりメソッドの中身はフィールドに入れられるほかの値と平等ではないということです。メソッドはクラスにべったり貼り付いてしまっている、つまり「メソッドの内容はプログラミング時に決まってしまう」わけです。そのせいで、例えばイベント・ハンドラとしてメソッドを1つだけ持った無名クラスを多用するはめになります。これには静的な型の場合と違ってあまり利点がありません。

GroovyはJavaとほぼ互換ですから、普通のメソッドがクラスにべったり貼り付いているのはJavaと同じです。その代わり、Closure(クロージャ)※3というクラスが用意されています。クロージャはメソッドのようなもので、引数を与えて呼び出せます。メソッドと異なるのは、クラスにべったり貼り付いていないのでどこでも定義したり、名前を付ければ自由に持ち歩きできたりという点です。

```
(1..100).grep { it % 2 == 0 }
```

これは1から100の自然数の中から偶数を選び出します。「{ it % 2 == 0 }」の部分がクロージャです。クロージャはメソッドや関数のようなものですが、単なるオブジェクトですから、この例のようにgrepの引数に渡すこともできます（ここではGroovyの慣例に従って、引数を示す( )は省略しています）。

```
(def multiple = { m, n -> n % m == 0 }
```

この例はnがmの倍数であるという関係（述語）を定義しています。ごく自然です。普通のメソッドのように呼び出すこともできます「(foo.multiple(7, 210))」が、次のように書くこともできます。

```
(1..100).grep multiple.curry(7)
```

これは7の倍数を選び出します。つまり「multiple.curry(7)」というのは、multipleの引数mを7に固定した「{ n -> n % 7 == 0 }」というクロージャと同じなんですね。これをcurry化といいます。関数型プログラミングに慣れた人にとってはシンタックス上、curry( )というのが鬱陶しいかもしれません。

```
def multiples = { max, m
-> (1..max).grep multiple.curry(m) }
multiples 100, 7
```

とすれば、multiplesは1からmaxまでの自然数からmの倍数を選び出すクロージャということになります。クロージャは関数型プログラミングの肝です※4。関数型プログラミングには、アルゴリズムを簡潔に正しく書きやすいという大きな特徴があります。また、コードの実行を後回しにしたり、再利用したりできます。関数型プログラミングには無限の深みがありますが、それについてはScalaについての記事もお読みいただければと思います。

※3）昔は日本語では閉包などと呼んでいました。今はクロージャと言うのが普通のようです。

※4）Java7にはクロージャも採り入れられるそうです。いっそのことGroovyをJava7にすればいいのに、と思いますね。もっともGroovyにとってはJavaの巨大な標準化の波に飲み込まれないほうがよいかもしれません:-)。

動的なオブジェクト操作—メタ・オブジェクト・プロトコル（MOP）

Javaにはリフレクション（reflection）という仕掛けあります。これを使うと、あるオブジェクトのフィールドやメソッドなどを取得できます（**リスト1**）。しかし、リフレクションを使うのは大変面倒です。そして限られたことしかできません。

なぜこんな手間を掛けてリフレクションを使うかというと、オブジェクトの操作をプログラム時ではなくプログラム実行時に行いたい場合があるからです。XMLで指定したクラスのインスタンスを、アプリケーション起動時に生成してワイヤリングしておきたい、なんてことがあるでしょう。しかし、手間を掛けた割にはリフレクションでできることは限られています。

Groovyでは、オブジェクトに関する操作をもっと簡単に、もっと自由に行えます。リスト2のようにするだけで、前述のJavaの例と同じように指定した名前のメソッドを呼び出せます。クロージャの力を借りれば次のようなことも簡単です。

```
Foo.metaClass.logging
 = { message -> log.debug(message) }
```

ここでは、実行時にクラスFooにloggingというメソッドを追加しています。カギになるのはmetaClassです。これは ExpandoMetaClassと呼ばれるもので、これを通せばクラスにメソッドやプロパティを追加するなど、さまざまないたずらをすることができてしまいます※5。さらには、invokeMethod( )

という「あるメッセージを受け取ったときにどうするか（どのメソッドを起動するか）」という本来のオブジェクト指向の肝になるメソッドまで書き換えてしまえます。ちょっと自由すぎますね！

このようにプログラムに動的な性質をどんどん持ち込んでくると面白いことが起こります。毎回毎回同じようなことを同じようなプログラムを繰り返し書くのではなく、「あるパターンのプログラムを書く」という、私たちの行為自体をプログラムの中に埋め込むような習慣が付いてくるのです。部品（コンポーネント）の再利用だけではなく、開発プロセスの再利用というわけです。これは究極の効率化、再利用ですね。

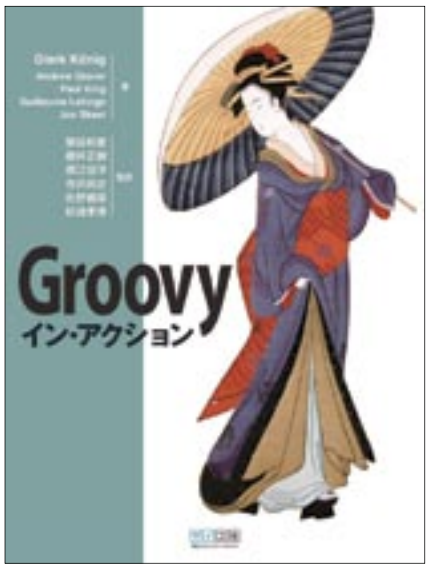
■リスト1　Javaの場合

```
String fname =
 "multiples";
Class c = foo.getClass();
Method m;
try {
   m = c.getMethod(fname,
    Integer.class,
    Integer.class);
} catch (Exception e) {
   // ...
}
try {
   m.invoke(foo, 100, 7);
} catch (Exception e) {
   // ...
}
```

■リスト2　Groovyの場合

```
def fname = "multiples"
foo."$fname"(100, 7)
```

※5）実はクラス・レベルだけではなく、インスタンス・レベルでも同じことができてしまうのです。ということは、あるインスタンスの型を見るだけではどんなフィールドやメソッドを持っているか決まらない!という静的型付け派にとっては悪夢のような状況が実現されています。



おわりに

Groovyについて書きたいことはまだまだいっぱいあります。今回はGroovyの動的な側面を表すいくつかの点についてだけ簡単にご紹介しました。GrailsというWebアプリケーション用のメタ・フレームワークは、Groovyのこれらの特徴を最大限に活用して作られています。そのおかげでGrailsは超高生産性、超柔軟性を実現できているのです。GrailsはGroovyのキラー・アプリケーションですから※6、Grailsのコードを読んでみれば、その面白さに打ちのめされることでしょう。

Groovyは普通のJavaプロジェクトの一部としても何の問題もなく利用できます。Javaのわかりにくいコードをその何分の一かの行数でわかりやすく書くことができるでしょう。ぜひ部分的にでもGroovyをJavaプロジェクトに取り入れて、高生産性、高品質、良アーキテクチャを実現してください。

Groovyについてもっと知りたければhttp://groovy.codehaus.org/をブラウジングするのが一番です。足りない部分はGroovyの標準的なテキストの邦訳である『**Groovyイン・アクション**』(Dierk Konig他著、関谷和愛他監訳、毎日コミュニケーションズ、ISBN-13：978-4839927271)で補うといいでしょう。原著の執筆時期の都合からExpandoMetaClassについて触れられていないのは残念ですが、Groovyの基本はしっかり押さえられています。また、首都圏で月一のペースで開催されているgrails code readingは、その名前の通りGrailsのコードを読む会です。開催はhttp://groups.google.com/group /grails-ja/で告知しています。

ばいばい。

※6）実際、例えばExpandoMetaClassなどは、最初Grailsで実装されたものが後でGroovyに取り込まれた例です。

Profile　プロフィール

有限会社メタボリックス
代表
山田正樹
YAMADA Masaki

SRA、ソニー・コンピュータ・サイエンス研究所を経て、1995年に有限会社メタボリックスを設立。オブジェクト技術、アジャイルプロセスを中心とした研究開発、コンサルティング、プロジェクト管理を行っている。ちなみにメタボリックスの意味は「肥満」ではなく、生物の基本的なアーキテクチャの1つ。最近のテーマは実行可能知識とエンタープライズアジャイル。

アジャイルプロセス協議会のご案内

●珠玉のコミュニティ、人の「和」が広がります！
●組織改革、ビジネスマインド向上によく効きます！
●先進的手法、業界動向、潮流をいち早く察知できます！
●書籍や雑誌には現れない本当の情報や知見が得られます！
●参加企業は皆、いきいきとしています！

Agile Process Association

Concept

■XPやScrum、Leanなど各種アジャイル手法を包括した、より広い開発プロセスをテーマとする（設計論、組織論、経営論も含む）
■小規模から企業や政府等の大規模でミッションクリティカルな分野への展開を図る
■企業や組織での活動に焦点をあて、日本のビジネスに適した取り組みを行う

Activity

協議会では、業界や国内外を問わず「アジャイル」に通ずる考えを持つ著名な方々をお招きしたセミナーの開催、ワーキンググループを主体としたアジャイルプロセスに関する各種手法の確立や知識体系の整備などを通して、常に新しいアジャイル像を追求し、その情報を発信し続けています。

◉現在活動中のワーキンググループ
見積・契約WG、アジャイルマインド勉強会、アジャイル・プロジェクト・マネジメントWG、アジャイル・ソフトウェアセル生産WG、アジャイルTOC WG、西日本アジャイル研究会、アジャイル組込みソフトウェアWG

Information

ソフトウェア業界のハブのような位置付けを目指しております。日本のソフトウェア業界を一緒に元気にしていきませんか？

◉入会資格
・企業（法人、個人企業）、団体、研究機関や教育機関であれば参加いただけます。それ以外の参加条件はありません。
◉会費
・従業員数もしくは団体会員数…30人未満 10,000円／年、300人未満 20,000円／年、300人以上 30,000円／年
・研究機関もしくは教育機関…10,000円／年
◉お申し込み方法
・アジャイルプロセス協議会Webページ（http://www.agileprocess.jp/modules/liaise/）よりお申し込み・お問い合わせいただけます。
・アジャイルプロセス協議会では、どなたでもご参加いただけるメーリングリストを作成しました（http://www.agileprocess.jp/mailman/listinfo/users-ml）。アジャイルの活発な意見交換の場としてご利用ください。

Greeting

株式会社豆蔵 取締役会長
羽生田栄一

今後、ソフトウェアのアジャイル開発は日本でもじわじわと増えていくと考えています。社会がそれを求めているからです。日本においてユーザーにとっても開発者にとっても価値のあるソフトウェア開発が少しでも増えるように、本会では組織的にアジャイル開発のための原理的／実践的／制度的インフラを整備していくお手伝いがしたいと思っています。ご協力いただける皆さんの参加をお待ちしています。

Endorsement

アジャイルアライアンス理事　Ken Schwaber

全世界におけるアジャイルの組織と実践者のコミュニティに、日本のアジャイルプロセス協議会を迎えることを光栄に思います。日本においてアジャイルの活動が行われることは、1986年1月の Harvard Business Review で竹内弘高氏と野中郁次郎氏が発表した「The New New Product Development Game」というセミナー資料がスクラムの起源となった点からも、とてもふさわしいと思います。また、トヨタの大野耐一氏によって実践されたかんばん生産方式はアジャイルプロセスの知識面および実用面の源となっています。アジャイルアライアンスの全メンバーがアジャイルプロセス協議会の発足を歓迎し、喜んでサポートしていきます。

とりあえずメソッド

的確な状況判断と最適なタスクを導くアジャイルなPDCA

福井　厚
FUKUI Atsushi

とりあえずビール

居酒屋で「とりあえずビール」と言うのはなぜでしょうか？それはすぐビールを飲みたいからというのもありますが、ビールを飲みつつ次々と出てくる料理に合う酒を、その場、その場の状況や雰囲気で決めていけるからです。つまり、さまざまな料理の取り合わせや会話、場の雰囲気という状況（コンテキスト）に対して、ビールを飲んだ自分の気持ちや周りの感想というフィードバックを得ることで、その時点で最適な選択を行うことが可能になるのです。それはお店の店員さん（できれば、きれいなおねえさんだとなおよし）との会話の中から、その店の自慢の肴を選択するようなものです。これは何かのフィードバックを得て状況を判断し、最適な答えを導き出すための方法と言えます。

宴会に見るウォーターフォールとアジャイル

最初にすべてを計画してそれを一切変えないとすると、このような最適な選択はできません。例えば、飲み会の企画をするとします。これをウォーターフォールとアジャイルで比較してみましょう。

