

[特別記事] 案件にマッチした最適なプロセスを組み立てよう



もう悩まない! 開発プロセスの 作り方

文=株式会社オージス総研
正木威寛 (PMP) MASAKI, Takehiro
山内奈央子 (PMP) YAMAUCHI, Naoko

ITシステムの大規模化・複雑化に伴い、分業を前提としたソフトウェア開発が当たり前になってきています。異なる組織、異なる役割の人たちがチームを組み、プロジェクトを進めるうえで肝になるのは、言うまでもなく開発プロセスです。「明文化されたルールがなく、手探りで開発をしている」「どのような作業成果物を作ればよいのかわからない」と悩んでいる方はいないでしょうか？ 本稿では、そうした悩みを解決するために、「開発プロセスの正しい作り方」を説明します。



プロジェクトを「見える化」する

現在、企業では日本版SOX法^{注1}への対応などにより、企業活動すなわち業務の透明性が求められており、ビジネスモデリングによって業務を「見える化」する取り組みが行われつつあります。一方で、そのような企業からシステム開発を受注するSIerやソフトハウスに対しても、これまでの短納期・低コスト・高品質だけでなく、プロジェクトの正当性や透明性が求められるようになってきています。これらすべてのニーズを満たすには、「明文化された開発プロセス」が必要です。開発プロセスによってプロジェクトの活動が「見える化」され、その正当性や透明性を示すことができます。また「見える化」されたプロジェクトの活動は、納期・コスト・品質に対するブレの幅を狭めると共に、それらから逸脱してしまうリスクを抑えることができます。

本稿では、プロジェクトマネジャーやプロジェクトリーダーなど、開発プロセスを策定・実践する立場にある方を対象とし、策定のための事前知識や策定作業の進め方、さらに適切な開発プロセスを作るためのポイントを説明します。



開発プロセスの意義と役割

筆者は、エンジニアやプロジェクトマネジャー、管理職、システム開発を依頼するユーザーなど、何らかのかたちで開発に関わるさまざまな方と接する機会があります。いろいろな企業文化や立場、経歴を持つ方々とお話していると、開発プロセスに対して実にさまざまな認識があると感じます。そこでまず、「ソフトウェア開発における開発プロセスがどのようなものか」「開発

のライフサイクルにおいて開発プロセスはどのような役割を果たすのか」について説明します。

開発プロセスは「リサーチ+創作+製造」

「開発プロセスとは何か」について結論から先に述べると、次のようになります。

開発プロセスとは、ソフトウェアを開発するためのリサーチ+創作+製造の3つの局面を併せ持つプロセス

そして開発プロセスは、この3つの局面でソフトウェアを作るために必要な連続した作業の集まりで成り立っています。

開発プロセスを、ちょうど料理の「レシピ」に近いと感じている方がいらっしゃるかもしれません。レシピは、レシピどおりに調理（作業）を進めれば、いつもほぼ同じ料理が作れます。ですが、開発プロセスはちょっと違います。この違いは何でしょう？ レシピは「いつも同じ料理を作る」ための調理のプロセスですが、ソフトウェア開発では「いつも同じソフトウェアを作る」わけではありません。

開発プロセスを料理に関する活動に置き換えてみると、次のような局面があります。

- ①どのような料理が求められているか調べる
- ②その料理を作るためのレシピを考える
- ③レシピどおりに調理する

このように、レシピどおりに調理してソフトウェアを作るということとは、開発プロセスの一部でしかありません。ソフトウェア開発

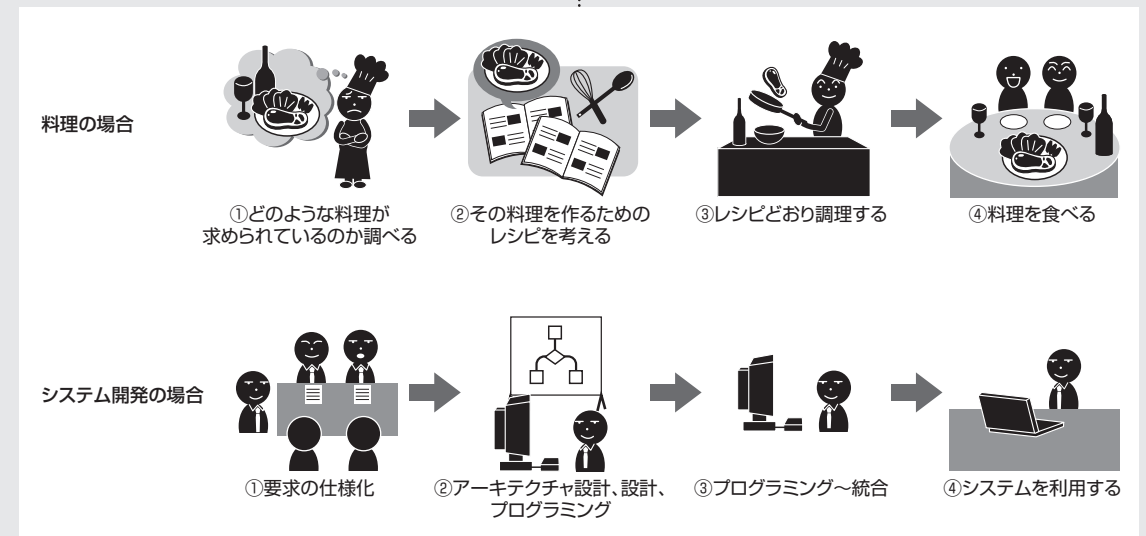


図1 料理とシステム開発のプロセス

が難しいと言われる理由の1つには、③の「調理する」ことよりも、①の「どのような料理が求められているか調べる」ことと、②の「その料理を作るためのレシピを考える」ことを、毎回毎回プロジェクトが立ち上がるたびに行わなければならないことがあります。「どのような料理が求められているか調べる」とは、お客さまがどのような料理を求めているのか考える、あるいはお客さまからどのような料理を食べたいのか直接聞き出し、要求を仕様化することを指します。たとえば、お店の新メニューとしてどのような料理を加えるべきか考えることが、これに当てはまると思います。

次に、②の「その料理を作るためのレシピを考える」とは、求められる味や香り、食感、食べやすさといった料理としての品質を実現するために必要な食材や、調理の手順を考えて仕様化する（レシピを創る）ことを意味します。

話をソフトウェア開発に戻して、これら料理の局面をソフトウェア開発のプロセスに当てはめると、次のようになり、両者を比較すると図1のようになります。

- ① 要求の仕様化（要求分析）
- ② アーキテクチャ設計、設計、プログラミング
- ③ プログラミング～統合

お客さまに当たるユーザーは、現行業務上の問題の解決や、新規ビジネスの立ち上げなど、さまざまな目的を持ってシステムを求めます。要求分析では、こうしたユーザーのニーズをも

