まえがき

# お手上げ

あなたはソフトウェアの会社に勤めており、現在とあるソフトウェア開発プロジェクトに参加しています。ところが、現在開発中のソフトウェアはバグだらけで、こちらを治せばあちらが壊れ、完全に制御不能です。

みんなでがんばってバグを治しているのですが、誰がどこを治しているのかはよくわかりません。おまけに、あなたが修正したはずのプログラムは、別の人が誤って上書きして消してしまうことがよくあります。動いていたはずの機能がいつの間にか動かなくなっていることもよくあります。ちゃんと前に進んでいるのか、後戻りしているのかも自信が持てない状態です。

バグはエクセルシートで管理していますが、やはりほかの人が同時にシートを変更するので、ファイルが上書きされて消えたり壊れたりします。シートには修正済みと書いてあるバグが、実際には治っていません。ほかの人が修正中のバグがどれで、自分が修正すべきバグはどれなのかも、把握できません。シートはいろんな人がコピーして作業するので、もはやどのシートが最新版なのかさえ、分からなくなってしまいました。

いったい、全部でどれだけのバグが残っているのでしょうか。もっとうまいやり方はないのでしょうか。あなたは答えを求めて書店をさまよい歩き、今、この本を手にしたところです。

# 本書のねらい

本書は、筆者が@ITというwebの媒体に2004年4月からおよそ1年間にわたって連載した「開発プロセス再入門」というタイトルの記事を大きく加筆・修正し、書籍としてまとめたものです。

元の記事は、ビルドの手順とバグの追跡という側面から、ソフトウェア開発のプロセス (手順) を説明したものでしたが、これに構成管理ツールとテストケース管理ツールの話題を加え、本書はソフトウェア開発に関するより包括的な入門書としました。

本書は、ソフトウェアを開発する手順を、ビルドのリリースとそれを支援する開発ツールという側面から、ある程度まで包括的に説明しました。ソフトウェア開発の入門書として、読者のみなさんに活用してもらうことが本書のねらいです。ただし、開発するソフトウェアの種類や規模、また開発のすすみ具合などによって、実用的で快適な手順は異なります。そこで本書では、基本的なソフトウェア開発の手順と合わせて、なぜそのような手順が必要となるのか、その考え方を併記するように努めました。この考え方を理解すれば、本書に示した手順の必要な部分だけをうまく取り入れて活用できます。

# 本書の主張について

本書が一貫して主張したいのは

**ソフトウェアは壊れやすい！**

ということです。ソフトウェアを開発するときは、まずこの大前提に立ち、もろいソフトウェアを壊わさないための配慮や工夫をあなたのチーム全員で実践する必要があります。どんなソフトウェアであっても、ほんの一部を書き換えるだけで全体が動かなくなることもあるのですから。これはとても当たり前なことですが、当たり前すぎて忘れている人も多いのではないでしょうか。読者のみなさんには、本書を読み、ソフトウェアのこの重要な性質を思い出してほしいと思います。

# 本書がソフトウェア開発の中心とする考え方について

ソフトウェア開発プロジェクトでは、複数のプログラムファイルを統合し、実行可能なソフトウェアを構築しテストすることを何度も繰り返します。この、実行可能な統合されたソフトウェアのことを「ビルド」といいます。ソフトウェア開発では、いろいろな作業を繰り返しながら進めていきますが、この一番基本的な作業の単位がビルドのリリースです。そこで、本書では、この「ビルド」を中心として、ソフトウェア開発の基本的な流れを説明します。各メンバーのタスクとして分割した作業が正しく完了したかどうかは、動作するビルドにより検証します。本書では、このようなアプローチを「ビルド・セントリック (ビルド中心) 」とよびます。

ソフトウェア開発では、ビルドが壊れないように、丁寧に育てていくこと、正しい方向に進化させていくことが非常に大切です。このため、本書は、ビルドがどのような性質や性格をもつのかなど、ビルドそれ自体を説明することにも多くのページを割いています。

# 本書の対象読者について

ソフトウェア開発プロジェクトに参加する人はすべて、ソフトウェアが非常にもろく壊れやすいことをよく理解し、開発中のソフトウェアをどのように扱うべきか知っていなければなりません。本書は、これを支援する開発ツールの活用方法を説明します。これらのツールは、ソフトウェア開発プロジェクトに属すメンバー全員が使う必要がありますから、ソフトウェア開発に従事するすべての人が、本書の対象読者となります。



図 1 本書の対象読者

しかし、ソフトウェア開発に従事する人すべてが対象読者といってしまっては、範囲が広すぎて、みなさんも本書の読み方がよく分からないと思います。そこで、本書の対象読者を4つのグループに分けました。