ウォーターフォール型の宴会

ウォーターフォールの人が幹事をすると、最初にすべての計画を立ててフェーズごとに作業を実施し厳密に各フェーズの完了確認を行う必要があります。

1. X月X日19時30分に居酒屋に到着
2. 全員の着席を待機する
3. 全員の点呼が完了したところで店員をコールし、以下の手順で注文を発注する
   - 3.1. アルコール摂取可能なメンバーの数をカウントし、同数の生ビールを発注する
   - 3.2. 全員の人数から先の発注数を減じた数のウーロン茶を発注する
   - 3.3. 料理は個別に注文すると管理が大変なため、メニューの各分類の中から1つずつ一括して「人数÷4」の数を注文する
   - 3.4. 抜け漏れを防ぐため、上記の方式に例外を認めない
4. 料理が配膳されたら代表者が乾杯を行う
5. その後の注文は予算内で「3」の作業を繰り返す
6. 21時になった時点で幹事から終了の号令をかけ、支払いを済ませた後解散する

さて、これで宴会がうまくいくでしょうか？例えば、時間通りに来ない人がいたり料理がなかなか出てこなかったりすると、ずっと待機することになりますね。最初に立てた計画を変更しないというのは無理があります。もし変更できないとするとさまざまな困難に直面することになります。例えば、

- 乾杯のあいさつが延々と続き、終わりが見えない
- 課長の説教が宴会中続く
- なかなか人が集まらないので延々と開始できない
- 一人でぽつんと座ったまま、終了予定の時間を迎える
- 誰もお金を払ってくれないまま、気がつくと全員がいなくなっている
- いつまでたっても料理が全然出てこない
- 頼んだものとまったく違うお酒が運ばれてくるのでルール上飲めない
- 生ビールに飽きたのに別のものを注文できない

これでは宴会のデスマーチです。もう宴会どころではありません。

アジャイル型の宴会

ではアジャイル型の場合はどうでしょう。

1. 店の情報と集合時間を参加者にメールする
2. その場にいるメンバーで生ビール以外の人の注文を聞き、あとの人数分は「とりあえず生ビールN杯」と店員に言う
3. 状況に応じて好きなものを好きなように頼む

アジャイル型の宴会のほうが素敵です。というか普通です。見方を変えると、ウォーターフォール型は最初に全部考えて、後は考えないということです。一方でアジャイル型は常に考えていることになります。これをチームに置き換えると、ウォーターフォール型は最初に頭の良い人がすべての計画を立て

て、後のメンバーは何も考えずに言われたことを言われたまま実行するだけということになります。最近の若いエンジニアがプログラミングで分からない問題に直面した時に、なぜそうなっているのか（Why）を聞かずに、どうやって作るか（How）だけを聞く傾向があると耳にしますが、これも頭を使わなくなっていることが原因ではないでしょうか。

一方、アジャイル型は常に状況に応じてチーム全員が考え行動するということです。チームの力を最大限に発揮することで直面する問題の解決に当たる姿勢が重要になります。

「とりあえず」という言葉の意味

ここまで何気なく「とりあえず」という言葉を使っていましたが、「とりあえず」とはどういう意味でしょうか。辞書で調べると次のような内容が見つかります。

とりあえず【取り▽敢えず】
（副）〔取るべきものも取らずに、の意から〕
(1)いろいろしなければならないものの中でも第一に。さしあたって。まずはじめに。
「―これだけはしておかなければならない」「―お知らせ申しあげます」
(2)すぐに。直ちに。
「取るものも―かけつける」
三省堂『大辞林第二版』
（http://dictionary.goo.ne.jp/）より

なかなか良い言葉だと思いませんか？いろいろやらなければならないことの中でも第一にすべきことに対して、「とりあえず」という言葉を使います。そしてそれは「直ちに」行われなければなりません。アジャイルの人はよく「とりあえず」という言葉を使うような気がします。

「とりあえずざっくりとプランを立てよう。」
「とりあえずイテレーションの期間を決めよう。」
「とりあえずタスクに分割して。」
「とりあえず優先順位を付けちゃいましょう。」
「とりあえず（機能はゼロだけど）ビルドが自動でできるようにしよう。」
「とりあえずペアプロで。」
「とりあえずテストを先に書いて。」
「とりあえず目に見えるようにしよう。」
「とりあえず動くものを見てもらおう。」
「とりあえずふりかえろう。」

アジャイルでは、まず「第一に」やるべきことを「直ちに」実行して、その結果を目に見える形にします。そして、その結果に対するフィードバックを得ることで、次にまた「第一に」やるべきことをアクションとして決めていきます。これは短い期間でPDCAのサイクルを回していることに他なりません。

アジャイルな人は飲み会が好き？

なぜアジャイルな人は飲み会が好きなのか考えてみました。人が好きというのももちろんありますが、常に変化する状況を受け止め、その中で最適な判断をすることが好きだからなのではないでしょうか。そのような判断力を鍛える場として飲み会を活用しているのではないでしょうか。後から人が増えたり、予定していた人数が集まらなかったりすることは日常茶飯事です。そのような変化に富む状況に自ら好んで身を置くことで変化への対応力を鍛えているのです。つまり飲み会も修行の一つなのです。また、アジャイルな人の特徴として参加者の中に知らない人がいても平気なことがありますが、それも新しい状況に適応するのが好きだからなのでしょう。

やってみましょう、とりあえず

私はこのアジャイル特有の行動を「とりあえずメソッド」と名付けることにしました。最初にすべてを計画するのではなく、臨機応変に対応するための足がかりとして、最初にやるべきことを「とりあえず～」と言ってしまいます。そして、それに対する状況の変化やフィードバックに対応して、次の「とりあえず」を決めていくのです。

「とりあえずメソッド」は、変化に対応するためのとても有効なメソッドだと思いませんか？そう思ったあなた、早速仲間や同僚に「とりあえずメソッド」を教えてあげましょう。明日会社に行ったら、「EM ZEROに面白い記事が載っていてさー」とこの記事を紹介してください。そしてみんなで「とりあえずメソッド」を実践してみましょう。といってもなんのことはないです。今までもやっていたはずのことです。やるべきことは最初に行うべき最も重要なことを考えて「とりあえず～しましょう」と言うだけです。ね、簡単でしょ。

ではやってみましょう、とりあえず。

Profile プロフィール

福井　厚
FUKUI Atsushi

コミュニティとお酒が大好きで、アジャイル界隈では一部から仙人と呼ばれたりするらしい。最近、 MCA（Microsoft Certified Architect）に認定され、ますます怪しさが増している感じ。XPJUGスタッフ、VSUG運営委員、デブサミコンテンツ委員、MS MVP for Solutions Architect。

ソフトウェア哲学の五稜郭モデル ITの世界観を描く手法

株式会社一
大槻　繁
OTSUKI Shigeru

アジャイルプロセス協議会（http://www.agileprocess.jp/）の見積・契約ワーキンググループ恒例の合宿で函館を訪問しました。地元のソフトウェア関連企業のエンジニアの方々との交流、北海道という大らかで雄大な自然が育む気質、港町特有の異国情緒が醸し出す独特の折衷美などから多くの発想上の刺激も得ることができました。合宿で各地を訪問すると、毎回、新しい考え方や手法を思いつき、ご当地モデルとでも呼ぶべき成果をまとめることができるのがご利益です。今回は、題して『五稜郭モデル』、言うなればソフトウェアに関わる哲学的、本質的困難を描くための図式です。どれくらい困難かを説明しちゃいましょう。

五蘊、陰陽道五行説

五稜郭というのは、敷地の形状が五角形をしています。数字の「5」の整理の仕方というのは、どちらかというと珍しい部類ですが、すぐ思いつくのが仏教思想の「五蘊」です。これは存在の構成要素で、世界は、物質的現象、感覚、表象、意志、認識の5つから構成されています。一方、「陰陽道」の「五行説」で、こちらは世界が、「木火土金水」から成り立つと言っています。木が燃えて火になり、燃えた残りで土になり、土の中に資源としての金があり、金が出るところには水が沸き、水が木を育てるといった相生関係と、木が土から養分を採り、土は水をせき止め、水は火を消し、火は金を溶かし、金は木を切るといった相剋関係とがあります。こういった五角形に配置して、その間の連関や対峙の関係を分析できるところが、「5」の優れたところです。

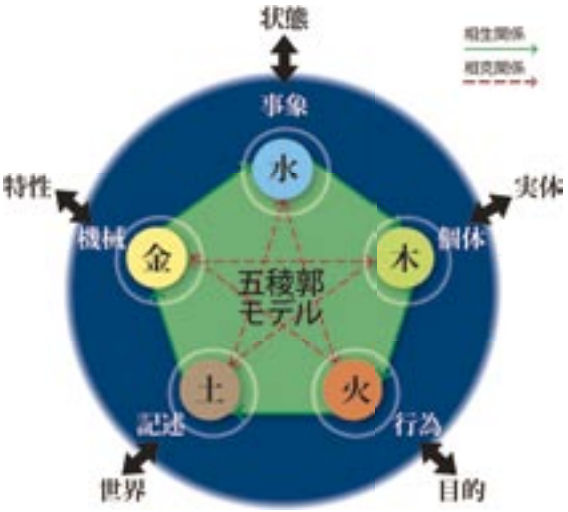
五稜郭で世界を描く

世界観や価値観について考えたり、伝えたりするには「5」というのは縁があるようですね。『五稜郭モデル』は、ソフトウェアやシステムに関わる世界観について記述するモデルです。五角形の周辺に言葉を配置するだけのモデルなのですが、不思議といろいろなものが見え、考えが深まる効果があります。

ソフトウェア哲学の五稜郭モデル

一つ目の例として「ソフトウェア哲学」という世界観を語ってみることにしましょう。簡単に言うと、ソフトウェアの仕様化、モデル化、開発方法論、アーキテクチャなどでいろいろ出てくる本質な困難、世界観の把握について考える際の構成要素みたいなものを整理できないかと思ったわけです。どうしても、開発手法やプロセスの哲学的な背景は、形而上学的な課題や、世界の認識の問題に入り込むことがよくあります。こういった時にちょっとした考え違いをしたりすると、深みにはまったり、場合によっては正しい分析を進めることができないので、考え方を示す原理（principle）のようなものがあると役に立ちます。

図1にソフトウェア哲学の五稜郭モデルと、5つの視点を配置したものを示します。それぞれの視点には、関連した2つの概念が対応しています。この2つの概念を取り違えると大抵の場合は



■図1　ソフトウェア哲学の五稜郭モデル

誤った分析になります。そして、これら5つの観点をバランス良く分析していくことにより、より高い成果や知見を得ることができます。「機械／特性」「事象／状態」「個体／実体」「行為／目的」「記述／世界」は、問題分析や仕様化の場面でよく現れるもので、それぞれ前者（機械、事象、個体、行為、記述）が源認識（より根源的な）であり、後者（世界、実体、目的、状態、特性）は上位認識（より派生的、あるいは、相対的な）です。

個別にそれぞれの5つの項目について、簡単に説明しておきましょう。

●機械／特性

ソフトウェア開発の目的は、抽象的な機械を作ることです。そのためにさまざまな仕掛けや仕組みが必要になります。機械は記号を処理して、計算（実行）するものです。機械の仕様と実現とを明確に語る必要があります。機械について語る局面はいろいろありますが、この中で重要なものが、機械の置かれる適用領域（application domain）での特性（feature）です。函館のワークショップでは、掃除機を例にしてフィーチャを抽出することをやりました。サイクロン搭載とかモータ電力といった実現（実装）でフィーチャを語ってしまう過ちが多いようです。一般に機械には機能の記述もありますが、これは得てして実装領域の概念に引きずられてしまうようなので注意が必要です。

●事象／状態

事象（event）が先か、状態が先かという議論がよくありますが、現象を分析したり、要求を述べるためには、事象のほうが根源的です。JSD（Jackson System Development）では、実世界モデルをダイナミックな表現方法で分析すべきだと主張し、第1ステップで事象を認識するところから始めています。世界の中での変化のほうが認識しやすいからです。しかも、JSDでの事象は、原則として個体間の共通事象（common action）として認識します。事象の順序関係を規定する中で状態概念が派生的に出てくるというのがジャクソンの主張です。ただし、通信（protocol）分野や、既存の知識体系が状態に依存した整理がなされている場合には、状態概念も現象学の主役になることもあり得ます。

