調査５ モデルカリキュラムの提言 コースウェア

１７．開発ツールに関するスキル

Ⅰ．概要	バージョン管理システム、デバッガ、バグ追跡システム、システムプロファイラ、カーネルデバッガなどのソフトウェア開発ツールについて、その種類と特徴、動向の理解に必要な一連の基礎知識と活用技術の概要を学ぶ。
Ⅱ．対象専門分野	職種共通
Ⅲ．受講対象者、 受講前提	入門カリキュラムのため、特に規定しない。 基本的なコンピュータ科学（ITSS レベル 1 程度）を習得、経験しているレベルの知識を有すること。
Ⅳ．学習目標	• 開発ツールを構成するソフトウェア、ハードウェア、ツールの概要と開発やデバッグの進め方を学ぶ。 • 開発ツールの主な機能と特徴、使用要件と開発やデバッグの進め方を学ぶ。
Ⅴ．使用教科書、 教材等	『Inside Linux Software オープンソースソフトウェアのからくりとしくみ』 佐藤竜一著、翔泳社刊 『オープンソース徹底活用 Eclipse による Java アプリケーション開発』 水島和憲著、秀和システム刊 上記に加えて、オリジナル教材を作成するものとする。
Ⅵ．習得スキル の評価方法	講義終了後の受講レポート、定量アンケート、知識確認ミニテスト、 演習問題の取り組み状況を総合的に判断して評価を行う。
Ⅶ．カリキュラム の構成	レベル 1　第 1 回～第 7 回 レベル 2　第 8 回～第 15 回

講座内容

第 1 回　開発の流れとツール（講義　90 分）

ソフトウェア開発環境を構成するソフトウェア、ハードウェア、ツールの概要と開発やデバッグの進め方を学ぶ。

（1）ソフトウェア開発プロセスの特徴と開発環境

1. ソフトウェア開発の特徴
2. OS
   - オープンソース OS である Linux のソフトウェア開発上の特徴（長所、短所）
   - 開発のチェックポイント
3. Linux における開発環境
   - コンパイル環境構築

（2）ソフトウェア実装

1. ブートローダ
2. ファイルシステム
3. 動作確認

（3）アプリケーション開発

1. ソフトウェアのリソース
2. デバッグ手法

第 2 回　ソフトウェア開発環境の概要（講義　90 分）

ソフトウェア開発環境の全体像とその構成、必要性を理解する。開発環境を用いた開発方法の特徴と役割を理解する。

(1)ソフトウェア開発環境の必要要素

1. 言語
   - ルールチェッカ
   - コンパイラ／クロスコンパイラ
2. プログラミング支援
   - 実行トレース
   - 変数シミュレーション
   - デバッグポイント
   - アプリケーションシミュレータ
3. システムプロファイラ
   - ハードウェアの情報調査
   - インストール済みのソフトウェアなどの情報を一覧
   - シリアルナンバーからの検索
   - ハードウェアキーの検出

(2)統合開発環境の構築

1. プラットフォーム管理
2. ソースコードバージョン管理
3. アプリケーションプログラミングインタフェース管理
4. ドキュメント管理

第 3 回　Linux 開発環境におけるソフトウェアアプリケーション開発の概要

（講義＋ワークショップ　90 分）

Linux 開発環境におけるソフトウェアアプリケーション開発の作業内容、手順を実際に開発を行い、理解する。サンプルを用いてプログラムを作成しながら、実習形式で開発ツールの特徴と役割を理解する。

(1)プログラムのコンパイルとリンク

1. C 言語によるプログラミングとコンパイル
   - C プログラムの基本構造
   - プログラムのコンパイルと実行
   - ライブラリの利用
2. gcc によるコンパイル
   - コンパイルにおいて行われる処理
   - プリプロセス
   - コンパイルと最適化
   - アセンブリ
   - リンク
3. オブジェクトファイルの構造
   - プログラムはどのように実行されるか
   - デバッグ情報の付与
   - シンボルテーブルの除去
4. ライブラリ
   - 静的ライブラリ
   - 共有ライブラリ

第 4 回　バージョン管理ツールの活用(講義　90 分)

バージョン管理ツールの機能と特徴、メリット/デメリット、バージョン管理ツールの開発手順を理解する。

(1)バージョン管理ツールの機能

1. バージョンの管理
2. バージョンの戻し
3. ブランチの管理

(2)主なオープンソースバージョン管理ツール

1. CVS
   - 機能
   - 特徴
   - ライセンス
2. VSS
   - 機能
   - 特徴
   - ライセンス
3. Subversion
   - 機能
   - 特徴
   - ライセンス

第 5 回　デバッガによるプログラムデバッグの環境(講義　90 分)