とに、ソフトウェア要求を分析して仕様化します。エンジニアの要求分析の上手い・下手、つまりユーザーのニーズをどれだけ収集できるか、収集したニーズを上手く整理できるか、要求仕様としてきちんと仕様化できるかは、ここで現れます。

次にそれを実現するレシピを創ります。“創る”と述べたのは、この活動こそがエンジニアがエンジニアとして知恵を絞る創作活動だからです。このレシピ創りには、レベルによってアーキテクチャ設計、設計、プログラミングが当てはまります。エンジニアの設計の上手い・下手、プログラミングの上手い・下手はここで現れます。

続く調理に当たる部分では、実装したプログラムを統合して実行できる状態（≒完成した料理）にします。

以上のように、ソフトウェア開発における開発プロセスとは、たんに製造するプロセスではなく、リサーチのプロセス、創作のプロセス、製造のプロセス、この3つの局面を併せ持ったプロセスなのです。開発プロセスを広義に捉えたと、さらに品質の観点からレビューやテストが各プロセスで行われますし、スケジュールやコストといったプロジェクトマネジメントの観点での活動も行われます。これらについては後で述べます。

プロジェクトマネジメントと開発プロセス

新規システムの開発やシステムの再構築のほとんどは、プロ

注1 企業の会計不祥事などを防止するため、会計監査制度の充実と内部統制の強化を求める法規制。

ジェクトとして立ち上げられます。プロジェクトでは、当然ながら開発範囲(スコープ)・スケジュール・コスト・品質といったことをマネジメントする必要が生じます。プロジェクトマネジメントと開発プロセスを別々のものとして考えているプロジェクトを見かけることがありますが、実際にはそれぞれは異なる視点で、とても強く結びついています。開発プロセスの連続した作業は、PMBOK^{注2}で言う「プロジェクトがマネジメントすべきアクティビティとその順序」そのものであり、プロジェクト計画で使われるWBS^{注3}の根拠になります。したがって、プロジェクトで開発プロセスを策定しているということは、プロジェクト計画の一部を行っていることになるのです。もし開発を行うあなたの組織に、ある程度いつも繰り返せるほど根づいた開発プロセスが出来上がってきたら、その作業をWBSテンプレートとして用意しておき、プロジェクト計画の際に再利用することができます。組織に根づいた開発プロセスがあるということは、メンバーであるエンジニアに迷いを与えず、開発に関与する利害関係者に安心感をもたらします。

ソフトウェアの品質と開発プロセス

前述したとおり、品質管理もプロジェクトマネジメントの役割のひとつです。ソフトウェアに求められる品質に応じて、開発プロセスで行われる品質管理の活動、つまりレビューやテストも調整する必要があります。たとえば、「要求仕様書・設計書・ソースコードなどのレビューをどれだけやるのか?」「単体テスト・統合テスト・シ

ステムテストなど、いくつのテストレベルに分けるのか?」「それぞれのテスト網羅性をどのぐらいに設定するのか?」といったことを開発プロセスとして決めておくのです。

求められる品質が低いシステムであれば、レビューやテストをする必要性は低くなり、それ以上にレビューやテストを行うことは過剰品質のシステムを作ることになります。過剰品質自体は、出来上がったシステムを運用するうえでは問題ないのですが、開発に余計な時間やコストをかけているという点で、カットオーバーの遅延や追加コストの発生といった好ましくない状況を生むこともあります。このような状況は、レビューやテストについて開発プロセスとして決めたのではなく、それぞれのエンジニア任せにしているケースで見られます。

契約と開発プロセス

プロジェクトによっては、システムへの要求を出すユーザーの組織と、そのシステムの全部あるいは一部を開発する組織が異なることがあります。つまり、ユーザー企業とSIerのような関係です^{注4}。この場合、両者の間では、システム調達(SIerから見るとシステム開発)に関する契約が発生します。ここで言う契約とは、たんに契約書だけを指すのではなく、もっと広義の意味でプロジェクトの開発範囲やスケジュール、コスト、品質、納品するドキュメントなどの作業成果物に関する取り決めも含まれます。両者の契約上の境界を明確にするものは、プロジェクト計画です。そし