このあたりについては、もう少々哲学的な考察が必要ですが、とりあえずは、事象のほうが根源的という世界観は説得力があると思います。

●個体／実体

世界の現象である事実を認識する単位を個体と言います。人、数、色、感情など、ものの見方で個体は決まります。事実に関する命題が真か偽かを、世界の中で認識された個体に対して判断できなくてはなりません。あれとか、これとか区別できるような、まさに根源的な認識の単位です。これに対し、実体というのは、言語によって切り取られた集まりです。データモデルで言えば、実体・関連モデルの実体集合（あるいは型）に相当します。太郎君という個体は、人間という実体（集合）に属しているでしょうし、同時に、従業員であり、男性であり、日本人であり…とまあいっぱい実体はあるでしょう。実体の概念は、

ものの見方（現象学）によって、揺らぐものです。

●行為／目的

経済学でよく前提とされる合理的人間とか合理的意思決定とかは、目的が定まると、最大利得になるように行動することになっていますが、世の中そんなに単純ではありません。わかっちゃいるけどダイエットが進まないとか、禁煙できないとかもあるでしょうし、そもそも目的そのものも状況に応じて変わったりします。従って、行為と目的との間にはすごいギャップが存在します。目的そのものも、明確に語ることができる場合と、そうでない場合とがあります。行為そのものは、個体に関わる事実なので、目的よりは根源的であると言えます。よく目的指向とか、目的を共有すればよいといった教えが横行していますが、これほど不毛なものはありません。

●記述／世界

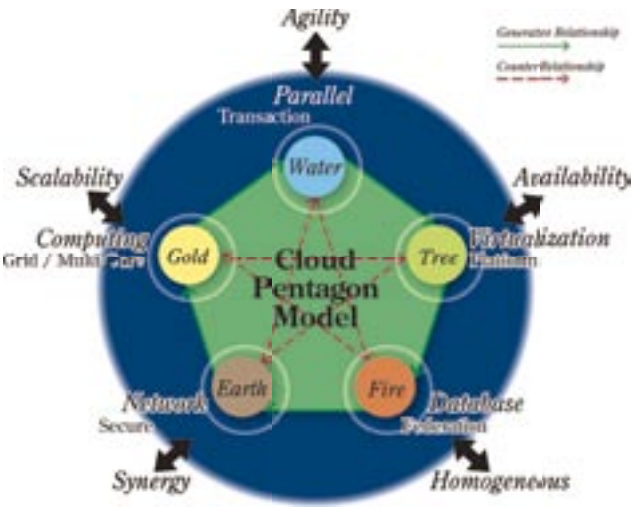
ソフトウェアエンジニアリングは記述の体系です。仕様記述、プログラム記述、各種ドキュメントとかすべて記述です。記述は言語に従っていますが、その意味は、厳格に定義ができる場合と、そうでない場合とがあります。プログラミング言語の場合には、言語処理系そのものが意味を決めている（操作的意味論）とも言えます。数学的に意味を決める方法（表示的意味論）とかもあります。一方、自然言語や、意味が文化やその時々の状況でずれていくものもあります。そういった記述は、それが使われる世界観に依存していますし、逆に、記述の用法が世界観そのものを変えていくこともあります。意味領域そのものが語り得ないこともあります。「トマト5個」という記述が、現実世界のトマトがあるということを示しているのか、お買い物に行ってきてちょうだいという意味なのかは、社会的コード（規範）によって決まるし、そういったルールをいちいち語ることも不可能です。ヴィトゲンシュタインの言葉を借りれば、世界は示しうるのみで語ることはできません。

函館合宿で検討した見積りと契約の観点から、ソフトウェア哲学の五稜郭モデルにどのようなご利益があるかを述べておきましょう。社会的にソフトウェアとしての抽象機械が位置付けられ、それが価値を生むことが期待され、それを構築し維持していくための費用や、複数の組織間での取引や価格設定が行われるという図式の中で、現象認識やさまざまな困難を克服するために五稜郭モデルは活用できると期待されます。五稜郭モデルの外側に位置づけた状態、実体、目的、世界、特性は、高度なバリューチェーンを構成しています。最終的な特性が「金（かね）」に対応しているように、最終的な価値を生むのは利用者から見た特性と位置付けられています。

五稜郭モデルは、いくつかの学問領域の相乗効果を促すマップも提供しています。事象／状態はジャクソンの教えが大きいです。機械／特性はオートマタ理論や計算の理論が基礎付けになるでしょう。個体／実体は論理学（命題論理）の考え方が底流にあります。行為／目的は言語行為論と呼ばれている領域でオースティンやサールといった人々の研究がベースになるでしょう。記述／世界は哲学や世界観の領域でヴィトゲンシュタインの言語ゲーム論が基底にあります。そして、これら全体を結んで具体的な手法に落とし込んでいくところにソフトウェア経済学や価値駆動アプローチが役に立てばよいと思っています。

クラウドコンピューティング五稜郭モデル

さて、五稜郭モデルのもう1つの例を示しましょう。今度は、今流行のクラウドコンピューティングです（**図2**）。このようないまだ定義も曖昧でよくわかっていない領域の世界を描くのにも使えます。欧米主導なので英語で記述することにしましょう。この五稜郭モデルは"Cloud Pentagon Model"と命名しましょう。米国国防総省ペンタゴンとは関係ありません。



■図2　クラウドコンピューティングの五稜郭モデル

木火土金水は、Tree, Fire, Earth, Gold, Waterとなります。Goldがいかにもお金を生み出す感じとか、Earthが大地や地球をイメージできるのも英語にする利点です。さて、5つの項目のうち、内側の源認識にはクラウドの構成技術を適当に掲げてみます。Parallel（並行）、Virtualization（仮想化）、Database（データベース）、Network（ネットワーク）、Computing（計算）といったものがあります。補足的に、グリッドとかマルチコア、トランザクションといったキーワードも配置してみると構成技術の特徴が明確になります。5つの項目の外側の上位認識には、その技術のもたらす効用を掲げます。Agility（アジャイル、あるいは、俊敏性）、Availability（可用性）、Homogeneous（均質性）、Synergy（シナジー、あるいは、相互作用）、Scalability（スケーラビリティ、あるいは、拡張性）といった感じです。

このような勝手な、しかし、綺麗な絵で世界観を描いて、周囲の人々に見せてみましょう。Computingが金を生むという位置付けがしっくりくるとか、NetworkがComputingを地球規模で支えているという相生関係がわかるとか、ComputingがVirtualizationを制している相剋関係がそれらしいねとか、いろいろな意見をもらえたり、結構深いコミュニケーションを図ることができます。先日もこの絵を友人に見せて「クラウドの本質とは？」といった禅問答をしていたのですが、ふと、そうか、

「クラウドコンピューティングとは、サービスのコモディティ化である」

なんて閃きも得ることができました。Scale upからScale outへといった説明もよくなされますが、これは技術者の勝手な言い訳であって、社会的なインパクトという観点から言うと五稜郭モデルの上位認識を集約した上記表現のほうが、私にはしっくりきます。

言葉や関係を1つの五角形という形状の中に配置することによって、いろいろなものが見えてくるから不思議です。ぜひ、皆さんも、もやもやとしている領域があったら、五稜郭モデルを使って自らの世界観を描いてみてください。

Profile　プロフィール

株式会社一 副社長
大槻　繁
OTSUKI Shigeru

日立製作所にてソフトウェアエンジニアリングの研究・開発に従事。2004年よりコンサルタント会社一（いち）副社長。ITシステム関連の調達・開発プロジェクトの見積り評価、診断・改善のコンサルティングを行うかたわら、コストモデルや経済モデルの研究・開発を進めている。
IPA/SEC定量的マネジメント部会委員、同価値指向マネジメントWGリーダ、JEITAソフトウェアエンジニアリング技術分科会委員、アジャイルプロセス協議会運営委員長・副会長を務める。

エンジニアの「マインド」履歴書
自分自身の目標をふりかえる

ゆめぱん
YUMEPAN

ゆめぱんと申します。平々凡々なアプリケーションエンジニアです。ひょんなことから投稿する機会をいただいたので、自分流の履歴書である「マインド」履歴書を紹介しながら自分自身の目標をふりかえる機会にしたいと思います。

よくある職務経歴書

履歴書といえば、この業界では**表1**のような職務経歴書が活躍する場面が多いと思います。私も何度か書きましたし、今でも継続メンテナンス中です。

職務経歴書には自分が関わったプロジェクトやその役割を書きます。対外的に自分の経歴を伝える際に役立ちます。特徴をまとめると次のようになります。

・経験の長さはわかる。
・経験した役割はわかる。
・程度の差はあるが、自分の成功や、（特に）失敗はあまり表現されていない。
・担当した作業ができるようになったのはいつか、実際にどれぐらいできるのかといった作業の品質はわかりにくく、能力もわかりにくい。
・プロジェクトで採用した技術要素が書かれていても、その技術とどう関わったのかはわかりにくい。

プロジェクトメンバーの採用など、よくある使われ方からすると職務経歴書における経歴の「られつ」は最適なのかもしれません。

では、この職歴のほかに次のようなコメントが追加されていたらどうでしょうか。

・□□雑誌で「仕事のやり方」について記事を投稿しています
・△△コミュニティーに参加してチーム開発をより良くする方法を模索しています。あるいは、○○を主宰しています
・仕事は早く終えるのがモットーです。「いかに仕事を簡単にするか」にこだわりがあります
・週2回は同僚と飲みに行きます。とにかくいろんな人を巻き込みます！

直接仕事に役立つかは別問題ですが、ふつうの職務経歴書からは読み取れないその人のエネルギーのかけ方、方向性、もしかしたら人柄や技術レベルさえも伝わることでしょう。

職務経歴書では見えないものを「マインド」履歴書に残す

職務経歴書と、ちょっとしたコメントでは受ける印象が違います。この違いは何なのでしょうか。それは、経歴書では「やったこと」を書くのに対して、コメントには「やりたいこと」が多少なりとも「にじみでている」ことです。「記事を書く」であれば、現状に問題を感じていて「何かを改善したい」という想いがあるのかもしれません。「付き合いが良い」のであれば、人間関係を大事にしているのかもしれません（ただのお酒好きのカモフラージュかどうかは見極めが必要ですね）。

ここで紹介する「マインド」履歴書では、経歴の中で自分が体験したことと、それにより「できるようになったこと」、その時点での「目標」を書いていきます。この方法は、わりと簡単に自分の足取りがつかめるので、効果を感じています。

表2に私のマインド経歴書の抜粋を紹介します。

ちょっとした書き方のコツがあります。過去を振り返りながら、自分が成長したと感じた瞬間を思い起こすのです。具体的に何かできなかったことができるようになった瞬間、またはそれに気づいたときのことを思い出します。そのできるようになったことを「できるようになったこと」の欄に書き、そのときの作業や行動、考えていたこと、出会いを「やったこと・イベント」の欄に書きます。可能であればそのときの目標も書きます。今の自分の仕事において、特に役に立っている技術やノウハウを10個くらいピックアップするとよいかもしれません。逆に、何か作業をしたとしても、得られるものを感じられなかった場合はリストには書きません。

なぜ「マインド」履歴書で目標がわかるの？

「マインド」履歴書の目的は、「本当の自分のやりたいこと」に向かっている軌跡にすることです。もっと簡単に、

「あなたの夢は何ですか？」

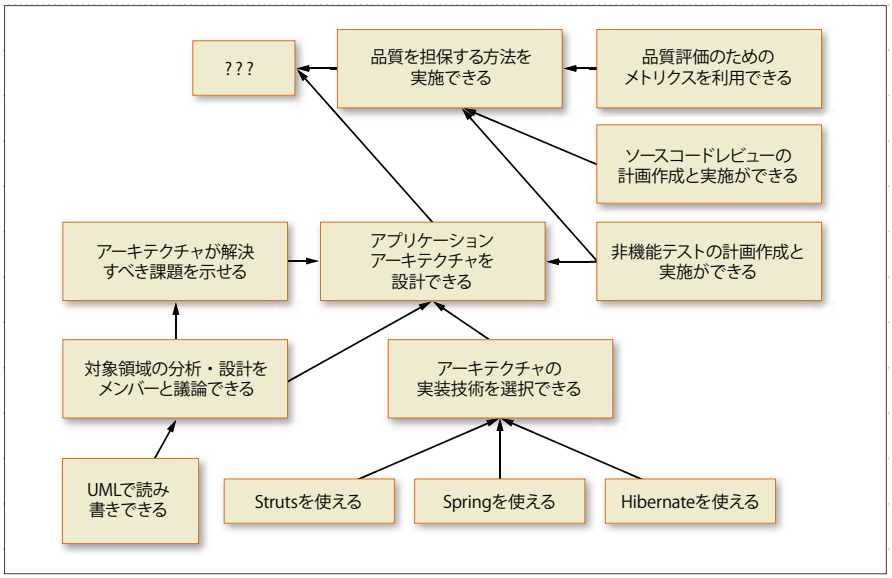
の答えを確信する（場合によっては発見する）助けとなることです。なぜマインド履歴書が自分の目標を見つける手段になりうるのでしょうか。これには逆転の発想が必要です。思い出に残るイベントや、感動した経験を思い出してみてください。人は自分が心から望むものに出会えたとき、心が揺さぶられ衝撃を受けます。そのときを境にして考え方が変わったり、夢や希望に近づいたと感じるものです。こ