ソフトウェア開発環境の全体像とその構成、必要性を理解する。開発環境を用いた開発方法の特徴と役割を理解する。

(1)基本的なデバッグの方法

(2)トレース機能

1. トレースの基本機能
   - 現象のシミュレート
   - トレースメモリの評価
   - データトレース可能
2. トレース機能の拡張
   - ブレークポイントの設定
   - トリガ設定可能(ポイント、エリア、シーケンシャル、カウントなど)
   - タイムスタンプ機能

(3)複雑に絡み合うアプリケーションとミドルウェア間のデバッグ

(4)データベース処理に関連するデバッグ

第 6 回　カーネルデバッガを使用したデバッグ（講義　90 分）

カーネルデバッガを使用したデバッグ環境の全体像とその構成、必要性を理解する。開発環境を用いた開発方法の特徴と役割を理解する。

(1)カーネルデバッガによるカーネルやアプリケーションのデバッグ

1. カーネル機能ごとのデバッグポイント
   - リアルタイム制御
   - マルチタスク
   - デバイス管理
   - タイミング制御
2. カーネルデバッガの機能
   - ハードウェアエミュレーション機能
   - マルチプロセス環境実現
   - プロセス状態管理機能

(2)ハードウェアエミュレータの必要性

1. 各プロセスの状態遷移による OS 動作に関する検証
2. プロセスの状態遷移解析
3. プロセスのパターン分析
4. 異常処理時間のプロセス検索
5. OS アナライザ連動

第 7 回　バグ追跡システムを使用したデバッグ（講義＋ワークショップ　90 分）

バグ追跡システムを使用したデバッグ環境の全体像とその構成、必要性を理解する。代表的なオープンソースのバグ追跡システム「bugzilla」を用いたデバッグ方法の特徴と役割を理解する。

（1）欠陥追跡システムの目的

1. 継続的な製品の品質向上
2. 欠陥情報の効果的な収集と修正作業への反映
3. バグ対応の生産性向上

（2）欠陥追跡システムの機能

1. 未解決のバグの追跡
2. バグの間の依存性と依存性の図示
3. 報告機能
   - 電子メール
   - XML
   - コンソール
   - HTTP API
4. バグ記録累積、トレースのための RDBMS
5. バグ解決プロトコル

（3）Bugzilla とは

1. Perforce、CVS など自動化されたソフトウェア構成管理システム
2. Bugzilla 電子メールインタフェースとチェックイン/チェックアウトスクリプト
3. queryhelp.cgi
4. Bugzilla の使用例
   - IT サポートキュー
   - システム管理デプロイメント管理
   - チップデザインと開発問題追跡
   - ソフトウェア/ハードウェアのバグ追跡

第 8 回　オープンソース開発ツールの種類と機能(講義＋ワークショップ　90 分)

オープンソースの開発ツールの種類とそれぞれの機能、使用方法などを理解する。サンプルを用いて実習形式で開発ツールの特徴と役割を理解する。

(1)開発ツールの種類と特徴

1. 分析･設計ツール
2. プログラミング支援ツール
   - ソース解析ツール
   - Java コーディング規約チェックツール
   - Java コードの静的解析ツール
3. ビルド･開発リソース管理ツール
4. カバレージテスト支援ツール
   - Java の単体テスト
   - カバレージ計測
5. 単体テスト･ツール分析/設計ツール
6. プロジェクト管理

(2)アプリケーション構築環境

1. J2EE 実行環境
2. EC 構築環境

(3)オープンソース開発ツールの特徴

1. Jude
   - 機能
   - 特徴
   - ライセンス
2. Checkstyle
   - 機能
   - 特徴
   - ライセンス
3. Jlint
   - 機能
   - 特徴
   - ライセンス
4. Maven
   - 機能
   - 特徴
   - ライセンス

5. jcoverage
   - 機能
   - 特徴
   - ライセンス
6. FindBugs
   - 機能
   - 特徴
   - ライセンス
7. JUnit／djUnit
   - 機能
   - 特徴
   - ライセンス
8. JBoss
   - 機能
   - 特徴
   - ライセンス

第 9 回　統合開発環境を用いた開発手順（講義　90 分）

統合開発環境の全体像とその構成、必要性を理解する。開発環境を用いた開発方法の特徴と役割を理解する。

（1）統合開発環境

1. 統合開発環境とは
2. IDE（Integrated Development Environment）
3. 対話型操作環境
4. プロトタイピング（RAD）機能