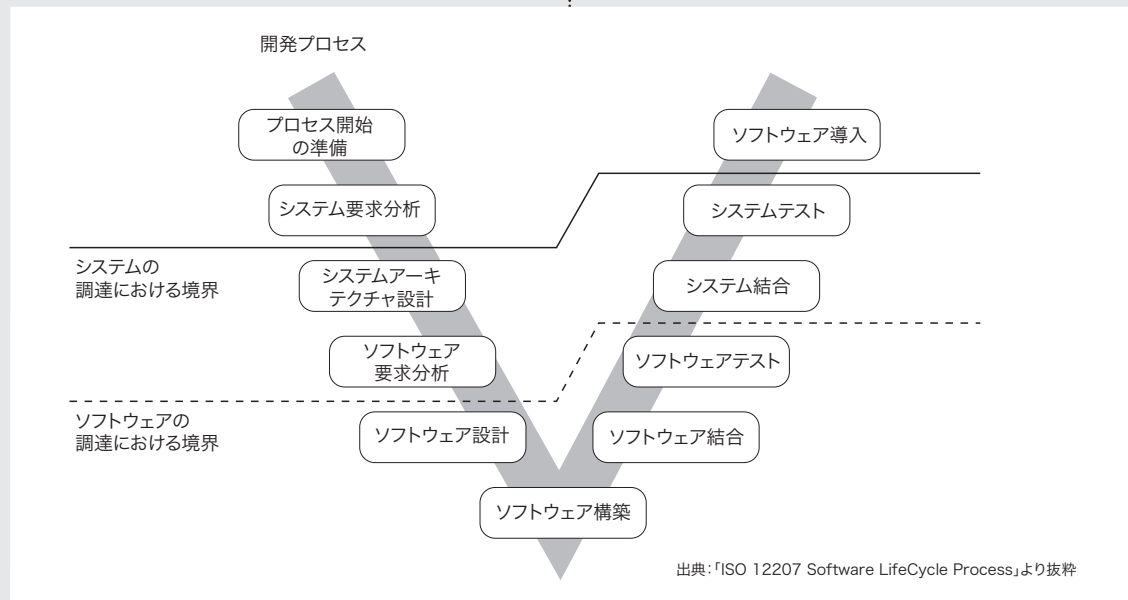


図2 システム開発のV字モデル

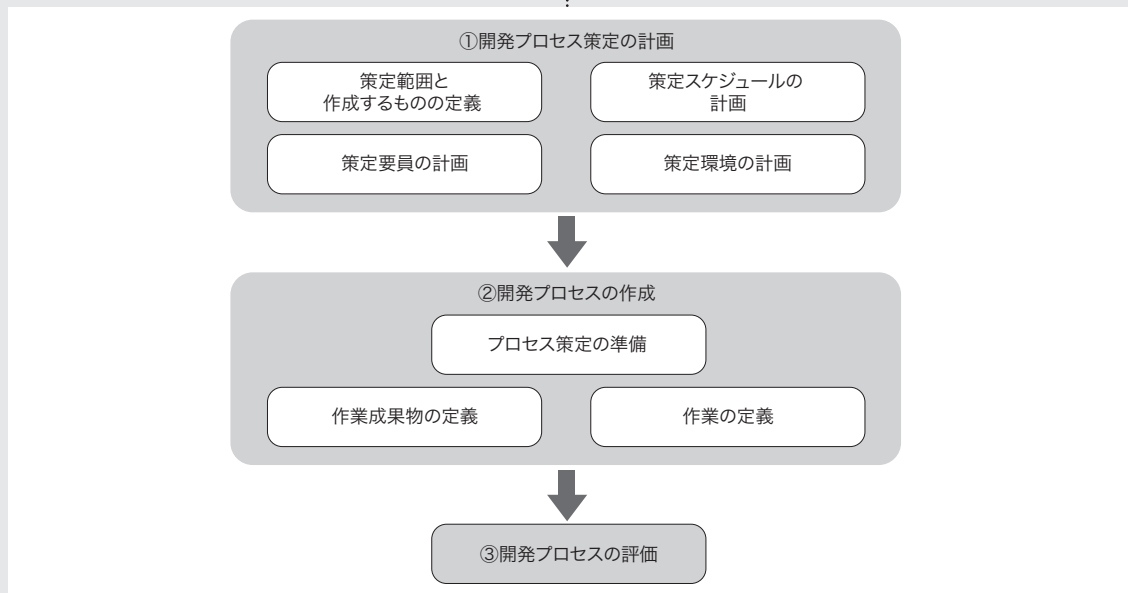


図3 プロセス定義作業の概観

て、開発プロセスが明確になっていることは、契約上のトラブルを防止する効果があります(図2)。特に開発範囲を依頼する境界と開発終了時の境界を、開発プロセスとして明確にするように心がける必要があります。この境界は、ユーザー企業(SIerから見るとお客さま)が調達するモノの要求仕様と、受け入れの根拠となるからです。プロジェクトの当初に明確にできない部分については、プロジェクトのどの段階で、何(作業、レビュー、テストレベル、作業成果物など)が明確にできるのかについて合意しておきます。システム調達においても、開発を請け負う側のSIerの開発プロセスを明確にしていることは、調達する側(お客さま)に安心感を与えます。

開発プロセス策定作業の全体像

では、開発プロセスを策定する作業の流れを概観しましょう。図3に示すとおり、開発プロセスの策定もシステム開発と同じように、①計画し、②作成し、③評価するという流れになります。では、それぞれの作業を順に見ていきましょう。

① 開発プロセス策定の計画

開発プロセスの策定として、「何を、いつまでに、誰が、どのように」作っていくのかを計画します。つまり、開発プロセス策定のプロセスを決めるのです。ここでは、以下の作業を行います。

策定範囲と作成するものの定義

この作業の目的は、開発プロセスの策定で「何を作るのか」を決定することです。開発プロセスの策定の範囲は、「契約と開発プロセス」の部分で説明したように、請け負った契約の範囲によって異なります。開発プロセスの範囲を決めるために、ベースとして表1(次頁)のような開発プロセスを利用すると効率よく策定でき、策定範囲や契約の範囲が明確になります。開発プロセスの策定で作成するものには、表2(次頁)のようなガイドラインやテンプレートがあります。

策定スケジュールの計画

この作業の目的は、開発プロセスで作成するガイドラインやテンプレートを「いつまでに」作成するのを決定することです。この策定のスケジュールは、プロジェクト計画に組み込まれ、開発のスケジュールと合致している必要があります。前述のような、開発を依頼したユーザーと開発するSIerとの関係であれば、契約を結

注2 Project Management Body of Knowledge。プロジェクトマネジメントの知識体系。事実上の標準として世界中で広く使われています。

注3 Work Breakdown Structure。プロジェクトのスコープを要素成果物(システムの一部や作業成果物)や作業(アクティビティ)へ分解し、階層的に表現したもの。スケジュールは、WBSをもとに作業の順序や時期を定義することで立てられます。

注4 以降では、ユーザー企業がSIerに開発を依頼するケースを前提に説明していきます。

ベースとなる開発プロセス	説明
社内の既存のプロセス	社内のプロセスがある場合、選択肢の候補になる。使い慣れたプロセスであれば、エンジニアが学習する負荷が減るというメリットがあるが、プロジェクトで使おうとする技術に合わない場合（たとえば、ホストのプロセスをオブジェクト指向開発に適用しようとするなど）、かえって使いづらいプロセスになるリスクもある。
ISOやJISなどの規格	国際規格のISO12207といった開発プロセスの規格。複数の会社に関わる場合などは、自社独特の用語の利用から発生するミスコミュニケーションのリスクを減らすことができる。
市販されているプロセス	RUPなどプロセス製品や、XP、SCRUM、Crystal、FDDといった開発プロセスに関する書籍。プロセス製品の場合は、ライセンス料がかかる。

表1 ベースとなる開発プロセスの例

ぶときに開発プロセス策定が完了していることが理想です。

策定要員の計画

この作業の目的は、決定した開発プロセス策定スケジュールの個々の作業を「誰が」行うのかを決定することです。「プロジェクトマネジメントとプロセス」の部分で述べたとおり、開発プロセス策定はプロジェクトマネジメントの一環として行われます。そのため、この作業はプロジェクトマネジャーやプロジェクトリーダーが兼務することもありますし、プロジェクトマネジメントスタッフの一員として専任の担当者（プロセスエンジニア）を割り当てることもあります。また、策定した開発プロセスが誰の承認によって公式リリースとするのかも決めます。これを曖昧にしていると、開発を依頼したユーザーやエンジニアなど、さまざまな利害関係者がそれぞれの立場で議論を始め、開発プロセス策定が進まない状況に陥ることがあります。

策定環境の計画

この作業の目的は、開発プロセスを策定するための手段、すなわち「どのように」策定するかを決定することです。開発プロセス策定に、開発環境に依存したガイドラインやテンプレートが含まれる場合には、開発環境が用意できていないと作れないこ