## 新米エンジニア (開発者、QA)[[1]](#footnote-1)

今年の春にソフトウェア開発企業に就職した方や、就職を考えている学生の方へ。本書は、新米エンジニアにとって一番役に立つように設計しました。ソフトウェア開発全体の概略を示した上で、みなさんが日々行うことになる作業の手順や、守るべき習慣を説明しました。また、本書には、もろいソフトウェアを壊さないようにするための工夫を集めました。本書に示した手順に従えば、とんちんかんなことをして周りに迷惑をかけることは少なくなるでしょう。

## 新米プロジェクトマネージャ

何年かソフトウェア開発を経験した後に、新米エンジニアを指導する、あるいはプロジェクトをリードする立場になった方へ。本書はソフトウェア開発のプロセスを改善するためのヒントを集めています。あなたのチームを統率する上で、有用な気づきを本書から得られるでしょう。あるいは、ある程度の経験をお持ちの方であっても、本書で紹介するソフトウェア開発の手順に驚かれる方もおられると思います。

## 新米ユーザー

これからソフトウェア製品 (システム) の開発を発注しようとするユーザー企業の方へ。本書は、いわゆる反復型開発プロセスというソフトウェアの開発手法の基本をわかりやすく解説します。発注段階で、システム会社から「反復型開発プロセスを使って開発します」とか、「マイルストーンごとに契約を更新しましょう」などと提案されたが、その意味やメリットがよくわからないという方は、本書をご一読ください。ソフトウェア開発に従事する人たちの関心事がどのようなものかを理解すれば、システム企業と、良いコミュニケーションがとれるようになります。また、良いソフトウェアを開発してもらうには、あなた自身がソフトウェア開発に上手にまきこまれることが必要です。その方法についても、本書からヒントを得られるでしょう。

## 新米セールス

自社のソフトウェア製品を営業する立場の方へ。あなたが販売する製品の、ソフトウェアとしての基本的な性質をもっとよく理解したい、ソフトウェア開発プロジェクトチームと共通の語彙を使ってより良いコミュニケーションがしたい、という方には、本書がその期待に応えると思います。ユーザーと開発チームの間に入り、新機能を盛り込んだり、製品をリリースしたりするタイミングについて、開発チームと折衝する必要がある方は、本書を読むことで開発チームの立場をより深く理解できるようになるでしょう。

このように、本書は対象読者を広く設定していますので、すべての人に対して直接的に役立つ記述とならない部分があります。しかし、本書はソフトウェア開発の手順を、ある程度包括的に説明することを指向しましたので、この点はご容赦下さい。その代わり (といっては何ですが)、あなた以外の人が、どのような立場でソフトウェア開発という仕事に関わっているのか、本書を読むことで少し見えてくるかもしれません。

# 本書が説明するソフトウェア開発の段階について

図 2を見てください。これは、典型的なソフトウェア開発プロジェクトが踏む段階です。本書で解説する事柄に対応する範囲に色をつけました。



図 2 ソフトウェア開発の段階

ただし、本書ではソフトウェアの設計技法、実装技法、テスト技法のどれについても説明しません。本書の関心事は、そのような技術ではなく、各段階で必要になる作業の手順についてです。また、図 2に示した段階は、実際にはプロジェクトを通して何度も繰り返す必要があります。本書では、その方法についても説明します。

# 本書が紹介するツールについて

本書では、次の4つのツールを説明します。

* ソフトウェア構成管理ツール (3章)
  + CVS, Subversion, Gitなど
* ビルドツール (4章)
  + Make, Ant, NAnt, MSBuild, Hudsonなど
* バグ追跡システム (5章)
  + Bugzilla, Trac, Jira, Redmineなど
* テストケース管理ツール (6章)
  + TestLink, Microsoft Test Managerなど

これらのツールは、ソフトウェア開発において重要な役割を果たします。しかし、これらのツールを使う上で難しいのは、1人で使うものではないことです。ソフトウェア開発プロジェクトに属するメンバー全員が、それぞれの立場から、お互いと必要なコミュニケーションを取りながら、各ツールを使う必要があります。また、各ツールをうまく連携させることも必要です。そのようなノウハウは、個別のツールによって大きく異なるものではありません。図 3は、複数の構成管理ツールの関係を表したものです。



図 3 本書で説明する部分 (構成管理ツールについて)

構成管理ツールには、この図に挙げたもの以外にもMercurialやPerforceなど、さまざまなものがあります。しかし、それらの使い方には共通の概念や戦略的な使い方があります。ビルドツールやバグ追跡システムについても同様です。そこで本書では、各ツールを個別に取り上げて説明するのではなく、どのツールにも一般的に適用できる運用方針と活用方法を説明しました。

各ツールについては、それぞれに良い専門書が多く出ています。しかし、それらを読む前に本書に目を通しておけば、なぜそれらのツールを使う必要があるのか、またなぜ各ツールにそのような機能が装備されているのかといった本質的な部分を理解し、それぞれの専門書をかみ砕く準備をすることができます。



