0.0.1. プログラムのテストと統合

0.0.2. 周辺装置やネットワークの利用

- ハードウェアを使用したプログラム
  - バーコードリーダー・IC タグ・スキャナ・カメラ
- ネットワークを利用したプログラム
  - Web サーバー・クラウド
- データベースを利用したプログラム
  - DBMS・SQL 言語

<API の活用> プログラム同士が機能や管理をするデータなどを相互にやり取りするために使われる「 **WebAPI** 」
- Web サーバー上の URL にアクセス -> 処理結果
- 事前の利用登録が必要なものもある（ **API キー**）

0.1. プログラムの統合

0.1.1. 結合テスト

- トップダウンテスト
  - 上位のモジュールから順に結合していく（スタブ要）
- ボトムアップテスト
  - 会のモジュールから順に結合していく（ドライバ要）
- サンドイッチテスト
  - トップダウンの両方から行う（時間短縮）
- インターフェーステスト
  - 個々のモジュールが正しく連携するかのテストをする
- シナリオテスト
  - システムで使われるシナリオをもとにテストをする
- 負荷テスト
  - 処理能力の有無をテストする
- 総合テスト
  - 要求定義所、外部設計書の内容が実現できているかどうかを確認する

1. 情報システムの開発管理と運用・保守

1.1. 情報システムの開発工程の管理

1.1.1. 開発手法の種類と選択

大型汎用機→クライアントサーバーシステム、パソコン ダウンサイジング、高機能、開発スピードが求められる

<開発手法> プロトタイプ型、スパイラル型、アジャイル型、ウォーターフォール型

開発手法	メリット	デメリット
プロトタイプ型	仕事が明確でなくても開始ができる、手戻りが起こりにくい	利用者の要望が増加しやすい、開発全体の納期や費用を見積もりにくい
スパイラル型	仕様変更に柔軟に対応できる、計画変更しやすい	開発初期に全体を把握しづらい、開発の終息に時間がかかることがある
アジャイル型	仕様が明確でなくても開始ができる、要望変更、市場や環境の変化にも対応できる	従来型の開発のほうが適している場合がある、利用者の協力が得られない場合は効果が出づらい
ウォーターフォール型	作業内容が明確であるため計画が立てやすい、開発事例が多く参考にしやすい	開発終盤まで実際に動くシステムを確認できない、不具合対応や仕様変更の影響が大きい

1.1.2. 開発工程の管理手法

プロジェクトチームでの開発 ※プロジェクトリーダーによる工程管理が必要

1. 開発工程名
2. 作業名
3. 担当者名
4. 作業予定期間
5. 作業実績期間

1.2. チームにおける開発手法

1.2.1. ソースコードレビュー

・作成担当者以外のメンバーがソースコードをレビューする
・プログラミング言語の知識 + ミスを生まない記述方法の知識

チェック内容 ：コーディング規約に沿って書いてあるかどうか ：正しいロジックで書いてあるかどうか

1.2.2. コーディング規約

プログラミング言語ごとに作る（記述ルールが異なるため）

・プログラムの冒頭に着けるコメント
・変数名のつけ方や宣言方法
・プログラムロジックの記述
・変数の型 etc...

1.2.3. プログラム開発の履歴管理

チームでの開発 -> 生活部の一元管理、共有が必要

分散型バージョン管理システム

1.3. 情報システムの運用と保守

1.3.1. 運用と保守の違い

運用：通常業務、利用可能な状態を維持する

保守：業務内容や環境の変化に対応した機能追加、トラブルや機能低下発生時、修理や復旧などの障害対策

契約不適合担保責任
...開発者が契約通りの機能を提供できなかった場合、修正や再開発を行う責任。

1.3.2. 担当部門の役割

・業務手順書の作成と更新：ユーザーの操作手順書（マニュアル）ではなく、システムの操作方法
・利用者教育
・システム監視：状態を監視し、以上を検知することでトラブルを未然に防ぐ
・定例報告と改善案の提案：内容はすべて記録し、ノウハウを蓄積（5W3H）

SLA

…サービスレベル契約（Service Level Agreement）の略。 サービス提供者と利用者との間で、サービスの品質や提供内容について合意した契約。