（2）統合開発環境の特徴

1. プロジェクト管理
2. バージョン管理
3. GUI の作成
4. チーム開発
5. 作成補助
6. ビルド、デバッグ補助

（3）IDE の例

1. プロジェクト管理
   - ソースコード
   - 設定用ファイル
   - リソースファイル
2. バージョン管理
3. GUI の作成
   - グラフィカルな作成
   - 成果物の一括管理
4. チーム開発
   - ソースコードの連携管理
   - 別の開発者によるソースコードの修正支援
   - ソースコードのバグ混入防止
5. 作成補助
   - 予約語や関数名の補完
   - ソースコードの記述ミス防止
   - ソースコードの連携ミスの防止
   - ソースコードの色分け
   - 特定のシンボル管理
   - リアルタイムでのコンパイル時エラー検出

6. ビルド、デバッグ補助
   - コンパイラ、リンカとの連携
   - ブレークポイント管理

第 10 回 オープンソース統合開発環境の種類と特徴(ワークショップ 90 分)

オープンソース統合開発環境の種類とその特徴、各ツールの構成、機能を理解する。それぞれのツールを用いた開発方法の特徴と役割を理解する。HelloWorld プログラムを実際にコンパイル実行し、その機能や使い勝手を理解する。

(1)ActiveBasic

1. 言語仕様
2. 機能
3. 特徴
4. ライセンス
5. 開発環境設定
6. コンパイルと実行

(2)Eclipse

1. 言語仕様
   - Java 向け
   - その他の言語向け
2. 機能
3. 特徴
   - 豊富なプラグイン
   - 支援ツール
4. ライセンス
5. 開発環境設定
6. コンパイルと実行

(3)WideStudio

1. 言語仕様
2. 機能
3. 特徴
4. ライセンス
5. 開発環境設定
6. コンパイルと実行

第 11 回 Linux 開発環境におけるソフトウェア開発ワークショップ(ワークショップ 90 分)

Linux 開発環境におけるソフトウェアアプリケーション開発の作業内容、手順を実際に開発を行い、理解する。サンプルを用いて実習形式で開発ツールの特徴と役割を理解する。

(1)ソフトウェア開発の基本ツール

1. ビルドの自動化と autotools
2. make によるコンパイル手順の自動化
   - make の概要
   - ルールの記述方法
   - make の実行
   - Makefile 内での変数の利用
   - 暗黙のルールと事前定義の変数
   - ビルド以外の処理を行う
   - 手順の記述に関する詳細
   - 複数のディレクトリにまたがる make
   - コマンドラインオプション
3. configure スクリプト
   - configure の目的
   - configure が実行するテストの種類
   - configure のオプション
   - configure と他のファイルの関係
   - システムの標準名
   - configure と Makefile 中の変数
   - キャッシュファイルの利用

(2)ソフトウェア開発の効率化ツール

1. Autoconf を利用した Configure スクリプトの作成
   - Autoconf の概要
   - Autoconf の実例
   - Autoconf のテストの基本
   - テスト結果の利用方法
   - 結果のコントロール
   - テストに失敗した場合の対応
   - 問題の報告
   - 事前定義の出力変数
   - configure.ac 記述上の注意

- 独自の m4 マクロの利用

2. Autoconf が提供する m4 マクロ
   - プログラムの存在調査
   - ライブラリファイルの存在調査
   - ライブラリ関数の存在調査
   - ヘッダファイルの存在調査
   - 構造体の調査
   - 型の調査
   - 宣言の調査
   - ファイルの存在調査
   - コンパイラ
   - システムサービスの調査
   - 特定のシステムの判定
   - パッケージのカスタマイズ
3. Automake の利用
   - Automake の概要
   - Automake の実例
   - Automake が提供する make ターゲット
   - configure によるテスト結果の取り込み
   - ターゲットの指定と統一命名規約
   - ディレクトリの分割
   - Automake のその他の機能

第 12 回 Linux 開発環境におけるソフトウェア開発支援ツール概要
（講義＋ワークショップ 90 分）

Linux 開発環境におけるソフトウェア開発を支援するツールの概要、機能、操作手順を実際に開発を行い、理解する。実習形式で開発ツールの特徴と役割を理解する。

(1)開発支援機能

1. Libtool による共有ライブラリの構築
   - Libtool の概要
   - libtool を単体で利用する
   - ライブラリのバージョニング
   - Libtool を Autoconf に組み込む
   - Makefile.am に対する拡張
2. pkg-config による依存情報解決
   - pkg-config の概要
   - pkg-config コマンドの利用方法
   - メタデータファイル
   - Autoconf に組み込んで利用する
   - メタデータファイルの生成

(2)ドキュメントの記述

1. マニュアルページ
   - マニュアルページとは
   - groff
   - マニュアルページの記述
2. Texinfo ドキュメント
   - Texinfo の基本
   - ドキュメントの生成
   - Texinfo ファイルの記述方法
3. DocBook ドキュメント
   - DocBook とは何か
   - DocBook ドキュメントの記述
   - DocBook ドキュメントの変換
   - DocBook 文書の分割