図 4 各ツールの専門書と本書との関係

# 本書が説明しないこと

前述のように、各ツールに個別の使い方やコマンドのほか、ユーザーの要求を獲得して定義する方法や、ソフトウェアの開発工数を見積もる方法は本書では説明しません。それは、これらを単純な手順として定義することがとても難しいからです。開発するソフトウェアのドメインによっても、それらは大きく異なります。これらを一般に、シンプルに支援するツールが存在しないことも、本書では取り上げない理由のひとつです。ただし、本書は決して「仕様書を記述する必要はない」などと主張するものではありません。仕様書の記述については、6章で簡単に扱います。

このほか、優れたアーキテクチャの獲得方法や、ソフトウェアの設計方法、きれいなプログラムの書き方、具体的なテスト技法などについても説明しません。これらは職人技に類するものであり、本書が説明するソフトウェアの開発手順とは直交するものだからです。もちろん、ソフトウェアの開発には職人技も必要ですが、それらについては他の記事や書籍を参照して下さい[[2]](#footnote-2)。

# 本書の構成

## 1章 ソフトウェアの進化と寿命

あるソフトウェア開発プロジェクトにおいて、開発中のソフトウェアがどのように発展し進化するのか (させるべきか) を説明します。そして、その進化を支援する道具を列挙し、それらを活用する際に意識すべき人の立場 (役割分担) を紹介します。

## 2章 ビルドの多様性

ビルドとは、実行可能なソフトウェアのこと、あるいはそれを構築する作業のことです。ソフトウェアを開発するときは、ビルドが壊れないように細心の注意を払いながら作業をすすめます。このため、ソフトウェア開発に従事するメンバーには、ビルドについて深く理解することが求められます。本章では、ビルドの管理方法や、ビルドのリリース計画の立案方法を説明します。また、ビルドにはどのような種類のものがあるのかを紹介し、ソフトウェア開発の段階がすすむにつれ、リリースすべきビルドの位置づけがどのように変化していくかを説明します。

## 3章 構成管理とサンドボックス

構成管理ツールを使って、ソフトウェアを構成するプログラムファイルの変更履歴を管理する方法を説明します。サンドボックスとは、開発者のための開発環境のことです。本章には、開発者の非常に日常的な作業手順が書かれています。また、ソフトウェア製品を上手に枝分かれさせて進化させるためのガイドラインも示します。本章は、ソフトウェア製品のリリースを計画し管理する人には有用なものとなるでしょう。

## 4章 再現可能なビルド

ビルドとは、実行可能なソフトウェアのこと、あるいはそれを構築する作業のことです。本章は、この手順と、それをビルドツールで自動化する方法を説明します。一般に、テストチームにリリースするビルドは1日に1回、もしくは週に1回の頻度で作ります。ビルドはいわゆる反復型開発における最も小さな反復の単位を構成するものです。また、ビルドの手順は自動化できる範囲が広いため、ソフトウェア開発に必要なコストを下げる要諦のひとつです。

## 5章 バグの追跡

ソフトウェア開発の下流工程では、バグを追跡するプロセスが非常に重要です。バグを発見して文書化し、その解決を実行可能なビルドでテストして検証するという手順で作業を進めます。このワークフローを自動化するツールがバグ追跡システムです。本章では、ソフトウェア開発においてバグをどのように扱うべきかを説明します。

## 6章 テストケースの実装

ソフトウェア開発とは、仕様を定義し、これを実行可能なものへ変換するプロセスです。誤解をおそれずにいえば、ビルドとは詳細化された実行可能な仕様そのものであるといえます。しかし、ビルドのほかにも重要な実行可能ドキュメントがあります。それはテストケースです。本章では、テストケースの性質を説明し、その作成手順や運用方法についてヒントを与えます。

## 7章 開発プロセスの構築

本章では、ウォーターフォール型のソフトウェア開発モデルを紹介し、これと対比させることで、反復型開発のメリットを明らかにします。また、ビルドという単位よりも大きな単位の反復を構成する方法を示します。また、いわゆるアジャイルといわれる開発手法を、本書で説明したインフラの上にのせて取り入れる方法を紹介します。

このほか、ソフトウェア開発に関わるさまざまなトピックや、開発プロセスを改善するためのヒントをコラムとして各章に挿入しました。みなさんが、本書を気に入っていただければ幸いです。

1. QAとは、ソフトウェアのテストを担当する責務を担う人で、テスターともいいます。このほか、ソフトウェア開発プロジェクトに必要な役割と責務は、第1章で説明します。 [↑](#footnote-ref-1)
2. ソフトウェア開発において特に重要な職人技のいくつかは、本書でもその入り口を適宜コラムで紹介しました。 [↑](#footnote-ref-2)