作成するもの	説明
作業成果物のガイドライン	開発によって作成される作業成果物とそのルールを定義したもの。様式や記述すべき内容、UMLなどのダイアグラムの表記に関するルールが含まれる。
作業手順のガイドライン	どのような手順で作業を進めるのかを定義したもの。作業の順序、作業の目的、作業のインプットやアウトプットとなる作業成果物を記載する。比較的軽量なプロセスであれば、「作業成果物のガイドライン」に含まれることもある。
テンプレート	作業成果物を作成するためのテンプレート。ExcelやWordなどのドキュメントテンプレートや、モデリングツールのモデルやドキュメントテンプレート。利便性を向上させるためにマクロなどのプログラムが含まれることがある。
ツールの利用ガイドライン	開発で用いるツールの使い方とそのルールを定義したもの。UMLモデリングツールやテストングツールなど、多機能なツールを利用する場合に作成する。他のものよりも策定に工数がかかることが多い。利便性や開発プロセスへの適合度を上げるために、これと併せてツールのアドインを開発することがある。

表2 開発プロセスで作成するもの

とがありますので、用いるツールやソフトウェアといった開発環境の調達時期を考慮した計画を立てなければなりません。同様に、開発プロセス策定で作成するもののための構成管理ツールや、策定したガイドラインをエンジニアに見てもらうための環境を整える計画も立てます。

② 開発プロセスの作成

開発プロセス策定計画に沿って、開発プロセスのガイドラインやテンプレートを作っていきます。ここでは、以下の作業を行います。

プロセス策定の準備

表3に示すような開発プロセス策定のためのガイドラインやテンプレート、策定のためのツールや環境を整えます。用語の不整合など、品質に問題があるドキュメントは、エンジニアの読む気も薄れてしまいます。簡単なものでもかまわないのでガイドラインを作成しておくといでしょう。このような策定のためのガイドラインの用語や文体などは、後で修正するのは手間暇がかかりますので、最初に決めておくことが肝要です。

準備するもの	説明
ドキュメントガイドライン	専門用語の用語集（特に訳語）、文章は「です・ます調」で書くのか「だ・である調」で書くのかといった文体や送り仮名のルール、バージョン番号の振り方などに関するガイドラインの文書管理ルール。
ドキュメントテンプレート	開発プロセスのガイドラインを作成するためのソフトウェアのテンプレート。たとえば、WordやExcelのテンプレート。見出しや本文のフォント、セクション番号の振り方などをテンプレート化しておくと便利。
ツールや環境	開発プロセスのガイドラインやテンプレートを作成するために必要なツールや環境。たとえば、Visioなどのドローイングツール、UMLやER図のモデリングツール、JustRightなどの文書校正ソフト、CVSなどのバージョン管理ツールなど。

表3 準備するものの例

作業成果物の定義

開発プロセスを実行することで生み出される作業成果物の様式や記述ルールなどを、「作業成果物のガイドライン」として定義します。この作業は、次に説明する「作業の定義」と並行して進めます。最低でも、プロジェクトの公式な作業成果物と決めたものについてはガイドラインを作成します。どのような作業成果物が「公式」となるのかは、前述したプロジェクトマネジメントや品質、契約などによって決まります。

作業の定義

前述の「作業成果物のガイドライン」として定義した作業成果物をアウトプットする作業を、「作業手順のガイドライン」として定義します。ここでは、作業の目的や作業担当者の役割、作業のインプット、作業ステップ、作業のアウトプットを決めていきます。

③ 開発プロセスの評価

開発プロセスの策定は、ガイドラインやテンプレートが出来たらお役御免というわけではありません。プロジェクトを進めていくにつれて「この作業がわからない」「この作業は利用するツールでもっと簡単にできる」といった開発プロセスの問題や要望、すなわちエンジニアからのフィードバックがあるものです。このよ

うな問題や要望はバックログとして管理し、必要に応じて開発中にもガイドラインやテンプレートへ反映していきます。



開発プロセス策定・実践の勘所

開発プロセスが何も明文化されておらず、プロジェクトの中で漠然と場当たり的に作業が行われている組織があります。アジャイルのような軽量プロセスであれ、そのような組織で「明文化された開発プロセス」を策定し、実践していくためには、意識しておかなければならないポイントがあります。そこで、筆者が開発プロセスを何度も策定してきた中で経験的に得たポイントを、「開発プロセスの策定」「プロジェクト計画」「開発中～プロジェクト終了」の3段階に分けて紹介したいと思います。

開発プロセスの策定におけるポイント

まずは、開発プロセスの策定に関する7つのポイントを取り上げます。

Point01 現在のプロジェクトの能力を分析する

どんなにきめ細かく文書化した開発プロセスでも、エンジニアが実践できなければ意味がありません。それまでにエンジニアが実践してきた開発のやり方について、よく情報を収集しておきましょう。もし、プロジェクトメンバーがこれまで明文化された開発プロセスを実行していないのであれば、「これなら実践できる」という“ほどほど感^{注5)}”のある開発プロセスにすることを意識し、そこから始める必要があります。あるいは網羅的に策定した開発プロセスを、プロジェクトの習熟度を見ながら段階的に適用することを考えるのもよいでしょう。策定するうえで意識しなければならないのは、理想と、実際のエンジニアの開発に対する成熟度との間で妥協点を見つけ、スケジュール・コスト・品質といったプロジェクトマネジメントの観点で、以前よりは健全なプロジェクトになることを目指すことです。プロジェクトの規模が大きければ大きいほど、一足飛びに理想に近づくことはできません。

Point02 プロジェクトの規模を分析する

プロジェクトの規模によって、開発プロセスでどこまで統制すべきかが変わります。少人数、たとえば5人程度のプロジェクトで

.....
注5 アジャイル系の開発プロセスでは、ここで述べた“ほどほど感”を表す言葉として、「best enough」ではなく、「just enough」や「good enough」といった表現がよく使われています。