■表1　職務経歴書（ゆめぱんの場合）

年月	プロジェクト	概要・規模	システム	作業内容
2004/1	（就職）			
2004/04-2004/09	○○社基幹業務システム	…/15人・2年	C/S（Swing/J2EE）	実装・テスト
2004/10-2005/01	生保外交員向けシステム	…/30人・1年	.Net/J2EE	実装・テスト・環境構築
2005/02-2005/03	△△社流通システム	…/4人	Web（Java）	テスト
2005/04-2005/10	流通・生産管理システム	…/15人・1年	Web・C/S複合	設計・実装・移行
…				

■表2　マインド履歴書の抜粋（ゆめぱんの場合）

年	やったこと・イベント	できるようになったこと
2004	TDD本に出会い、JUnitを自分で実装してみる。EasyMockとJMockを使ってみる。	振る舞いテストを書ける。
2005	DI＋ORMで設計がシンプルになることを知る。	Struts＋Sprint＋Hibernateの組み合わせでWebアプリを作れる。
2005	構成管理が行われていないプロジェクトで、リリースミスが多発。	プロジェクトへSCMを適用し、BTSと連動させられる（環境構築ができる）。Mavenでビルドプロセスを記述できる。
2005	小さなプロジェクトで、成功を収める。アジャイルプロセス協議会に加わる。	コミュニケーションコストが実は非常に大きいことに気づき、それを克服する方法のいくつかを実践できる（現在も模索中）。ドメインモデルで分析・設計をメンバーと議論できる。
2006	非機能テストの計画策定および実施を担当する。	非機能テスト（パフォーマンス・メモリ・不可・構成・セキュリティ）の計画作成と実施ができる。
2006	JNIで大規模レガシーシステムをラップする手法を知る。Findbugsで品質向上を図る。	ソースコードレビューの計画作成と実施ができる。品質評価のために、コードのメトリクスを収集し活用できる（継続学習中）。
2006	アーキテクチャ構築に携わる。	バッチシステムを構築できる。Java/.NetのC/Sシステムの基盤を構築できる。
…		



■図1　現状問題構造ツリーを、目的を発見するために使ってみる

れ、すなわち「成長」です。

「マインド」履歴書に残されるのは、これらの「感動」と「成長」の記録です。あなた自身がピックアップした「感動」は、あなたにとって無関心ではいられなかった事柄であり、本当の夢・希望に触れた瞬間である可能性があるからです。自分で気づいているか、気づいていないかにかかわらず、感動の記録は役に立ちます。

さらに一歩進んで、これから「何をすべきか」を見つけるためには「できるようになったこと」の因果関係を整理するとよいでしょう。ヒントとなったのは「現状問題構造ツリー※」を作成する方法です。詳しい手順は思考プロセスの書籍に譲ります。**図1**に、私のマインド履歴をもとにした因果関係分析の一部を掲載しました。最後には自分が「できるようになりたいこと」を導きます。そのために不足している「能力」も発見します。

※TOC（制約理論）にある思考プロセスツールのひとつ。解決すべき問題がいろいろあって全部解決したいのだけれど、何をすればよいかがわからないときに役立つ、問題の因果関係を表現する図です。現状を把握し、多くの問題の中でも根本的な問題を発見するのに役立ちます。

「マインド」履歴書のパワー

もともと自分がやりたいことを自分が理解するために考えついた「マインド」履歴書ですが、別の側面もあります。プロジェクト新しいメンバーが加わるとき、お互いを理解する上で強力かつ有効なポイントが「マインド」履歴書にはあります。

- 業務上の秘密事項は含まれにくいので、本人さえその気になればメンバーと共有可能。
- 職務経歴書では決して見えない、興味や分析力が一目瞭然。
- メンバーの目的・希望がある程度共有できるので、協力しやすい。
- メンバーにとってのやりがいのある作業がわかりやすい（叶えられるかは別問題）。

小さなチームなどでは、キックオフイベントなどのごく短時間の間に、メンバー各自が履歴書を作り、自己紹介につかってもよいかもしれませんね。

Profile　プロフィール

ゆめぱん
YUMEPAN

本誌某編集者に名付けられた名前です。おそらく本誌に記事を投稿するメンバーの中で、自称最若手（年じゃないですよ、経験年数ですよ！）だと思っておりますが、私よりも経験年数が少ない方がおりましたら、ぜひとも投稿に挑戦してみてくださいな。

50代からはじめたアジャイル

足立久美
ADACHI Hisayoshi

アジャイル・XPとの出会い

私は50代になって初めてアジャイル・XPに出会いました。それは3つの出会いからなるもので、すべてが感動でした。「もっと早くアジャイル・XPと出会っていたら」と思ったこともありましたが、「早い、遅いという問題ではなく、出会ってからどう行動するか（態度）が大切である」と考え、日々精進しています。

幸せの順送り

私は目の前でアジャイル・XPのプラクティスの素晴らしさを実感させてくれる良い先生に出会うことができました。物事を楽しく学ぶこと（楽しさの共感）はモチベーションの向上につながることも学ぶことができました。この出会いを大切にしたいと思います。そこで、良いことを教わったら今度は自分が先生になって次の人に伝えていくことにしました。私のできる範囲でKPT、マインドマップなどを、さまざまな機会を使って広めています。

アジャイル・XPとの3つの出会い

1. KPT（天野勝さん）との出会い

アジャイル・XPとの最初の出会いは2年程前になります。某シンポジウムのクロージング・パネルのパネラーと司会者とで打ち上げをすることになった時のことです。「お疲れさま」の乾杯の後、少ししてからパネラーとして一緒になった天野勝さんが、かばんから手製のボータブルのホワイトボードを取り出し、今回のパネルについての「ふりかえり」を始められたのです。私はこれを目の当たりにして、「このようなやり方があるんだ!」と目から鱗でした。何か1つやり終えた後に、すばやくかつ手軽にふりかえりをできるのがたいへん素晴らしいと思いました。また、KPTとの出会いは、自分がまだまだ“井の中の蛙”であったことにも気付かせてくれました。今では私の担当している部署でKPTの帳票を作るなど、さまざまな活動のふりかえりで活用しています。

2. マインドマップ（串田幸江さん）との出会い

昨年の某委員会のWGにてイベントの講演資料の項目と分担決めをしていた時です。空中戦ばかりで議論が見えなくて困っていた時に、串田幸江さんがプロジェクタを自分のパソコンに接続して、議論の内容をマインドマップで可視化してくれました。このマインドマップとの出会いも、私を「このようなやり方があるんだ!」と目から鱗状態にしました。マインドマップはほんの少しの決まりを覚えるだけで、考えを整理し、深化させてくれる気付き（築き）のツール”であると実感しました。今では講演や議事メモの作成、アンケートの集計やまとめ、特性要因分析（ボーンチャートの代わり）などでマインドマップを活用しています。

3. プロジェクトファシリテーション、XP、ファシリテーショングラフィック（名古屋アジャイル勉強会）との出会い

3つ目の出会いは、名古屋で昨年末に山本博之さんが立ち上げた名古屋アジャイル勉強会です。8月までに4回の勉強会が開催され、私は、初回、2回、4回に参加し、プロジェクトファシリテーションやXP（朝会、計画ゲーム、ペアプロなど）、ファシリテーショングラフィックについて学びました。

- 第1回：プロジェクトファシリテーション（西河誠さん、天野勝さん、山本博之さん、他）
- 第2回：XPとは？（土屋秀光さん、前川直也さん）
- 第4回：ファシリテーショングラフィック（西河誠さん）

特に第2回は4つのXPのプラクティス（朝会、計画ゲーム、ふりかえり、ペアプログラミング）を使って、45分（15分×3回の繰り返し）で「爆笑4コマ漫画」を製作するといった体験プログラムでした。この出会いでは、

- チームワークとは、「厳しい制約条件のなか、メンバー全員が同じ目標を共有し、お互いを尊重し、協力し、助け合って、目標を達成すること」である
- 時間内でのビジネスゴールの達成といった視点も重視している
- 物事を習うのには、楽しいことが大切である

ことを実感することができました。また、楽しく新鮮で、久々に緊張感（スリル）と達成感を味わうことができました。

私のアジャイル・XPに対するイメージ

アジャイル（Agile）を辞書で引くと、「素早い、身軽な、機敏な、生き生きとして活気がある」とあります（大修館書店『ジーニアス英和辞典』より）。アジャイル・XPに接して私が感じた「すぐできる、お手軽、使いやすい、効果がすぐ出る、伝えやすい、教えやすい、やりがいがある、かゆいところに手が届く」といったことにぴったりでした。なんだか、「うまい、はやい、やすい」のあの牛丼屋もアジャイル・XPに見えてきます。また、私はアジャイル（Agile）の意味として「生き生きとして活気がある」が辞書に載っていることをたいへん気に入っています。楽しくて活気がないと、いくら迅速であっても長続きはしないからです。

アジャイル・XPはエコツール

アジャイル・XPのプラクティス、特にKPTやマインドマップは“気付き（築き）”のエコツール”だと思います。人に優しく、人の潜在能力をうまく引き出すことができるので、“知力、体力の省エネ（車で言うと燃費が良いこと）”になるからです。ただ、アジャイルに限った話ではありませんが、「いくら良いツール（手法）があっても、その使い方を誤れば、効果が得られるどころか逆効果になってしまいかねない」ということを忘れてはいけないと思います。

心と形とのバランスが大事

アジャイル・XPは仕組みだけでなく、“心”も重視していて、両者のバランスが取れていることがわかりました（これも適用の仕方によります）。したがって、“形”偏重になりがちなプロセス改善活動を、アジャイル・XPのプラクティスの活用により“心”をうまく吹き込んだものにできるのではと期待しています。

若い人たちへ

人口の逆ピラミッドがさらに進んでおじさんが多くなるので、今後は若い人たちがおじさんたちを使う時代になると思います。若い人たちにとって「いかにうまくおじさんを活用することができるか」が大きな課題になるかもしれません。ただ、おじさんたちにとっても「単に若い人に使われるだけでなく、いかにうまく若い人たちを支援することができるか」といった姿勢を持つことが大切です。そのためには、“心の若さ”を保つことが必要だと思います。

昨年WOWOWで「銀河英雄伝説外伝」（田中芳樹原作、徳間書店）という番組を見ていた時に、次の面白いセリフに出会いました。

「若さと老いの間には歴然たる差があるのだよ! 若さとは何かを手に入れようとすることで、老いとは何かを失うまいとすることだ!それだけで総括できるものではないということは無論だが、この逆ではないということは確かなのだ!」

実にうまく老いと若さの差をとらえていると思いました。

最近、すでに“老い”が出ているような若者を見かけることがあります。体力と同じように、若いころから“心の若さ”をしっかり保っておくことが必要です。そのためには、チャレンジ精神を持ち、アクティブに行動し、多くの人たちと出会い、自分の思い（意見）を発信し、相手の意見にもしっかりと耳を傾ける（傾聴する）ことが大切です。そして、明確な目的を持って主体的に行動し、多くの人と接してお互いに成長できる関係を築き、多くの成功体験を積み自信を付けることが、自分を取り囲んでいる環境を良い方向に変えていくことにつながると思います。

Profile プロフィール

足立久美
ADACHI Hisayoshi

車載組込み制御装置のソフトウェア開発エンジニア。現在は、プロセス改善に関する社内外の活動に参加。“身の丈改善、身の丈チャレンジ”をスローガンに頑張っています。他に、原因分析、安全設計、アジャイル・XPについて勉強中。

Takumi Lab Corporation

www.takumi-lab.co.jp

お客さまにありがとうと喜ばれたり、いろいろなフィードバックが帰ってきます。このように自分の修行を認めてもらい、少しずつ成果を感じながらも終わりのない道を突き進んでいくわけです。道は歩かされるものではなく、自分自身が歩いていくものです。自分で歩くからこそ好きであることも重要になるわけです。好きであるからこそ楽しむことができ、楽しむことにより極めることができるはずなのです。なだらかな道もあればいばらの道もあり。歳を重ねるごとに急な勾配になっていくのかもしれませんが、プログラムでも、設計でも、SEでも、IT以外のどんなお仕事だって、自分自身の道だからこそ楽しむことができ極めることにつながると言えます。例えば、茶道であれば基本の作法と所作を会得して一人前になった時点がスタートだと言われるそうです。基本がわかっているからこそ自分流の味付けや工夫ができるのですね。エンジニアのお仕事はそこまで厳しいものではないのかもしれませんが、ある種の共通点が見えてくるような気がします。

