まえがき

**状況の例 ... 青字の部分**

**この本が取り上げるテーマに沿ったものにもっとした方が良い**

# 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、スケジュールは遅れ、一体どうしたらいいのか皆目見当もつきません。

このプロジェクトで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しまい、消されてしまいます。インターフェイスの変更が多く、おまけにチーム間の連絡が悪いためにいつまで経っても結合テスト[[1]](#footnote-1)を開始できません。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。

**表現直した方がいい**

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

**ファイル名を見ると、扱っているのは構成管理領域が主のように見えます。**

**もっと範囲を絞った書き方にした方が良いのでは？**

本書は、ソフトウェア開発のすすめ方の基本的なガイドラインと、その手順を具体的に説明します。ソフトウェア開発手法の入門書として、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、手順というのはとても変わりやすいものです。どんなソフトウェアを作るのか、どんな規模のものを何人で開発するのか、現在の開発段階は初期か末期か、などによって、最適な手順は異なります。すべてのソフトウェア開発に適用できる開発モデルはないのです。そこで本書では、手順については基本的なものだけをとりあげました。そして、なぜそのような手順が必要となるのか、基本的な考え方をガイドラインとしてまとめました。また、この考え方は普遍的かつ不変的なものですから、あなたがどのようなソフトウェアを開発しようとも、このまま活用できるはずです。この、ソフトウェア開発に基本的な手順を一度身に付けてしまえば、その知識は古くなることはありません。あなたがその人生をソフトウェア開発に捧げるつもりなら、本書はあなたの人生を助け、豊かにするでしょう。

**ガイドラインとか開発モデルとかいろいろな言葉がでてきていますが、どっかで定義されているの？**

**まえがきなので、もっと平易で意味が明らかな用語を使った方が良いと思いますよ。**

# 本書の主張について

**思いはわかるが前後のつながり、全体の内容にどう関係するのかはっきりしない。唐突感がある。**

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。どんなに堅牢なソフトウェアであっても、ほんの一部を書き換えることでシステム全体が動かなくなるようにできるのですから。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

**S/W業界で言われている「堅牢」とは違った意味でつかわれているようです。違う言葉にした方がいいのでは？**

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、この繰り返しの一番基本的な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。ソフトウェアの開発計画とはビルドのリリースを計画すること、ソフトウェアの品質管理とは各ビルドの健康を管理することです。また、各メンバーのタスクとして分割した (細かい) 作業が正しく完了したかどうかは、動作するビルドを使って検証します。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。

**個人的にはちょっと感性的すぎる感がします**

**後半...ちょっと言いすぎ**

**前半...誤解される**

**どこを指しているかいまいち明確ではない**

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

**はじめの最初の事例ともっと整合させた方がいい**

**＆記述整理した方がいい**

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ておかねばなりません。もし、そうでない人がいて、うかつにも開発中のソフトウェアに甚大なダメージを与えることにでもなれば、その復旧作業のための手間 (コスト) がかかってしまうため、チームのみんなが迷惑します。本書は、開発中のソフトウェアが壊れないようにするためのガイドラインを説明しますので、ソフトウェア開発に従事する人のすべてが、本書の対象読者となるでしょう。本書で述べているのは、ソフトウェア開発をすすめる上で非常に基本的かつ常識的なことですから、特に新米の方には本書が役に立つはずです。

**ちょっとひっかかります**

**前の通りだとすると「特に」ではなく、新米の方にしか有用でなくなってしまうのでは？**

**でもそうじゃないんですよね。**

**表現直した方がいい**

**表現いまいち**



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新米の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。

**QAの定義に関し、現状確かにそういう会社が多いですが、本来のQAではない。別の用語を使った方が良いと思いますよ。**

## 新米エンジニア (開発者、QA)[[2]](#footnote-2)

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、守るべきプラクティスを詳細に説明しています。本書を読めば、ソフトウェア開発の現場がどのような手順で作事を進めるのか、具体的なイメージをつかめるはずです。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常識や素養を多く盛り込みました。本書にあるプラクティスに従えば、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。

**いろいろなところでいろいろなふうに述べている → 冗長感あり、結局何をどういったレベルで書いてあるのかはっきりしない**

**??**

**用語**

## 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。みなさんには、本書が説明する事柄は当たり前に過ぎ、かつ簡単に過ぎるかもしれません。しかし、当たり前だからこそ、逆に忘れてしまっていることも多いはずです。また、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幅広く集めていますから、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はずです。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基盤に新鮮な驚きを感じる方もおられるかもしれません。