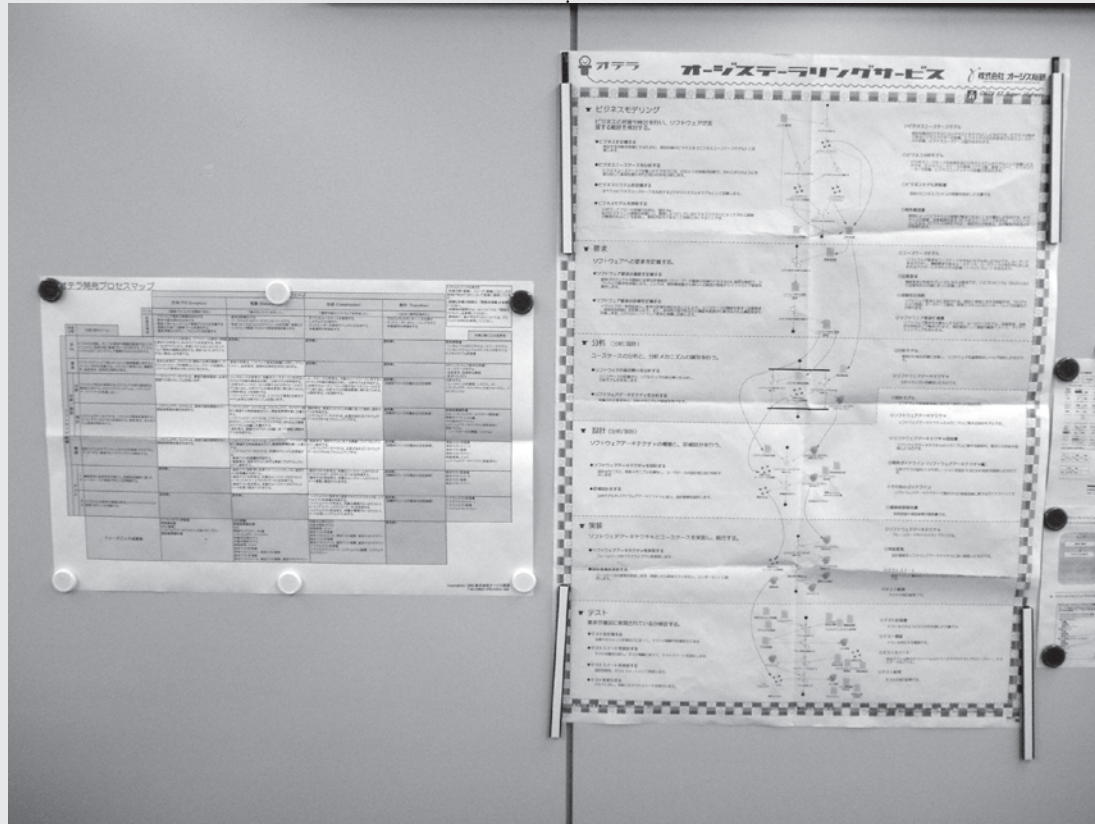


図4 プロセスマップ(左)と開発プロセスのフロー(右)

あれば、開発プロセスの作業内容の多くは口頭でも合意できます。またプロジェクトの最中でも、各自がどのような状況なのかはスタンドアップミーティング^{注6}で確認できますし、レビューもピアレビュー^{注7}やペアプログラミング^{注8}のような非公式なもので十分です。さらにユーザーとのコミュニケーションも、オンサイトカスタマー^{注9}のような手段で迅速にできることがあります。

一方で、プロジェクトの規模が何十人、何百人となってくるとそう上手くはいきません。何十人、何百人に対して開発プロセスの作業内容を伝え、実践してもらうには、開発プロセスの文書化は最低限必要です。また開発中のプロジェクトの状況も、メンバー・チーム全体と階層的に収集する仕組みが必要です。要求仕様書・設計書・プログラムなどの作業成果物の出来映えを評価するには、公式なレビューや公式なテストとその結果の収集が必要です。そして、それらを正確にユーザーやユーザーの責任者へ公式に報告し、合意できるような手続きが必要になってきます。開発プロセスの策定にあたっては、プロジェクトの規模を分析し、プロジェクトの規模に合致したチーム分けや、報告の手段や手続きなどの統制を行いましう。

Point03 トップダウンで策定する

コーディング規約や、要求仕様書や設計書などのドキュメントテンプレート、ツールの選択といった観点から、ボトムアップで開発プロセスを策定するケースがあります。しかしそれよりも、まずシステム開発というライフサイクルの大きな流れから考えて、それぞれを詳細化していくと、全体の整合性がとれた良いプロセスになります。また、そのような開発ライフサイクルを俯瞰できるような図やドキュメントは、読み手であるエンジニアにとってもプロセス全体が把握しやすく、プロジェクトにとって有効なツールとなります。筆者は図4のようなプロセス全体のフローや、プロセスマップと呼んでいる表を大きなサイズで印刷して使います。これをプロジェクトの利害関係者への説明に活用したり、壁に貼ってメンバーがいつでも「いま自分たちは開発ライフサイクルのどこにいるのか」を確認できるようにするのです。

Point04 アーキテクトのアドバイスを得る

開発プロセスの各作業の詳細を決める際に考慮しなければならないのが、UMLやER図のモデリングツール、それにフレーム

ワークを利用する開発です。これらは、プロジェクトの立ち上げ時に利用の有無や、どのツールやフレームワークを採用するのかといった選定ができないことが多いからです。もし、モデリングツールを利用するのであれば、そのツールにより設計の質の一貫性を保ち、設計書の一部となるモデルを印刷し、さらにコードを生成することで開発効率のアップが期待できます。こうした効果を開発プロセスが阻害しないようにしなければいけません。

たとえば筆者は以前、こんなケースを見たことがあります。あるプロジェクトでは、利用するツールの印刷機能に対応していない設計書の様式を採用していました。開発プロセスがその様式を定めているからです。そのため、エンジニアがわざわざツールから設計書の様式にコピー＆ペーストするという無用な手間をかけていました。

また、こんなケースもあります。それは、O/Rマッピングのフレームワークと、それに関連するコード生成ツールを利用するプロジェクトでした。採用したツールでは、DBスキーマからコードは生成できますが、設計書が生成できないために、エンジニアがわざわざリバースエンジニアリングして、使われない設計書を作成していたのです。

こうした例はまだあります。たとえば、すでに実績があり枯れたフレームワークなのに、そのフレームワークに対する余計なホワイトボックステストがテスト計画に入っていたりといった具合です。

いずれのケースでも、客観的に見ると無駄な手間がかかっています。信じられないことですが、「プロジェクトに合った開発プロセスを用意しよう」というスタンスの欠けたプロジェクトでは、こうした例が実際に多く見られます。これを避けるには、ツールやフレームワークを選定する立場にあるアーキテクトと協力して、プロセスの細部を策定することが重要です。そうしなければ、無駄な手間がかかるためにスケジュール遅延やコスト超過を招くだけでなく、プロジェクトメンバーの士気の低下にもつながりかねません。

Point05 例外ケースを詰め込みすぎない

システム開発は、要求される内容や利用する技術などにより、さまざまなケースが考えられます。それらすべてのケースを網羅したガイドラインを作成することは不可能です。しかしながら、より良いガイドラインを作ろうとして、思いつくすべてのケースや、エンジニアから報告のあったガイドラインに記載していないケースまで書こうとしてしまいがちです。このようにしてガイドラインをどんどん充実させていくと、結局はエンジニアが読む気を失

うほど膨大な文書量になってしまいます。ガイドラインには作業ステップではなく、作業の目的を明確に記載することに重点を置き、ある程度の例外ケースへの対応は実践するエンジニアへ委ねるとよいでしょう。