「道楽もの」の本当の姿

少し話を変えて、「道楽」というとお金と時間もかかってある程度の余裕がある人しかできないというイメージがあります。お金持ちのどら息子が家業は親父の言いつけをしっかり守るまじめな弟に任せ、優雅にお金を使いながら毎日をのんびり過ごすなぁんてのも昔の小説やドラマに出てきそうな話ですね。こんな極端な話は例外としても、ある程度「道」を極められた方々だからこそできる「道楽」なんてのもあるわけです。相撲の世界からできた「谷町」（日本語のパトロンさんという意味ですね）であったり、「一見さんお断り」の京都のお茶屋さんなど、ある程度の財力が必要とされるものもあります。しかし、これらの背景には「本物嗜好（しこう）」というものが見え隠れしませんか？

道を極めるためにはいろいろな苦労もいりますし、それだけではなく、広い視野とその視野から必要な本物を見つめる力が必要になります。道を極めるための「技」と、それを実現させる「体」、そして「心」が必要なのです。先ほど述べた茶道と同じように、基本の心技体が揃ってからようやく歩み始める「道」もあります。それらをさらに深めていく方法の1つが「本物をつかむ」ということなのではないでしょうか。例えば、ITエンジニアの世界でも設計・開発の基本や、SEとしての基本ができあがってようやく一人前と呼ばれるようになった後、「この商品やシステムは何を求められているのだろう」といった『価値としての本質』や、「どうやったらこのプロジェクトをうまく動かすことができるのだろう？」といった『人や集団としての本質』などなど、いろいろな意味での本質を探る機会が増えてくると思います。ビジネスとしての本質もあれば技術の本質なんていうものもあるでしょう。そのためにはどうしても「本物を見定める力」が必要になります。そして、そういったスキルを高めていくためには常日ごろから本物を見つめる「目」を鍛えていかねばなりません。

これらは現代の社会でもこれまでの伝統的な社会の中でも変わらないものだと言えませんか？少しの財力は必要かもしれませんが、昔の方が一人前になって初めて体験していた能や歌舞伎、文学や絵画などなど。純粋に楽しもう！という気持ちもあるかもしれませんが、自分自身の目で体感し、それらの中に含まれている「本物」を見極め自分自身の力に変えていこう！という強い思いがあったはずです。芸能を演じる演者、文学や音楽を創り出す作者も、それらを受け取るほうもどちらも真剣勝負であり、そこには「本物と本物のぶつかり合い」があったはずです。見方、考え方、行動の3つのバランスを取りながら基本を押さえ自由に楽しむことができるようになるのです。ご自身の「道」を楽しむことを意識しながら、考え、行動し、その結果をさらに見つめながら本質をつかんでいく。一生終わりのない道かもしれませんが、こんなに素敵なものはないと思います。

「道楽」。今日からこの気持ちを忘れずに毎日毎日のお稽古（けいこ）を嗜（たしな）んでみませんか？そして、この記事を読まれたみなさんは、今からは「自分の仕事を道楽でやってます！」って自信を持って使ってくださいね。

■コラム

みかんちゃんの今日のポイント

- エンジニアとして自分自身の「道」を感じてみましょう
- 毎日のお稽古（けいこ）には必ずフィードバックがあり、それを楽しみながら道を歩いてみましょう
- 基本を身に付けると自分流のアドリブを楽しめるようになる
- 一人前かな？と思ったら本質を探るという道を歩み出すことに
- どんなお仕事でも「目利き力」は必要となる
- 「目利き力」を鍛えるには、日頃から「本物に触れる」ことが重要

Profile プロフィール

Project Facilitation Project (PFP)

前川直也

MAEKAWA Naoya

メーカーのSEPG+社内アジャイル研修講師。箏と三味線の師匠の特徴を活かしつつ、チームビルディングと開発現場の改善に勤しむ。著作に『システム開発現場のファシリテーション』(共著、技術評論社)、「ホスピタリティプロジェクト」(@IT自分戦略研究所)がある。

Project Facilitation Project (PFP)

鈴木遊子

SUZUKI Yuko

マーケティングリサーチ(移動体通信、ヘルスケア)特にフィールドリサーチからデータベースマーケティングのBPOを経て2003年よりPMコンサルティングの企業でマーケティング・セールスに取り組む。「ザ・プロジェクトマネジャーズ」(http://www.promane.jp/)編集事務局。趣味は文章、コラージュなどの創作、歌唱。絵画・写真・歌舞伎を愛する。

【連載】

日本文化とエンジニアの本分

―第1回― エンジニアの道楽

すだち師匠とみかんちゃん

■登場人物

この連載には「すだち師匠」と「みかんちゃん」が登場します。2人をご紹介しておきましょう。

すだち師匠…開発経験12年のすだち君は、こてこての関西人ですが、身に付けたノウハウを活かして現場を改善している、とても熱いエンジニアです。日本文化にも興味を持つ彼は、周りから「すだち師匠」と呼ばれています。

みかんちゃん…入社5年目の「みかんちゃん」は、楽しく仕事をするためにどんどん新しいことにチャレンジする！活発で元気なエンジニアです。「すだち師匠」のヒントを、メンバーと共に実践しています。

「日本文化とエンジニアの本分」復活！

身の周りの日本文化をもう一度見つめ直し、日本で働くエンジニアのみなさんに役立つポイントを探ってみようという「日本文化とエンジニアの本分」がEM ZEROの場をお借りして帰ってきました※。せっかくの復活記念。今回は筆者2人が温めていた企画の中から特に思い入れのあるテーマをとりあげます。久々のすだち師匠とみかんちゃんの和やかな会話に耳を傾けてまいりましょう。

※元々は『エンジニアマインド』(技術評論社)で連載されていました。

み…師匠！師匠！ちょっと聞いてくださいよ～！

す…お！みかんちゃん。どないしたんや。そんな怒った顔して。なんぞあったんか？

み…この前、師匠に演奏会に連れていってもらったじゃないですか。でね、そろそろ自分の時間も作っていけるようになってきたし、私も新しい趣味でもと、いろいろ探していたんですけど、うちの頑固なおじいちゃんが「おい！みかん！まだ社会人になって4年目なのに、趣味なんてまだまだ先のことや。道楽する時間なんてないだろ」なんて怒るんですよぉ。どう思います？

す…ははは。おじいさまも心の底からそうおっしゃっているわけでもないやろ。「たまには渇を入れとかな」っていう、大好きなみかんちゃんへの温かいアドバイスやんか。自分の本業を大切にしているからこそ趣味も充実するんやで、という暖かいメッセージやと思うよ。ところで、その「道楽」っていう言葉やけどな。確かに良い意味で使われることはあまりないよな。「この道楽息子！」とか。でも、純粋に考えると面白い言葉やと思わへんか？

み…というのは？

す…「道楽」ってのは「道を楽しむ」って書くやんかぁ。武道、剣道、柔道、華道、茶道、書道とか「道」って漢字はいろんなところで使われるけど、日本文化にとって「道」という字には、人が何かをする上で修めるべき道理やお作法、筋道といった深い意味も含まれてると思うねん。意味するところは厳しくてかっちりしたもんかもしれんけど、それを楽しむ！って、なんか素敵やと思わへんか？

道を楽しむ

「道楽」という言葉を辞書で調べると、いきなり「本職以外の趣味にふけること。(三省堂『大辞林第二版』より)」なんて出てきて、そりゃそうなのですけど…なんだか否定的だな！仕事を忘れて、他のことに没頭し過ぎるってこと？と、つい悪い印象を持ってしまうのは私だけでしょうか。何か始めようと思ったときに「道楽だね」って言われてしまうと気が引けるよなぁ。どうしよう…なんて、ついつい考えてしまったことはありませんか？

確かに「道楽」という言葉自体がマイナスイメージで取られることも多いですが、すだち師匠の言う「道を楽しむ」という視点で考えてみると、趣味だけではなくみなさまの毎日のお仕事についても何かヒントが見つかりそうな気がしませんか？

能の大家である世阿弥は、『風姿花伝』(現代語訳、水野聡訳、PHPエディターズグループ、ISBN-13：978-4569641171)の中で、

「道を嗜み、芸を重んじて私心を去れば、能の授く恩恵を得られぬことなどあろうか。」

という前フリとして、

「当世この道の輩を見るにつけ芸の嗜みは疎かであり、他芸に現を抜かし、たまたま当芸に当たるや、ただ一夕の悟り一旦の名利に染まり、源を忘れ流れを失うありさま、道すでに廃れる頃かと…」

と書いています。つまり、源流を忘れ、ちょっとしたことで満足してしまい、修行をおろそかにしていると道を外れてしまい恩恵を得ることができないよ！ということですが、日本文化の中で「道」というものはやはり厳しいもので、いつまでも極め続ける終わりのないものなのですね。だから「道」という名前が付いたわけで、「嗜む」ものであっても悠長に「楽しむ」ものではないのだぞ！という古来の戒めの意味もあわせて「道楽」というアンチテーゼなキーワードができたのかもしれませんね。

とはいえ、まずは「楽しん」でみるということも必要だと思います。趣味だけでなく、みなさんの毎日のお仕事だって自分の「道」であると考えてみると、毎日毎日は「道」を進むためのお稽古・練習の場とも言えます。その成果として、周りの方に喜ばれたり、

連載 | メディアコラボレーション【第1回】

@IT自分戦略研究所、InfoQ Japan

**EM ZEROではエンジニアの輪を広げるだけではなく、メディアの輪も広げます。
このコーナーでは魅力的なコンテンツとマインドを持つメディアを紹介していきます。
第1回目は@IT自分戦略研究所とInfoQ Japanのご担当者に登場していただきます。
またオーム社様からEM ZERO読者の皆様にプレゼントを頂戴しております。
振るってお申し込みください。**

@IT自分戦略研究所（http://jibun.atmarkit.co.jp/）

はじめまして。「@IT自分戦略研究所」編集部の岑（みね）です。

EM ZERO読者の皆さんは、「@IT自分戦略研究所」をご存じでしょうか。ITエンジニア御用達のWebサイト「@IT」の姉妹サイトで、「ITエキスパートのための成長支援メディア」と銘打って展開しています。キャリア構築やスキルアップ、ワークライフバランスからコミュニティ活動まで、ITエンジニアの成長を支援する記事やサービスを毎日、更新しています。

日本のITエンジニアを応援するメディアとして、EM ZEROと共に頑張っています。ここでは、皆さんの役に立つであろう記事や連載を3つ紹介します。

おすすめコンテンツ紹介

1. ITエンジニアのための ビジネスコミュニケーション力診断
http://jibun.atmarkit.co.jp/lskill01/index/index_bizcom.html

コミュニケーション力は性格に左右されるものではなくスキルです。診断ツールで、今日からできるコミュニケーション術を習得しましょう。

2. 5分で絶対に分かる職務経歴書
http://jibun.atmarkit.co.jp/lcareer01/special/fivecv/cv00.html

転職活動に必要な職務経歴書。何を書けばいいの？誰に見せるものなの？何枚くらい書けばいいの？誰もが悩む職務経歴書のポイントを5分で解説します。

3. よしおかひろたかの「初めての勉強会」
http://jibun.atmarkit.co.jp/lcom01/index/index_first.html

シリコンバレーの勉強会文化に触れ、自らも1999年から勉強会「カーネル読書会」を運営する「生涯一プログラマ」のよしおかひろたか氏。日本を勉強会大国にするため、勉強会を利用し尽くし、運営するためのノウハウやマインドを紹介します。

あなたのコラム、待ってます！

さらに「@IT自分戦略研究所」では、今年9月から新たに「エンジニアライフ」というコラム投稿コーナーを開設しました（http://el.jibun.atmarkit.co.jp/）。ITエンジニアなら誰でもコラムが執筆できます。EM ZERO同様、原稿料は出ませんが、プロの編集者の文章術指南付きで、皆さんの思いの丈をぶつけることができます。現在、約60人のコラムニストが日々、コラムを公開中。あなたもコラムニストとして参加しませんか？

ITエンジニアのスキルアップに役立つ問題を1日1問無料で配信する「ITトレメ」や、3分でできる「ストレスCheck」などのサービスもご用意しています。ぜひ一度、アクセスしてみてください。

Profile プロフィール

アイティメディア株式会社
「@IT自分戦略研究所」編集担当
岑　康貴
MINE Kouki

インターネットとメディアが好きで、「Webで記事を書いて生活できたら幸せだろうなあ」という思いからアイティメディアへ。2年間の広告営業を経て編集者へジョブチェンジ。編集業務の傍ら、ニュースを書いたり新しいサイトを立ち上げたりしています。最近は自分が何屋さんなのかを見失い気味。