**「はずです」はおかしい**

**論理的じゃない。**

**常識であれば、忘れない**

## 新米ユーザー

**以下、これまでと同種のコメントは省略**

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。

**（）の意味がよく分からない**

**一般的には別職種**

## 新米マーケッタ

自社のソフトウェア製品をマーケティング (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、自社の開発工程におけるしっかりした品質管理手法をユーザーにアピールしたい、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。また、ソフトウェア製品のリリース計画は、その開発計画とは切っても切り離せません。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。

**前の記述と一貫していない**

**取り上げているの？**

**とにかく記述を整理した方がいい**

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進め方を、ビルドのリリースという側面から、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。

**最初にこれ書いた方がいいのでは？**

# 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。

**ボックス1つ、単語2であれば、ここは「要求獲得・要件定義」の方がいい**



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、そのような職人の技術ではなく、各段階で必要になる作業手順や、開発プロジェクトチーム内のメンバーがどのようなやりとりをすべきかについてです。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。

**エンジニアと“職人”は違う**

# 本書が紹介するツールについて

最近は、ソフトウェアを開発するときに重要なのは人と人とのコミュニケーションである、とする記事や文章をよく見ます。ユーザーと開発者とのやりとり、開発者同士のやりとりなど、確かにコミュニケーションは重要です。しかし、単にコミュニケーションを密にしようと努力するだけではうまくいきません。当然ですが、ソフトウェアは壊れやすいという前提に立ち、必要なコミュニケーションを整然ととらなければ、ソフトウェア開発プロジェクトは正しい方向にすすみません。つまり、間違った努力をするよりも、正しいコミュニケーションを促進する**仕組み**を組織に実装することがより合理的でスマートなアプローチです。ありがたいことに、ソフトウェア開発を支援するための仕組み (ツール) がすでに多く開発されています。本書は、人と人のコミュニケーションと同じく、そのようなツールを重視します。それらのツールをうまく使えるようになれば、必要なコミュニケーションも正しく取れるようになるはずだからです。また、それらのツールは、ソフトウェアの壊れやすいという性質にうまく対処できるようになっています。本書では、そのようなツールとして以下の4つを説明します。

**間違った努力なんだから、これは自明→文章おかしい**

**何を指しているのか？？**

* 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
* 課題追跡システム (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、それがなければソフトウェアを開発することはできない、というくらいに大切なものです。これらは、まさしくソフトウェア開発を支えるユーティリティ[[3]](#footnote-3)となっているのです。本書では、これらのツールをユーティリティとして活用するためのプラクティスを示します。

ただし、本書では各ツールを個別にとりあげて使い方を説明することはしません。すべてのツールに共通の概念や、共通の使い方を解説します。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)

この図に挙げたもの以外にも、MercurialやPerforceなど、さまざまな構成管理ツールが利用可能です。しかし、本書では各ツールを個別に取り上げて使い方を説明することはしません。この図に示すように、構成管理ツールには、共通する概念や戦略的な使い方があります。本書では、それを説明します。

各ツールには、それぞれの専門書が多く出版されていますので、詳細な使い方を知りたい方はそちらを読むことを勧めます。しかし、それらは各ツールの使い方に特化されているために、開発対象のソフトウェアをどう扱うべきかという視点が欠けているものもあります。それらの専門書を読む前に本書に目を通しておけば、個別のツールについてもずっと理解が早く進むはずです。ビルドツールや課題追跡システムについても同様です。



図 4 各ツールの専門書と本書との関係

もし、みなさんの組織がこれらのツールをまだ導入していないなら、まず各ツールにはどのようなものが利用可能かを調べ、それぞれを評価し、どれがみなさんの組織に一番適しているかを検討して下さい。各章の冒頭に、利用できる代表的なツールの名前を挙げておきました。それぞれについて、多くの専門書が出版されており、Web上の情報も豊富です。無償で利用できるものがほとんどですので、実際にみなさんのPCにインストールして評価できます。

# 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[[4]](#footnote-4)。

**少なくとも要求と要件/仕様は区別した方がいい**

# 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。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。

**サンドボックスについても触れた方が良い**

## 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、ビルドは日次もしくは週次の単位で実施します。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。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。

**言いすぎ**

## 7章 開発プロセスの構築

現在、流布されているウォーターフォール型モデルに対する誤解を解きあかし、改めて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. 複数の人やチームによる成果物を結合 (統合) して実施するテストのことです。 [↑](#footnote-ref-1)
2. 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-2)
3. 一般にユーティリティとは、電気やガス、水道などのインフラのことです。 [↑](#footnote-ref-3)
4. 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-4)