Point06 タイムボックスで策定する

開発プロセスの策定は、タイムリーに行わなければいけません。プロジェクトが、次に何かに取りかかろうとするときには、その工程に関する開発プロセスの詳細が策定できている必要があります。そうでなければ、プロジェクトは場当たりに進行せざるを得なくなります。

策定を進めるコツとして、次のことを覚えておくといでしょう。つまり、プロジェクト計画のマスタースケジュールを策定するタイミングで、開発プロセスのアウトライン(開発ライフサイクル)を決め、詳細はそれぞれのフェーズや工程の開始までに必ず間に合うように策定するのです。これは、アジャイル系の開発プロセスで実践されている「タイムボックス開発^{注10}」と同じ手法です。机上ではわからないことがある場合でも、策定の手を止めてわかるまで検討するのではなく、わかる範囲で必ず間に合うように策定を終了させることを優先してください。途中までしか書いていない「動かない開発プロセス」よりも、「そこそこだけと出来上がった開発プロセス」を作り、後は実践の場でPDCAサイクルを回して改善していくほうが、プロジェクトとして上手くいくものです。筆者の経験では、前掲した図2の工程1つにつき、2週間程度のタイムボックスで策定を終えることが多いと思います。

Point07 マネジメントツールを活用する

たんに文章として書かれた開発プロセスをプロジェクトメンバーが実行するよりも、開発プロセスを可視化するツールを活用するほうが有効です。こうすることで、メンバーが開発プロセ

注6 メンバーが立ったまま行う短時間のミーティング。各自の状況の報告と、スケジュールなどのプロジェクト計画の確認や問題に対する是正を行います。アジャイルプロセスの1つであるSCRUMでは、これをスクラムと呼んでいます。日本では朝会と呼ぶこともあります。

注7 先輩エンジニアなど技術的に優れた者と、1対1で技術的なことについて非公式に行うレビュー。

注8 2人1組で1つのプログラムと一緒に作成すること。技術的な意見交換を即時に行い、質の高いプログラムを短時間で作ることができます。

注9 プロジェクトメンバーとしてユーザーを加えること。要求仕様についての合意や、実現したシステムの評価を迅速に行うことができます。

注10 反復型の開発プロセスのルーツと言えるジェームス・マーチンの「RAD(Rapid Application Development)」で紹介されたプラクティス。1つの反復の期間を決めて、その期間内で作業を終了させる手法。

案件にマッチした最適なプロセスを組み立てよう

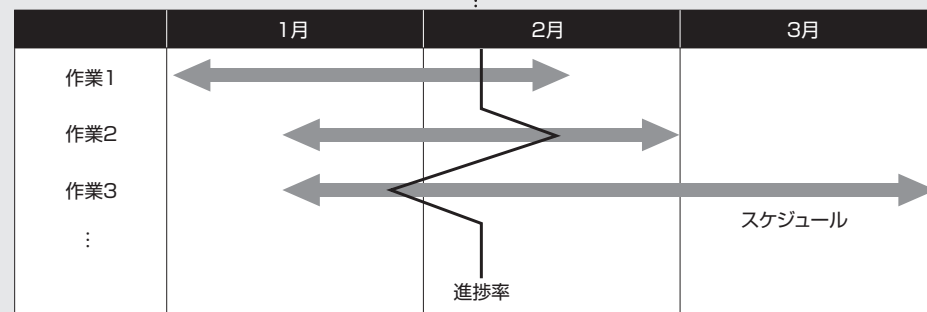


図5 イナズマ線



図6 ダウンチャート

スのもとにプロジェクトを実行するモチベーションを高めることが
 できるからです。プロジェクトの状態をビジュアルに表すツールと
 しては、イナズマ線(図5)やダウンチャート(図6)、オープンク
 ローズチャート(図7)などのグラフがあります。さらに、これらをプ
 ロジェクトメンバーにアピールするためのハードウェアもあります。
 たとえば、グラフを壁に大きく貼り出すために、A1版に対応した
 大判プリンターを利用するのもよいでしょう。現在では、比較的
 安く(20万円以下)入手できます。また、小規模なプロジェクトで
 軽い開発プロセスを採用するのであれば、壁かけ式のホワイト
 ボード(2万円程度)に直接記入してもよいでしょう。

プロジェクト計画におけるポイント

次に、プロジェクト計画を立てる際のポイントを紹介しましょう。

Point01 学習にかかる時間を忘れない

どんなに優れた開発プロセスを策定しても、それを実践するプロジェクトメンバーが慣れていなければ、いきなり効率がアップ

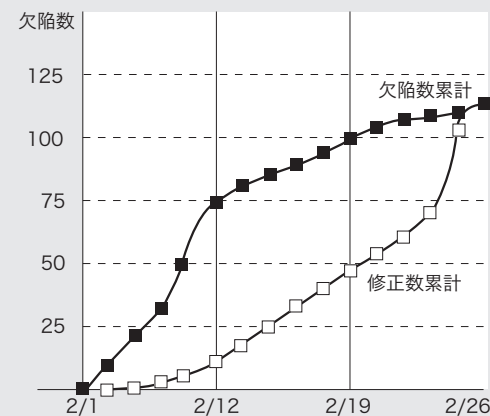


図7 オープンクローズチャート

したりすることは稀です。開発プロセスが本来のパフォーマンスを発揮するには、メンバーが学習し、習熟する時間が必要だからです。策定した開発プロセスを初めて適用するプロジェクトであれば、スケジュールを立てる際に、メンバーの学習に要する時間も考慮した見積りをしてください。

Point02 開発プロセスの説明会を計画する

メンバーが開発プロセスにあまり慣れていない場合は、たんにプロジェクト計画のスケジュールやWBSを説明するだけでは、開発プロセスの概観である開発ライフサイクルのイメージが伝わらないものです。開発ライフサイクルをイメージできるかどうかは、とても重要です。これができないメンバーは、不安が募ったりモチベーションが下がってしまうからです。プロジェクトの初期にプロジェクト計画を説明する際に、まずは開発ライフサイクルと直近の作業内容について説明会を開きましょう。