InfoQ Japan（http://www.infoq.com/jp/）

InfoQは、中上級エンジニア向けにエンタープライズソフトウェア開発における最新動向を発信しているサイトで、現在、英語、中国語、日本語、ポルトガル語の各言語圏の国際コミュニティで構成されています。InfoQ JapanはこのInfoQの日本語サイトで、2007年10月よりサービスを開始しています。

入門記事の氾濫した技術情報サイトが多い中、InfoQはJava、.Net、Ruby、SOA、Agile、Architectureの6つのカテゴリで、それぞれの専門分野のエキスパートが執筆した日々のニュースやコラム、ビデオコンテンツ、電子書籍を提供しており、全世界の最新の高度な技術情報を、言語の障害を受けることなく、リアルタイムに肌で感じることが出来ます。

おすすめコンテンツ紹介

1. トヨタの新車開発—Agile2008より

http://www.infoq.com/jp/presentations/Toyota-Kenji-Hiranabe-ja

カナダのトロントで行われたAgile系のカンファレンスであるAgile Conference 2008で、平鍋健児氏がトヨタの新車開発プロセスについて語りました。この動画を日本語の字幕付きで掲載しています。

2. JavaScriptへのマルチスレッド・プログラミングの導入

http://www.infoq.com/jp/articles/js_multithread

IPA未踏ソフトウェア創造事業に採択された、JavaScriptの処理をシングルスレッドからマルチスレッド化する技術の紹介を行っています。

3. 塹壕よりScrumとXP

http://www.infoq.com/jp/minibooks/scrum-xp-from-the-trenches

Henrik Kniberg氏による書籍、『Scrum and XP from the Trenches』を日本語訳し、電子書籍として掲載しています。この本では、とあるスウェーデンの会社が、40人ほどのチームでScrumとXPを実行し、その後約1年にわたってプロセスを改善していった方法を記述しています。

今後の展開

InfoQ Japanは皆様の知的好奇心をくすぐるようなギークなコンテンツを配信し続けます。特に日本発の情報をより充実させ、それらのコンテンツをInfoQコミュニティを通じ、各言語圏へ配信していきます。ガラパゴス化した日本のIT産業の国際化プラットフォームとして、日本から世界への架け橋になれればよいと考えています。

Profile プロフィール

InfoQ Japan
田島一輝
TAJIMA Kazuteru

メーカー系SIerに就職後、2005年5月に関連会社であるコンポーネントスクエアへ出向。2006年よりInfoQ Japanの立ち上げに従事。現在はコンポーネントスクエア マーケティング担当兼InfoQ Japanチーフエディタとして InfoQ Japanの運営を行っている。

オーム社様より読者プレゼント

オーム社（http://www.ohmsha.co.jp/）様よりEM ZERO読者の皆様にプレゼントをいただきました。硬派なエンジニアもうなる“Pragmatic Bookshelf”シリーズの3冊です。ぜひこの機会にお申し込みください。なお、数に限りがありますので、品切れの際はご容赦ください。

『アジャイルプラクティス
達人プログラマに学ぶ現場開発者の習慣』
Venkat Subramaniam and Andy Hunt著、
角谷信太郎・木下史彦監訳、
ISBN-13：978-4-274-06694-8

『Manage It! 現場開発者のための
達人式プロジェクトマネジメント』
Johanna Rothman著、でびあんぐる監訳
ISBN-13：978-4-274-06729-7

『アジャイルレトロスペクティブズ
強いチームを育てる「ふりかえり」の手引き』
Esther Derby and Diana Larsen著、角征典訳、
ISBN-13：978-4-274-06698-6

連載

プロジェクトファシリテーションをはじめよう!

Project Facilitation Project（PFP）
西河　誠
NISHIKAWA Makoto

第3回　「気づき」をつかまえる

Project Facilitation Project（PFP） の西河です。今回はプロジェクトファシリテーションの実践手法の中から、「気づきをつかまえる」ためのテクニックやツールについてまとめてみたいと思います。「気づき」は開発現場のあらゆる場面に潜んでいます。たくさんの気づきが出てくるように促して、習慣として定着させることができれば、プロジェクトの生産性は向上し、チームはどんどん成長していきます。気づきをつかまえる技術をみていきましょう。

「気づき」は突然に

システム設計やコーディングをしていると、突然とても効率の良い処理手順を思いつくことがあります。ドキュメントを作っているときでも、わかりやすい表の作り方がわかることがあります。レビューをしながら、ほかの人の良いやり方を学ぶことも多いですね。

「ああ、そうなんだ！」

エンジニアの仕事をしていると、よくこのような場面に遭遇します。今まで知らなかったことが見えてきたりすると、なんとも思ってなかったことが面白く思えたり、仕事の効率が上がったりします。エンジニアはこうした気づきを蓄積していくことによってスキルを磨いていきます。開発プロジェクトにおいては、お互いに気づきを発展させて習慣に組み込むことでチーム全体の開発効率が上がっていきます。プロジェクトは1つの目標を成し遂げるのが目的ですが、その活動を通して人は多くのことを学びます。「気づき」は個人やチームの成長の原動力なのです。

「気づき」を誰かに話していますか?

「気づき」は個人のスキルを伸ばすきっかけになります。もし、その気づきがほかの人にとっても有用なものだったとしたら、チームのメンバーと共有した方が気づきの価値は上がります。気づきがほかのメンバーによってさらに改良され、より強力なものに発展する可能性もあります。

場合によっては、プロジェクトの問題点や課題が気づきになることがあります。「このモジュールの実装は効率が悪いなぁ」とか、「ここの実装部分は、担当が不明確だから放置されているよ」とか、メンバーそれぞれが個人的に抱え込んでいることがあります。これらの課題も、1人で悩んでいるだけでは解決が難しくても、ほかの人なら改善のアイディアを見つけられるかもしれません。気づきを共有することによって個人とチームはさらに成長していきます。

「ふりかえり」による気づき駆動開発!

気づきをメンバーから引き出して共有する場をプロジェクトに提供しましょう。「ふりかえり」は、名前の通り開発をふりかえって考えるミーティングのことです。P-D-C-A（Plan-Do-Check-Action）のCとAに当たります。

ふりかえりの目的は、次の4点です。

- **これまでの行動を思い返し、そこから新たな気づきを得ること**
- **やってみてうまくいった行動をチームに定着させること**
- **行動が可能な改善策を探し、試す勇気を得ること**
- **参加メンバーの多様性を受け入れ、信頼関係を築くこと**

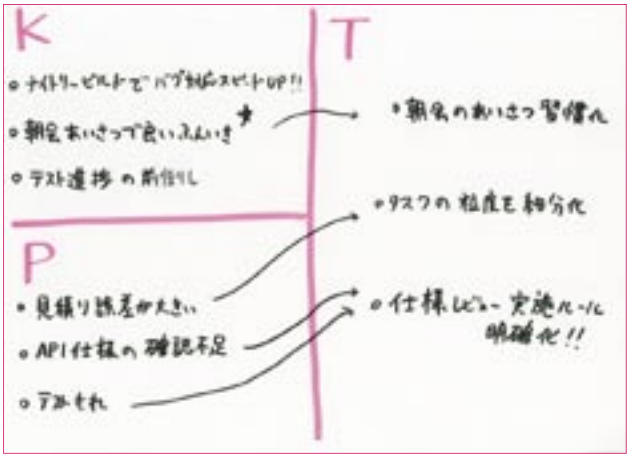
ふりかえりの周期も重要です。週ごとの定例会議とか、全体ビルドのたびにとか、イテレーションごとに実施するのがポイントです。すべての開発日程が終わってしまってからやる「反省会」では、メンバーの気づきを活かしきれません。ふりかえりを繰り返すことで、気づきを拾い、すぐに行動に移していきます。

「KPT」によるふりかえり

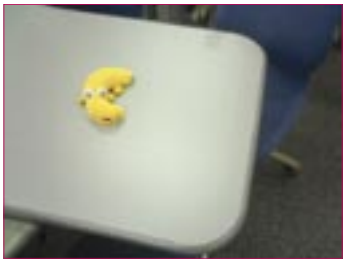
ふりかえりを効果的に行うフォーマットとして、「KPT（Keep、Problem、Try：けぷと）」をご紹介します。やり方は簡単で、定例会議などでホワイトボードの前に集まります。次に、線を引いてください。

1. **ホワイトボードの真ん中に、左右2分割する縦線を引く。**
2. **向かって左側の領域を上下2分割する横線を引く。**

これでホワイトボードが3つの領域に分かれます。分割したホワイトボードの左上の領域をKeep、左下をProblem、右側をTryとします。それぞれの領域に



■図1　KPTの例


■図2　某プロジェクトのバナナワニ園

は次のことを記入していきます。

・Keep：良かったこと
・Problem：問題点、課題
・Try：次にしたいこと、改善したいこと

TryはKeep（それいい！続けてみよう！）や、Problem（これを改善しなければ！）から導き出されることが多いと思います。

全員でKPTを書いていくことで気づきを共有できます。また、定例のミーティングであれば、これを記録しておくことで前回のKeepが継続されているか、Problemが改善されているかを確認でき、カイゼンが習慣になります。KPTは全員で意見を挙げて進行役（ファシリテーター）が書き留めていくのですが、意見が出にくい場合は「1人1つは出すように！」というルールを決めてもよいかもしれません。継続することでお互いの気づきを活用できる雰囲気が作られていくはずです。

『名前付け』で気づきをつかまえろ!

KPTなど、ふりかえりの場を通して気づきを共有できるというイメージはつかんでいただけたと思います。これをもう少し工夫すると、さらにその効果を倍増させることができます。

効果的なのが『名前付け』です。無意識のうちに行っている組織も多いと思うのですが、チームの優れた行動やり方に、愛着の湧きそうなインパクトのある名前を付けることです。そうすることで良い行動が定着し、うまくいけば別の組織にまで広まることもあります（※）。例をいくつかご紹介しましょう。

・山田テスト：山田さんが考えたテスト手順を「山田テスト」と名付けた。手順や目的・効果を毎回話さなくても一言でやることが共有できる。
・バナナワニ園：ある打ち合わせスペースに「熱川バナナワニ園」のキャラクターぬいぐるみが置いてあるだけ。「○○会議室で会議」と言うよりも、「バナナワニ園集合！」のほうが気楽な状態で話し合いを始められます。ローカルな名前だと定着しやすいですね。

※）「かんばん、あんどん」はトヨタ生産方式の代表的ツールです。名前を付けたことで世界的に通用するプロセスとして定着しました。

「気づき」を活かすチームづくり

個人、組織を成長させる原動力である「気づき」は、メンバーから無限に創出される大切な資源です。メンバーそれぞれがお互いに刺激し合い、たくさんの気づきを得て、それを最大限に活かす。KPTや名前付けは、そんなすばらしいチームを作る上での第一歩になります。まずはこれから始まる会議や連絡会でKPTをやってみる。そんなところから始めてみてはいかがでしょうか。

■参考文献
・プロジェクトファシリテーションTOP—オブジェクト倶楽部
http://www.objectclub.jp/community/pf/
プロジェクトファシリテーションの公式資料はここにアップされます。今回のテーマについて、深く知りたいという方には「ふりかえりガイド」がとても参考になります（http://www.objectclub.jp/download/files/pf /RetrospectiveMeetingGuide.pdf）。
・Project Facilitation Project（PFP）
http://ProjectFacilitationProject.go2.jp/Wiki/
プロジェクトファシリテーションの研究・推進を進めるコミュニティ“PFP”のWikiページです。過去のワークショップ資料がアップされています。いろいろなプロジェクトでのPF導入事例も掲載されています。

Profile　プロフィール

Project Facilitation Project（PFP）
西河　誠
NISHIKAWA Makoto

組込みソフトウェアエンジニアとして、各種マイコン制御ソフトウェア、オペレーティングシステムの設計・開発を担当する。
ブログ：http://d.hatena.ne.jp/mnishikawa/

第3回

身近なところから広がるコミュニティ活動

宮田　哲
MIYATA Satoru

ハンディキャップのある方へのサポート

私にとって、ハンディキャップを持つITエンジニアのためのボランティアグループ「智恵の和（http://red.japanlink.ne.jp/chie/）」への参加は、ある意味必然なことだったと思います。これまで私は、障害者スポーツのひとつである「障害者フライングディスク大会（http://homepage1.nifty.com/jffd/）」で、運営ボランティアを春・夏の年2回、スケジュールが合う限りやってきました。この活動を続けるうちに、「自分の仕事であるITに関係することで、ハンディキャップのある方をサポートできないものか」と思うようになっていました。ただし、私の仕事はIT業界といっても組込みソフトの分野なので、参加してもどこまでサポートができるのか不安もありました。

