まえがき

お手上げ

本書のねらい

本書の主張について

本書がソフトウェア開発の中心とする考え方について

本書の対象読者について

本書が説明するソフトウェア開発の段階について

本書が紹介するツールについて

本書が説明しないこと

本書の構成

1章 ソフトウェアの進化と寿命

1-1. ソフトウェアの進化と寿命

1-2. ソフトウェアの進化の枝分かれ

1-3. 種としてのソフトウェアの断絶

1-4. ソフトウェアを育てる

1-5. ビルド中心の生活

コラム ソフトウェア開発のメタファ① 旅行

1-6. 本書で説明するツール

1-7. 開発プロジェクトに必要な役割 (ロール)

コラム 開発中の催し① キックオフミーティング

1-8. まとめ

参考文献

2章 ビルドの多様性

2-1. ビルドとは

2-2. ビルドの位置づけとリリースの順序

2-3. ビルドをリリースする範囲と、反復の階層

コラム 開発中の催し② デイリースタンドアップミーティング

2-4. なぜビルドを定期的にリリースするのか

2-5. ビルド曜日を決める

2-6. ビルドの世代交代

2-7. ビルドに番号をつける

2-8. バージョン番号とは

コラム TeXのバージョン番号

2-9. バージョン情報の書式

2-10. ビルドのリリースを計画する

2-11. ビルドで作業の進捗を測る

コラム ソフトウェア開発のメタファ② 建築

2-12. ビルドのライフサイクル

2-13. ビルドの種類

コラム C/C++言語のプリプロセッサとデバッグビルド

コラム ドッグフードとは

コラム 動詞としてのビルドの種別

2-14. リリースノートの添付

2-15. ビルドの配置 (インストール)

2-16. まとめ

参考文献

3章 構成管理とサンドボックス

3-1. ソフトウェア構成管理とは

3-2. リポジトリ

3-3. サンドボックス

3-4. リポジトリにファイルをコミットする

コラム プログラムファイルは会社の資産

3-5. 清潔なサンドボックス

3-6. サンドボックスの構築手順の確立と文書化

3-7. 複数のサンドボックスを切り替えて作業する

コラム ソフトウェア開発のメタファ③ 音楽

3-8. ビルドマシン

3-9. リポジトリで管理すべきファイル

コラム プログラムとは

3-10. コミットするときによくある失敗

3-11. コミットの手順

コラム アトミックなコミットとは

3-12. ビルドブレークに対処する手順

3-13. リポジトリの汚染に備える

3-14. バグが混入したタイミングを調べる

3-15. タグを付ける

3-16. ブランチを切る

3-17. ブランチとビルド番号

3-18. ブランチ間で差分をやりとりする

3-18. バックポートとフォワードポート

3-19. マージの追跡

3-20. 誰がブランチを切る？

コラム 開発中の催し③ ブラウンバッグミーティング

3-21. ブランチを切りたいとき

3-22. メインラインモデルとプロモーションモデル

3-23. ブランチを上手に使うためのプラクティス

3-24. ポートを忘れないようにする

3-25. リポジトリへのアクセスを制御する

3-26. SCM サーバの管理

3-27. まとめ

3-28. 参考文献

4章 再現可能なビルド

[4-1. ビルドの概要](#_Toc277450905)

[4-2. ファイル間の依存関係](#_Toc277450906)

[4-3. 再現可能なビルドの実現](#_Toc277450907)

[4-4. 安全なインクリメンタルビルドの実現](#_Toc277450908)

[4-5. ビルドブレーク](#_Toc277450909)

[コラム ソフトウェア開発のメタファ③ 自動車工場](#_Toc277450910)

[4-6. MakeとAnt](#_Toc277450911)

[4-7. IDEとビルドツールの分離](#_Toc277450912)

[4-8. 誰がビルドマスターを務める？](#_Toc277450913)

[コラム 開発中の催し④ ワンオンワンミーティング](#_Toc277450914)

[4-9. ビルド手順の概要](#_Toc277450915)

[4-10. ビルドスクリプト (build.xml)](#_Toc277450916)

[4-11. Antの独自タスクを書く](#_Toc277450917)

[4-12. ビルド手順の詳細](#_Toc277450918)

[4-13. ビルド手順書の記述](#_Toc277450919)

[4-14. リリース列車を定刻通りに発車させる](#_Toc277450920)

[4-15. 適切なビルドの頻度を探す](#_Toc277450921)

[4-16. 継続的インテグレーション](#_Toc277450922)

[4-17. 継続的インテグレーションとビルド時間](#_Toc277450923)

[4-18. 継続的インテグレーションとビルドサーバ](#_Toc277450924)

[4-19. 継続的インテグレーションのまとめ](#_Toc277450925)

[4-20. まとめ](#_Toc277450926)

[4-21. 参考文献](#_Toc277450927)

5章 バグの追跡

5-1. バグの追跡

5-2. バグ追跡システム

5-3. バグとは

コラム ソフトウェアの不具合をバグというのはなぜ？

5-4. バグの発見と文書化

5-5. バグ報告票のライフサイクル

コラム ソフトウェア開発のメタファ⑤ 医療

5-6. バグの処理方法 (Resolution)

5-7. バグ報告票に記載すべき項目

コラム 絶対基準の項目には、その基準を表示すべし

5-8. バグ報告票によるフィードバック

5-9. バグ追跡のアンチ・プラクティス

5-10. リグレッション

5-11. リグレッションテスト

5-12. トリアージ

5-13. トリアージとプリオリタイズの違い

5-14. トリアージ基準

5-15. トリアージ基準の変化

5-16. トリアージのまとめ

5-17. バグ収束曲線

5-18. ゼロ・バグ・バウンス

5-19. バーンダウンチャート

コラム 開発中の催し⑤ バグバッシュ

5-20. BTSをタスクリストとして使う

5-21. BTSをハードウェア管理票として使う

5-22. BTSの二重化

5-23. トヨタのかんばん方式と課題追跡システム

5-24. まとめ

5-25. 参考文献

6章 テストケースの実装

6-1. なぜテストは難しいか

6-2. ソフトウェアのテストはサンプリング

6-3. テストケースの管理

6-4. テストケース管理システム

6-5. テストケースとは

コラム ソフトウェア開発中の催し⑥ ポストモータムミーティング

6-6. テストケースを実装する手順

6-7. 1ページ仕様書の記述

6-8. 機能設計仕様書の記述

6-9. テスト設計仕様書の記述

6-10. テストケースの記述

コラム ソフトウェア開発のメタファ⑥ 車の運転

6-11. オートメーションコードの記述

6-12. テストハーネス

6-13. 単体テストと統合テスト

6-14. テストケースの追加とバグ報告票

6-15. まとめ

参考文献

7章 ウォーターフォールの起源

7-1. ウォーターフォール型の開発モデル

7-2. ウォーターフォール型モデルの問題点

コラム 仕様と実装を近づける試み

7-3. 反復型の開発プロセス

7-4. 反復の構築

7-5. 反復型開発のメリット

コラム ソフトウェア開発のメタファ⑦ お料理

7-6. アジャイルな開発プロセス

コラム ソフトウェア開発で使われる用語

7-7. 次のステップへ

7-8. まとめ

7-9. 参考文献