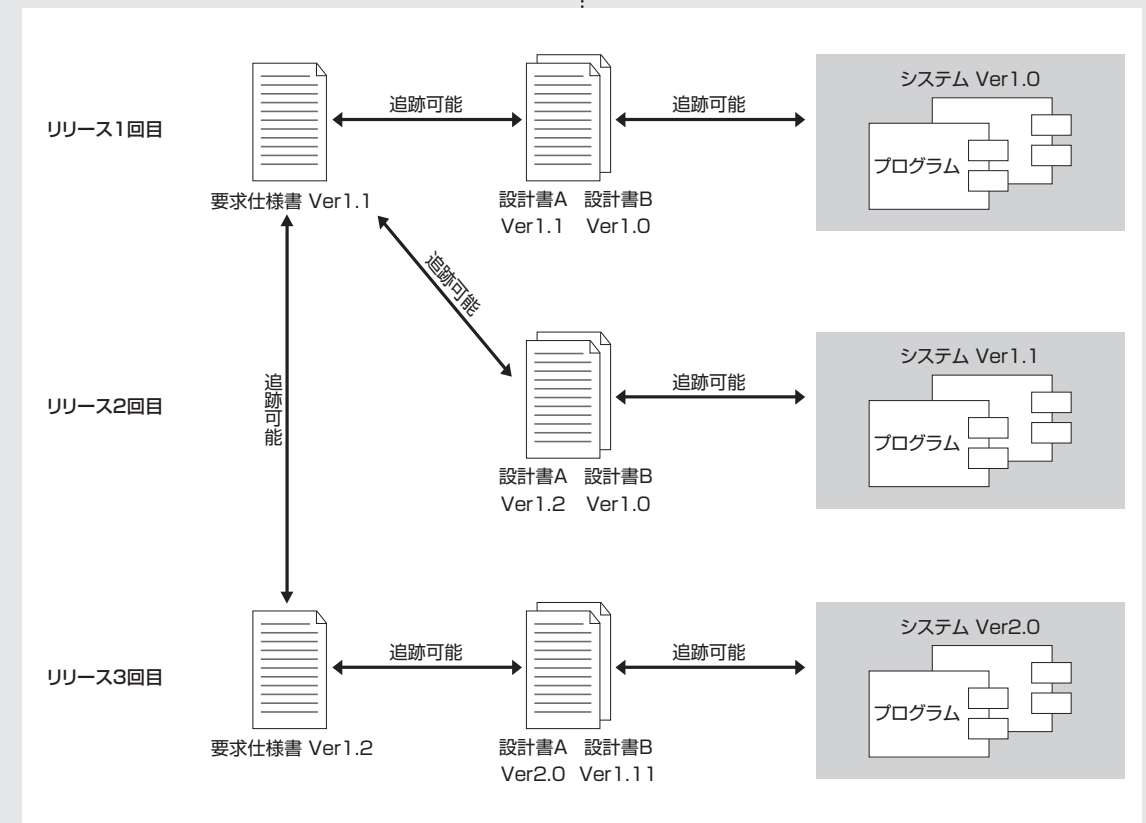


図8 作業成果物の構成管理

Point03 疑問を共有するための工夫をする

メンバーがあまり慣れていない開発プロセスを実践すると、要求を収集するためのインタビュー技法や、アーキテクチャそのものなど、本来は開発プロセスの範疇に含まれないことにまで気がいってしまい、開発プロセスに対してどんどん疑問が湧いてきます。また、プロジェクトで策定したばかりの開発プロセスを実践していくと、机上でいくら念入りに検討したとしても、例外は発生するものです。

エンジニアが疑問を持ったら、それを相談して答えが得られる場を用意しておくことはたいへん重要です。小規模なプロジェクトであれば、その都度口頭で説明したり、スタンドアップミーティングの際にホワイトボードに書いて情報を共有すればよいかもしれません。ところが、数十名程度のプロジェクトになると、そう上手くはいきません。そうした場合、各メンバーが情報交換できるように、FAQなどを用意したプロジェクトポータルを立ち上げることが有効です。プロジェクトポータルは、blogやwikiなどを利用してもかまいません。また、組織がNotesやサイボウズ

などのグループウェアを導入しているのであれば、それを活用することも可能です。

ちなみに筆者は、あるお客さまのところでこんな経験をしたことがあります。そこでは、wikiや既存のグループウェアを利用できないだけでなく、メールすら使えないような厳しい制限を受けながら開発を進めました。そのときに思い浮かんで実践したのが次のような方法です。まず、共有ディスク上に「疑問とその回答」など、共有したい情報をhtmlで日々書き加えていきます。そして、メンバー全員のWindowsのアクティブデスクトップを、そのhtmlを参照するように設定します。こうして、少なくとも朝にPCを立ち上げたとき、共有したい情報を目に見えるようにしたのです。このようになっとした工夫でも、メンバー間で疑問とその回答といった情報を共有できることは、非常に大きな効果があります。

Point04 成果物の位置づけの合意を得る

開発プロセスが進むと、要求仕様書や設計書、プログラム、操作マニュアルなど、さまざまな作業成果物が生み出されます。

これらの作業成果物の多くは、プロジェクトが終了すると、開発を依頼したユーザーの運用担当者や保守担当者へ引き継がれます。一方で、小規模なシステムをプロトタイピングしながら開発する場合は、要求仕様書や設計書はまったく書かず、プログラムと操作マニュアルだけがあればよいこともあります。あるいは、要求仕様書や設計書は書き、公式なレビューもするけれど、それらは一過性の作業成果物であり、最終的にユーザー側には引き渡さない(納品しない)ということもあります。

利害関係者の間で、このような作業成果物の位置づけを曖昧なままにしておくと、フェーズの終わりや納品の前後で、「形だけの作業成果物」を作ることに慌しく時間を費やすような状況が生まれます。こうなると、現場のエンジニアは「開発プロセスには無駄が多く、重すぎる」としか思えません。当然、彼らのモチベーションが向上するわけもなく、開発プロセスを実践するうえで大きな妨げとなります。プロジェクト計画を立てる際には、作業成果物を次のように分類し、利害関係者の合意を得ておきましょう。

●プロジェクトの最中に公式にレビューされ、終了時に引き渡される(納品される)もの

●プロジェクトの最中に公式にレビューされるが、一過性の作業成果物であり、開発プロセスが次のステップに進むと、メンテナンスされず、プロジェクト終了時に引き渡されないもの

●プロジェクトの最中に覚え書き的に作成され、公式なレビューも終了時の引き渡しもされないもの

Point05 公式な成果物はすべて構成管理する

開発プロセスが進む過程では、要求仕様書や設計書、プログラムなどの作業成果物が次々に生み出され、アップデートされていきます。開発プロセスを円滑に進めるには、これらのバージョンや相互の関係を管理することが重要です。すなわち、前項図8に示すような構成管理の仕組みと、それを実現するツールの導入が必要になります。

このとき、エンジニアから「どの作業成果物を構成管理の対象とすべきか?」という質問をよく受けます。その背景には「VSS(Visual SourceSafe)やCVS(Concurrent Versions System)などのバージョン管理ソフトで管理するのは、ソースコードだけでいいのか、設計書も含めるのか、いやいや要求仕様書も含めたほうがよいのでは?」といった疑問があります。この疑問に簡潔に答えるならば、「最低限、公式な作業成果物はすべて