しかし、その不安は何回か参加者のみなさんとご一緒することで解消されていきました。サポートは教えるという役割だけでなく、ハンディキャップのある方の代わりに手を動かす役割もあったわけです。今では月に1度の智恵の和で、みなさんと一緒に勉強するのが楽しみになっています。

根っからの地元っ子

智恵の和に興味を持つきっかけとなった障害者フライングディスク大会ですが、この大会への関わりはとても身近なところからでした。私の子どもがお世話になっている小学校のミニバスケットボールクラブでは、スポーツだけではなく、社会参加活動にも力を入れています。そのひとつが夏の障害者フライングディスク大会でした。私は主に表彰係を担当しています。また、障害者スポーツ大会の表彰式の特徴である万歳三唱を声が枯れるまで行っています。

この小学校は私の出身校でもあり、クラブの指導をされているコーチの中にも卒業生の方が何名かいらっしゃいます。この小学校では、コミュニティとして同窓会がしっかりと組織されていて、この先輩方は同窓会幹事もされています。その縁もあり、私も数年前から幹事としてお手伝いをするようになりました。主な活動は毎年新卒業生を歓迎する同窓会を開催することですが、このうち2年に1回は全卒業生に呼びかける同窓会です。ここ数年お世話になった校長先生のお話によると、公立の小学校でこういった同窓会が継続して開かれているのはあまり聞いたことがないそうです。諸先輩方が築き上げてきたものですが、幹事の末席に名を連ねるようになったことを誇りに感じています。

また、同窓会は卒業生が対象のコミュニティですが、私は「お父さんの会」という、まさに今小学校に通っている子どものお父さん向けコミュニティ活動にも参加しています。こちらは妻がPTAの委員をやっていたときに半ば義理で参加したものでした。今では、この会の趣旨である『子どもと過ごす時間を共有すること』を実現するために、夏は校庭全体で消防ポンプまで使ってしまう「水遊びの会（**写真1**）」、秋には都会ではあまり見なくなったたき火をして「焼き芋の会」を開くなどして、子どもたちのガキ大将になって遊んでいます。

■写真1　水遊びの会

今では仕事の一部？

私が参加しているコミュニティ活動で忘れてはならないものがあります。本業である組込みソフト関連のイベントである「ETソフトウェアデザインロボットコンテスト（**図1**、愛称ETロボコン）」の実行委員会です。2003年、当時はUMLロボコンと呼ばれていたこの大会に、職場の若いモンと一緒に参加し、思いもよらぬ好成績を残したことが始まりです※。翌年以降もチームメンバーやオブザーバーとして、毎年何らかの形で大会に顔を出してきました。昨年からは地区大会も開かれるようになり、いつの間にかスタッフとしてその運営のお手伝いをし、今年からは全体の技術委員として競技ルールを決めるなど、参加者のみなさんが楽しんで大会に参加できるように奔走しています。実行委員として、これまで行ったことのないところへも行かせてもらっています。そのため、休日返上となってしまうこともしばしばですが、その土地のおいしい物を食べられるのも楽しみだったりします。

■図1　ETソフトウェアデザインロボットコンテスト（http://www.etrobo.jp/）

※）この時の参戦記が、http://with.esm.co.jp/workshop/2004/uml-robocon2004/sansennki/TDI/に、まだ残っているようです(^^;。

これから

今回ご紹介したコミュニティは、スタッフとして参加しているものでしたが、これら以外にもメンバーとして参加しているコミュニティもあります。どのコミュニティでも、いろいろな人と話をし、たくさんの刺激を受け、また、たくさんの気付きを得られます。これは普段の仕事をしているだけでは得られないことを経験できるからだと思います。その根底にあるのは、何より自分が楽しんで参加しているということです。そういった意味で、今の私にとってコミュニティ活動は欠かせないものになっています。

Profile　プロフィール

宮田　哲
MIYATA Satoru

根っからの組込みソフトウェアエンジニア、略して「くみこまー」。最近は現場から離れて、要員スキルアップのカリキュラムをしているらしい。

社内SNSがオープンソースになりました!!

SKIPは、社内コミュニケーションを活性化させ、
企業内に散在する“暗黙知”を“見える化”して繋げていく、
ナレッジマネジメントを実現するSNSです。

Enterprise2.0を
実現するための
プラットフォーム

暗黙知と
KnowWhoを
重視したナレッジ
マネジメント

さまざまな
ソーシャル
アプリケーションを
統合

エンタープライズ
向けに特化した
SNS機能

以下の全ての機能が
フリーで使えます。

- マイページ
- ブログ
- ブックマーク
- プロフィール
- グループ
- アンテナ
- 全文検索

☛ダウンロードは

⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇⬇

http://www.openskip.org/

http://www.openskip.org/
お問い合わせ　info@openskip.org

豆パンくんの！ナマスヴィレッジ


©Freedom Studio
2009年もヨロシク!!
お願いしますアルー!!
がしっ
ボク がんばるアルネ〜!!
でぇでがくね?
どこぱん
のぐ
うぅん
どうすぉー
下のテストであなたのパンダ度がわかるアルヨ!
スタート!!
3度のごはんよりどこぱんが
♥ネギ♥
密の業
Yes
NO
やってみてぇ〜!
Yes
NO!
EMZERO
を読んでいるとナゼか無性に上野動物園に行きたくなる衝動に駆られる〜!
お・おさえられん
ヤバイョ
ぐっ
ヤマザキのジャンボ肉マンより
「豚まん」のひびきにぐっときちゃう
ほっかほか
サムゲターンよ
と聞くと、血がさわいで落ちつかない
ぐっ
自分のニックネームの語尾に「パン」をつけると異常にモチベーションが上がってしまう
どこぱんとか…
のぐパンとか…
タヌという言葉よりフリ〜という表現に惹かれる
カラフルな色より白黒の方が萌える!
も、もえ〜
もふ×2
した物をさわるとパンダの赤ちゃんを想像してしまう
最近「トラクター」とか「農業」とか聞くとウキウキわくわくしてしまう〜♥
ピーカンな青空の下より、山奥の竹林の中がスキ
『くまのプーさん』のEDと『パンダコパンダ』のEDが並んでいたら迷わずパンダコパンダをえらぶ!
パンダ度
うーんもうちょいっ!
70%
がんばって上をめざそう!
パンダ度
かんぺきっっ
120%
ゆーことなっしんぐでる
パンダ度
ハーフやね
50%
バランスいいかも♥
パンダ度…
かなり要素あるクロ…
人間
ですネww
そ、そーですか…
さぁ〜あなたは何%でしたか?ww
2009年もヨロシクお願い致します。
くナマスヴィレッジ

連載

私のテキトー読書術

第3回「読書の記録」

こんにちは、あまのりょーです。読書周りの私の個人的な事情をさらしてみるこの連載。今回は「記録」について書いてみたいと思います。

まず、「なんで記録するのか」という問いに対する最も大きな理由は「将来の自分のため」です。昔読んだ本をまた買ってしまう、ということを何度となく繰り返したことが発端です。冗談みたいな話ですが、書店で立ち読みして確認してから購入しているのにもかかわらず、3分の2ほど読み進んだところで、「あ、これ昔読んだ」と気づくことが頻発したのです（特に宮部みゆきさんや東野圭吾さんなどの多作な作家の場合）。

オンラインで記録しておき、ケータイからもチェックできるようにすることでこのトラブルをかなり削減できます。まぁ、最近はハードカバーで出した作品を3年後くらいに文庫化する時に、タイトルを変えて出すケースもあったりするのでまだ侮れないのですが……

さて、私の「記録」に関する方法は主に2つです。

デジタルな記録

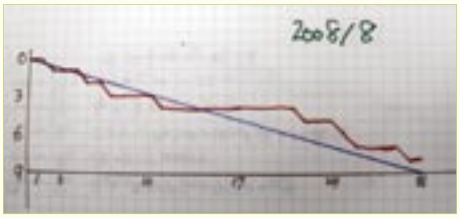
2007年末から読了本と積読本、そしていわゆるウィッシュリストの管理にMediaMarkerというサービスを利用しています(**図1**)。Amazon やbk1などのオンライン書店との連携はもちろん、全国の公共図書館検索も行えるのがお気に入りの機能です。利用している図書館に蔵書があるかを調べて予約まで行うのが非常にスムーズにできるのです。また、ケータイからもアクセスできるのは、店頭で「重複買い」を防ぐという第一の目的を満たすのに必須と言えます。書評の入力、タグ付け、読了日記録、評価ランク付けなどの記録に関する基本機能と、Twitter連携やブラウザ連携機能などもそろっていて使いやすいので、この手のサービスを探しているのなら一度試してみる価値はあろうかと思います。少なくとも本稿執筆時点では精力的に機能のアップデートがされているのも楽しみなところです。

■図1　MediaMarker（http://mediamarker.net/u/beakmark/）

アナログな記録

普段使っている手帳に、月ごとに「Book Burn Down Chart」を付けています(**図2**)。これは読書ペースの把握のために記録していて、一般的なバーンダウンチャートと違って「青線」は「予定線」ではありません。あくまでペースのベンチマークのために、「月に9冊読むとしたら」という仮定の線として引いてあります。どうやら私は年間100冊程度がペースなようなので、月当たりに換算すると8、9冊ということになります。

ポイントは青線から離れていても別に気にしないことです。本を読んでいないからといって焦ることはなく、その分の時間はきっと別のすてきなコトに費やしているのですから:-)。後で見返して「ああ、このころはプロジェクトが佳境だったからなぁ」なんてことを思い出すのも楽しいものです。



■図2　Book Burn Down Chart

この1冊

今回はMediaMarkerの図書館検索のおかげで手に取ることができた本として『残像に口紅を』（中央公論社、ISBN- 13：978-4122022874）という作品を紹介します。筒井康隆氏の代表作の1つなのでファンにはおなじみなのでしょうが、私は氏の長編作品をちゃんと読んだのはこの本が初めてでした。Wikipediaの説明を引用すると、「現実世界から文字が1文字ずつ消えていき、その文字を含む言葉は存在さえも失われていくという実験的小説」です。最後には50音のほとんどが失われますが、それでも文章を綴（つづ）っていける作者のパワフルさに感嘆するとともに、ソフトウェアエンジニアが直面する仕様に起因する実装上の制約について、つい考えてしまう1冊となっています。

Profile　プロフィール

あまのりょー
AMANO Ryo

最近は森博嗣さんのミステリィの中から「Gシリーズ」を順に攻め始めました。理系とはいえ、物理や数学にはあまり縁のなかった生物系の身としては、φやらθやらτやらと言われてもさっぱりピンと来ないのですが…。

連載

夜の旋律、君の傍らで

第2回 ◉ 酒井智巳
SAKAI Tomomi

見上げると眩暈がするような灰色のビル群の隙間をしばらく歩き、塀のある一戸建てが続く地域に出た。ここからだと彼女の家はすぐに分かる。懐古調の洋館は他にはほとんどないため、一際目を引くからだ。門のすぐ前まで来ると、ちょっとためらってから脇にあるインターホンのボタンを押した。名前と今日来た目的を伝えると、低い唸りとともに鉄格子の門がゆっくりと左右に分かれて奥に開いた。植木の間に続く石畳を歩いていくと、彼女のお母さんが玄関で出迎えていた。

居間に通された後、感情失調症について図書館で調べたことを要約して伝えた。

「それで、何かきっかけと思われるようなものがあれば、もしかしたら何かお手伝いできるかもしれないと考えたんです」

「そうね、変わったことといえば…」

お母さんは空中を見つめながら、何かを思い返している。

「ピアノを弾かなくなったことかしら」

「すみません、ピアノを弾くとはどういうことでしょうか」

「あぁ、そうね。ごめんなさい。ピアノなんて今時知っている人のほうが少ないでしょうね」

こちらへどうぞ、と言いながらソファーから立ち上がり、ドアを開けて部屋を出た。彼女の部屋から2つ手前にある部屋のドアを開け、中に入った。

「どうぞ」

そこは彼女の部屋の倍くらいの広さで、床にはメダリオン模様のペルシャ絨毯が敷かれていた。窓際から少し離れたところに、それはあった。

「これですけど」

お母さんは蓋を開けた。黒く、大きな、不思議な形の家具。

「もうずっと昔からうちにある…、100年以上前のものね。ここを押すと音が出るんです」

そう言って白黒の配列の1つを押した。鈍い音がした。お母さんはもう1回、強く押した。今度は明るい澄んだ音を立てた。余韻が部屋に響いた。

「これがピアノなんですね。音楽の中でこういう音が鳴っているのを思い出しました。でも何か違う…、綺麗なだけじゃなくて、何と言えばいいのか…、少し恐い音も含まれているように思います。生々しいというか」

お母さんはこちらを見て頷いたあと、窓の外に視線をやりながら話し始めた。

「昔はこういうもので音楽を創っていたのね。祖母がいた頃、あの子は本当に小さかったんですけど、よく一緒に弾いていたんです。何か、惹かれるものがあったんでしょうね。それ以来ずっと続いていたのに、突然やめてしまったの。感情失調症と判断される1〜2ヵ月前くらいに」

「症状が出た後にピアノを聴かせてあげたことはありますか」

「ないわ。祖母の他には、誰も弾けないの」

そうか。確かにそうだ。僕も音楽はプログラマがコンピュータに創らせるものだと思っていた。もともと人間が弾いていたのを、今はコンピュータが肩代わりしているだけなんだな。人間がコンピュータ並みの速度でピアノのキーボードを叩くのは大変だろう。習得に努力が必要なことは想像がつく。それでいつの間にかコンピュータが主流になったんだろう。ピアノを見つめながらそんなことを考えていたら、お母さんがこちらを向いて口を開いた。

「ピアノの演奏を再生できる機械が遺品の中にあるかもしれません」

玄関を出て、隣に建っている洋館に案内された。倉庫として使われているようだった。

「この部屋にあるはずなんですけど」

棚にある数多くの木箱を、2人で手分けして探していった。それがどんなものか知らないが、大半は衣服や食器といった明らかに違うと分かる物だから、判断はできる。金属製の箱型のものを見つけたので、何か聞いてみた。それです、とお母さんは答えた。一緒に入っていたのがスピーカーだということは僕にも分かった。基本的な構造は今のものと同じようだ。平面でなく、かなり厚みがあるが。

その時、部屋の入り口に誰かが来た。

「あ、お帰りなさい」

「ただいま」

彼女のお父さんのようだ。

「クラスメイトの方が心配して図書館で調べてきてくれたの。それで今探し物をしていたの」

「わざわざ、ありがとう。見つかったのかい、探していたものは。CDプレイヤー？」

お父さんは遺品についていくらか知っているようだ。

「生成されたものじゃない、生のピアノ演奏を再生したくて…」と切り出した。

「そうか。一応使い方は知っているよ、長い間動かしたことはないけれども」

そう言って、しゃがんで機材を確認した。立ち上がって奥の戸棚に向かい、いくつかある同じ形をした箱の中から1つを選んで取り出した。埃を払い、「piano」と手書きされた蓋を開けると、中にメディアらしきものが詰まっていた。

「これがCDなんだよ。今のようにオンラインで配信される前は、こういった音楽の入った媒体をわざわざ買っていたんだよ。これは祖父が骨董趣味で再生していたもので、もっとずっと昔のものなんだ」

「そんなに古いものなんですね…」

「じゃ、これを向こうへ持っていこうか。手伝ってくれるかな」

かなり重いのでお父さんと一緒に2往復し、彼女の部屋の前まで運んだ。ドアをノックして入ると、彼女は以前来たときと同じ場所で同じ格好をしていた。声を掛けても黙ったままだ。

「コンセントは今と同じなんだ」と言いながら、お父さんは配線を済ませた。最初にメディアを駆動するCDプレイヤーという機械のスイッチを入れた。緑色の文字が表示された。

「次はアンプだ」

左端にあるスイッチを押した。その一瞬後、ケースの中が明るく光り、何かを引き裂くような大きな音が続けざまにした。

「駄目だ！」そう叫びながらお父さんは慌ててコンセントを引き抜いた。アンプ上部のスリットから煙が立ち昇っていて、その周辺は黒焦げになっている。鼻を刺す匂いが部屋中に漂い始めたので、お母さんがすぐに窓を開けた。冷たい風が部屋に入ってきた。僕たちが作業をしているその間、隣にいる彼女は一度もこちらを見なかった。

「壊れていたんですか」と尋ねると、お父さんが返答をした。

「おそらく。もうずっと使っていなかったからね。秋葉原にでも行かないと修理できないだろう」

「秋葉原」

「第5層にある古い街で、今も古い機材好きが隠れて住み着いてるって話だ。ただ、知っての通り第4層以下は立ち入り禁止だ。犯罪者になってしまう」

僕は、アンプを見つめていた。

「このアンプお借りしてよろしいでしょうか」

「構わないけど、どうするの？」

「何とか自分で修理できないか、調べてみます」

翌日の授業が終わるとともに図書館に急ぎ、帰宅制限時間一杯まで調べた。しかし、サービスマニュアルは現代のものばかりで、大昔のものはともかく、少し前の資料であっても、不思議なくらい何もなかった。

すっかり暗くなった道を家に向かって歩きながら考えた。結局、秋葉原まで潜るしかないんだな。見つかったら、犯罪者か。でも、アンプの修理って悪いことなんだろうか。事情を話せば、分かってもらえるんじゃないか。

こんなことを書いているけど、本当はアンプを借りる時に薄々予感はあった。僕は秋葉原まで潜ってアンプを修理することになるだろう、ってね。

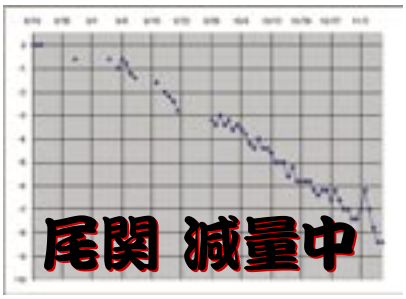
■個人広告

豆蔵ラボ
http://labo.mamezou.com

豆蔵Coラボは、豆蔵ソフト工学ラボの愛称です。
所長 羽生田栄一（豆蔵 取締役／フェロー）

すぐに役立てたいなら
301頁目から読むのを
オススメします。

永和システムマネジメント
コンサルティングセンター
天野 勝



執筆・講演承ります。

講演「オフィスで踏み出すRubyの世界」
（東京Ruby会議01）がニコニコ動画
「sm4367870」で配信中。

EM ZERO Vol.1、Vol.2 の記事もよろしく！

しば（koshiba@4038nullpointer.com）

年末年始は関西に帰省しています。
誰か遊んでください。

木下史彦（http://fkino.net）

orz

■バックナンバー

EM ZERO [イーエム・ゼロ] Vol.0
2008年7月1日発行

EM ZERO [イーエム・ゼロ] Vol.1
2008年9月5日発行

●編集後記

このたびなんとあのメジャー系IT情報サイト「アットマークアイティ」様にEM ZEROの活動をとりあげていただきました（http://jibun.atmarkit.co.jp/lcom01/rensai/border/05/01.html）。自分戦略研究所の岑様、取材をしてくださってどうもありがとうございます。これからエンジニア同士はもちろんのことメディア同士のつながりも深め、新しい時代を切り開く力を生み出していきたいと思います。その第一歩として、新コーナー「メディアコネクション」では岑様と、InfoQ Japanの田島様にご寄稿いただいておりますので、ぜひご覧ください。

今回の野口メソッド被害者は「とりあえずメソッド」をご執筆いただいた福井さんです。「福井情報技術者協会」という全国の福井と名の付くエンジニアが集うイベント、ではなく「へしこ」で有名な福井で行われたソフトウェアイベント後、飛行機待ちの福井さんを捕まえて「とりあえず」書いていただいた、懇親の力作です。これからはよりアジャイルにその場で原稿を書いていただく新手の編集詐欺、もとい編集方法を確立していきたいと考えております。今後ともご支援ご指導をよろしくお願い申し上げます。**（野口）**

編集者としては超初心者のわたくしですが、今回は大半の記事で最初の編集をさせていただきました。本誌の発行は3回目になりますが、少しは成長できているのかな？

新しいことを始めて試行錯誤しながらもやってみることは、ソフトウェア技術を学ぶのと同じでやっぱり楽しいことですね。本誌の発行を重ねていくことで、同じように自分自身も成長していければと思っています。執筆してくださったみなさま、読者のみなさま、ぜひこれからもご指導いただければと思います。ちなみに鬼編集長のスパルタはいらないです（^^;

編集との並行作業でしたが、10月下旬にマナスリンクのWebサイトをリニューアルしました。オンライン用のちょっとした記事の掲載や、個人広告のお申し込み、最新号5部無料お取り寄せ、広告掲載のお問い合わせがサイトからできるようになりました。ぜひご活用ください！

誌面、オンラインともども、EM ZEROとマナスリンクはみなさまに支えられて成長を始めたばかりのメディアです。これからもひとつよろしくお願いいたします。でわでわ。**（進藤）**

最後までお読みくださり、誠にありがとうございます。（あれ？いつのまにか、みんなXXぱんとパンダ族を主張していますね？笑）

今回のナマスヴィレッジは前回と変わり、縦型にしてみましたがいかがだったでしょうか？これからもいろいろとチャレンジしつつ、今後もっとナマスヴィレッジの面積を広げられるように頑張りたいと思います。豆パン君もトコトコと歩き出し、一歩一歩着実に皆さんの記憶の中に足跡を残し始めました（笑）。これからどんな冒険が待っているのでしょうか…ドキドキします（^^; でも楽しみながら、彼の成長を描き綴っていきたいと思います。

EM ZEROも回を重ねて…3回目の発行。これもこの冊子を手に取ってくださる方々、原稿を寄せてくださる方々、EM ZEROを愛してくださる方々のお陰だと深く感謝いたしております。ありがとうございます。こんな豆パン君が見たい！等のリクエストもお待ちいたしております（爆）。どうかお寄せくださいまし。2009年もよろしくお願いいたします！**（山崎）**

◎株式会社マナスリンクについて

株式会社マナスリンクはEM ZEROの運営を目的として設立された会社です。マナスとはサンスクリット語でマインドを意味します。良いマインドを持った人々をEM ZEROを通じて結び付け、良い人の流れ良い情報の流れを作り出し、ソフトウェア業界を盛り上げていくお手伝いをいたします。

◎EM ZERO配布のお願い

EM ZEROはイベントでの配布＆EM ZEROに共感してくださる方の草の根配布を拠り所としています。よろしければ本誌を何冊かお持ちいただき、周囲の方に紹介していただけると嬉しく思います。

◎広告出稿のお願い

EM ZEROでは広告を掲載してくださるクライアント様を募集しています。企業、団体、個人は問いません。EM ZEROの存続にご協力していただける方、広告効果の可能性を感じていただける方がいらっしゃいましたら、ぜひご相談させてください。

■個人広告のお申し込み

http://www.manaslink.com/ad_personal

■企業・団体広告のお問い合わせ

http://www.manaslink.com/ad_company

◎お取り寄せ

最新号5部を送料無料でお取り寄せいただくことができるようになりました。また、イベントや社内での配布用に、5部以上での送付も送料をご負担いただければ承ります。部数に限りがございますので、お早めにお申し込みください。

■EM ZEROお取り寄せフォーム

http://www.manaslink.com/send_req

EM ZERO［イーエム・ゼロ］Vol.2
2008年11月24日発行

編集長：野口隆史
編集：進藤寿雄、
豆パン君イラスト：山崎直子
デザイン：ミヤムラナオミ
表紙イラスト：miti

発行元：株式会社マナスリンク
〒152-0034
東京都目黒区緑が丘2-3-8
http://www.manaslink.com/
お問い合わせ先：contact@manaslink.com

印刷所：佐川印刷株式会社
http://www.sakawa.jp/

EM ZERO [イーエム・ゼロ]　Vol.02
2008年11月24日発行

発行 ： 株式会社マナスリンク
〒152-0034
東京都目黒区緑が丘2-3-8
http://www.manaslink.com/
お問い合わせ先 ： contact@manaslink.com

ManasLink

たまり場
～EM ZERO 応援ソング～

信じてきたこの長い坂を
あとどれだけ歩いていくのか？
その想いが叶わぬうちは
どうかあきらめないで
泣きたいときも笑いたいときも
君を待ってるたまり場だよ
そうひとりひとりは小さいけれど
ここにはたくさんの仲間がいる
君と僕が手をつないだら
明日への虹になるよ
二度とこない時間の中で
誰もがみな道を迷ってる
「もしかしたら迷ってるのは自分だけじゃないか？」
なんて
君の笑顔や君の優しさを
待ってる人がいるたまり場だよ
ひとりひとりは小さいけれど
ここにはたくさんの仲間がいる
みんなが手と手をつないだら
世界は一つになるよ♪
Lalala...
信じてきたこの長い坂を
僕はまた歩き出す

song by 侍塊 s（2008）
URL：http://katamaris.jp/