管理する」となります。公式な作業成果物は、システムをリリースしたり納品したりするうえで、個々のバージョンやそれらの関係を維持し続けることが不可欠です。一方で、覚え書きのような非公式な作業成果物であれば、次の作業のインプットとして何らかのかたちで反映された時点で、それ以降アップデートされる保証はありません。したがって、それらを構成管理の対象とするかどうかは、バージョン管理ソフトを使う手間なども含めて考え、管理すべきだと真に思えるときだけ管理すればよいでしょう。たとえば、ホワイトボードに書いた覚え書きのクラス図のバージョンを管理するのは難しいことですが、モデリングツールで書いたものであれば、そのバージョンを管理することが、逆に書き手であるエンジニアにとって便利だったりします。

開発中～プロジェクト終了におけるポイント

最後に、システムの開発を始めてからプロジェクトが終了するまでの間で、押さえておくべきポイントを紹介します。

Point01 プロジェクトを駆動する

これまでと違うやり方(開発プロセス)で開発することを提案すると、必ずと言っていいほどメンバーに戸惑いが生まれるものです。そこで、従来のやり方からなるべく変わらないような開発プロセスにして、メンバーが経験を積むごとにそれを少しずつ変えていくというアプローチが考えられます。しかしこの方法は、これまでメンバーがそれぞれ自由気ままに、なんとなく存在する暗黙のやり方^{注11}で開発していた現場では、そう上手くはいきません。そのような現場でも、そうでない現場でも、プロジェクトで新たな開発プロセスを実践していくには、プロジェクトをグイグイ駆動する必要があります。プロジェクトマネジャーや開発プロセスの策定者がこの役割を担い、プロジェクトの次のフェーズや作業内容を示したり、戸惑っているメンバーを見つけて相談にのったり、ときには一緒に作業をしてあげるような手厚いフォローアップをしたりして、きめ細かく強い推進力でプロジェクトを駆動することが大切です。

Point02 プロセス全体の整合性に注意して改善する

「開発プロセスの評価」の部分で、「プロジェクトの中でエンジニアからのフィードバックを得て、開発プロセスを改善してい

注11 アジャイル系の開発プロセスでは、このような開発を「カウボーイ・コーディング」と呼ぶことがあります。

く」と述べました。この改善の際に注意すべきことがあります。

フィードバックで得られる情報の多くは、「この作業がやりにくいので、このように変更してほしい」といった内容になります。もちろん的確な改善案もありますが、そのまま反映すると、次に続く作業のインプットとして情報が不十分になったり、契約上納品しなければならないドキュメントとして不十分な内容になったりする要望もあります。また、提案された改善案よりも効果的な対策がある場合も少なくありません。たとえば、開発プロセスを変更するのではなく、要望を出したエンジニアにマンツーマンで説明するなどのフォローアップをしたほうが素早く、望ましいこともあります。出された改善案をそのまま取り入れるのではなく、まずは「なぜそのような要望が出たのか」と根本的な原因を分析する必要があります。そのうえで対策の選択肢を分析し、開発プロセスとしての整合性を保ちながら最良の対策を見つけ、改善していくことが重要です。

Point03 プロジェクトメトリクスを収集する

開発プロセスを継続して改善するには、その尺度となるプロジェクトの開発効率や品質といったメトリクス(これをプロジェクトメトリクスと呼びます)が必要です。そこで、ユースケースあたりの工数や、ユースケースあたりのバグ数などを収集していきましょう。詳細なメトリクスは、開発プロセスを改善するための分析には役立ちますが、収集にかかる現場の負担も考えてください。先ほど、メトリクスの例として「ユースケースあたりの工数やバグ数」を挙げたのは、この点を考慮したためです。ユースケースあたりの工数やバグ数といった程度の情報ならば、収集するのにそれほど負担はかかりません。これらのメトリクスは精度が多少粗くても、収集できるかどうかで、その後の改善度や、プロジェクトの責任者^{注12}の開発プロセスに対する理解に大きな影響を与えます。

Point04 最初から効果が上がることを期待しない

明文化された開発プロセスを実行し、プロジェクトメトリクスを収集できると、プロジェクトの責任者は「すぐに開発効率や品質が向上するだろう」と期待するものです。しかし、新たな開発プロセスを策定し、メンバーが学習しながら開発する場合は、短期的な効果が期待できないことがあります。以前のやり方とそれほど変わらない、場合によっては開発効率が低下することさえあります。これは、プロジェクト初期のメトリクスには、メンバーの学習する時間が含まれているからです。開発効率や品質などに開発プロセスの効果が現れるまでには、ある程度時間がか

かります。そのことをプロジェクトの責任者に理解してもらい、継続的にメトリクスを収集・報告し、改善していきましょう。

地道な実践を重ねよう

本稿では、開発プロセスを策定・実行するうえで必要となる基礎知識を説明しました。いかがでしたでしょうか。

机上であれこれ悩むよりも、まずは始めることが大切です。地道に実践を重ね、少しずつ良い開発プロセスに育てていくうちに、じんわりと効果が現れてくるでしょう。本稿がその一助となれば幸いです。🍀

注12 プロジェクトマネジャーの上司や、開発を依頼したユーザー。

参考文献

- 『ISO/IEC 12207:1995 Information technology-Software life cycle process』
- 『プロジェクト知識体系ガイド 第3版』Project Management Institute／著
- 『アジャイルソフトウェア開発スクラム』ケン・シュエイパー、マイク・ピードル著
- 『ラショナル統一プロセスガイドブック』パー・クロール、フィリップ・クルーシュテン著
- 『XP エクストリーム・プログラミング入門 第2版』ケント・ベック著
- 『アジャイルソフトウェア開発』アリスター・コーバーン著

PROFILE

正木 威寛(まさきたけひろ)
システムエンジニアやプロジェクトマネージャを経験し、現在は企業のシステム開発における調達プロセスや開発プロセスのコンサルティングに従事。ユーザー企業とSIerとの両方が幸せになれるような健全な開発プロジェクトを実現するために、日夜プロセス改善に取り組んでいる。与えられた制約の中で、最大限の知恵を絞り、良いアイデアを出すことが求められるプロセスコンサルティングの仕事は自分に合っていると自認している。

山内奈央子(やまうちなおこ)

金融系や人事系システムなど数々のシステム開発を上流から保守まで経験。ひよんなことから開発プロセス策定の仕事を経験し、その奥深さにハマること5年。人と企業文化とシステム開発というねりの中で、しなやかな開発プロセスを目指し、策定そして実践する日々。ちなみに最近ヨガに目覚め、心身のしなやかさにも磨きをかけている。