(3)RPM によるソフトウェア管理

1. rpm コマンド
   - RPM とその背景
   - rpm の基本的な利用方法

- カスタムクエリ

2. RPM パッケージの作成方法
   - 独自の RPM を作成するメリット
   - パッケージ作成の流れ
   - パッケージ構築の前準備
   - spec ファイルの記述方法
   - パッケージのビルド
3. 高度な RPM のパッケージング
   - アーカイブの展開
   - サブパッケージの利用
   - スクリプトの実行
   - CPU アーキテクチャと OS の選定
   - マクロの利用
   - SRPM の利用

第 13 回 ソフトウェア開発ツールの評価(講義 90 分)

ソフトウェア開発ツール評価の観点とその内容、必要性を理解する。開発環境を用いた開発評価方法の特徴と役割を理解する。

(1)開発ツールの評価

1. 作業内容
2. 目的
3. 実施タイミング

(2)開発ツールの評価項目

1. 購入および保守の費用
2. 操作性と操作方法の習得期間
3. 前提となる各種開発技法の習得の必要性
4. デバッギング効率
5. 他のツールとの接続性と生産物の互換性
6. 統合開発環境としての評価
   - 開発生産性
   - バグ作り込み率の低減
   - テスト効率性

(3)開発環境の開発効率および品質向上の評価

第 14 回 Eclipse を用いたソフトウェア開発(ワークショップ 90 分)

Eclipse を用いたソフトウェア開発環境におけるプログラム開発および環境構築について実際の事象ごとの手順と留意点を実習を行いながら理解する。

(1)Eclipse のインストールと基本操作

1. J2SDK のインストール
2. Eclipse のインストール
3. 日本語化
4. Eclipse のユーザインタフェース
5. ワークベンチ
   - テキストファイルのエンコーディングを変更する
   - エディタに行番号を表示する
   - ファイルの関連付け
6. パースペクティブの機能
7. レイアウト変更

(2) EclipseUML プラグインの UML モデリング

1. EclipseUML プラグインの特徴
2. EclipseUML のインストール
3. EclipseUML でモデリングを実行
   - ダイアグラムを新規作成する
   - ユースケース
   - クラス図
   - シーケンス図
   - UML ダイアグラムの出力
4. ソースからのクラスダイアグラム生成
5. パッケージの依存関係の図式化
6. パッケージのクラス図
7. クラスの依存関係

(3)JDT

1. JDT
2. Java プロジェクト作成
   - HelloWorld クラスの作成
   - HelloWorld.java の編集
   - プログラムのコンパイルと実行
3. ヘルプの使い方
4. Java プログラミング

- クラス／インタフェース
- クイック・フィックス（即時修正）
- コンテンツ／コード・アシスト
- コメントアウト
- インポート宣言の編成

5. ナビゲート
- パッケージ・エクスプローラ
- アウトラインビュー
- 型階層ビュー
- ナビゲーション・ヒストリ
- タイプ（クラス、インタフェース）
- ファイル内部の Java エレメントにジャンプする
- タスク・タグ

6. 検索と置換
- 検索／置換機能
- Java 検索
- 検索ビュー
- インクリメンタル・サーチ

7. コード生成とテンプレート
- コード生成の使い方
- コード生成のカスタマイズ
- テンプレートの使い方
- 独自テンプレートの作り方

8. ビルド機能
- 自動ビルドの設定
- 手動でビルドする
- Ant を利用する

（4）JDT の便利な機能

1. ファイルの比較
2. Java スクラップブック
3. ローカルヒストリを使って復元する
4. ストリングの外部化
5. コンパイルによるストリングの外部化
6. オープンしたファイルをすべて閉じる

第 15 回 Eclipse を用いたソフトウェア開発ワークショップ（ワークショップ 90 分）

Eclipse を用いたソフトウェア開発環境におけるプログラム開発管理／デバッグについて実際の事象ごとの手順と留意点を実習を行いながら理解する。

（1）JDT を使ったデバッグ

1. デバッグの方法
   - ブレークポイントの設定
   - デバッガの起動とデバッグパースペクティブ
   - 基本ステップ実行
   - デバッグ中の変数値の参照・変更
   - 起動時の引数（起動構成）設定
   - 起動時のコマンドライン引数確認
   - 設定した起動構成をチームで共有
   - 高度なステップ実行
   - 変数へのアクセス・変更の検知と中断
2. プログラム解析に便利なプラグイン
   - Call Hierarchy プラグイン
   - Implementors プラグイン
   - Profiler プラグイン
   - プロファイル情報の見方

（2）Eclipse でリファクタリングを実践する

1. リファクタリングとは
   - リファクタリングの利点
   - いつリファクタリングするのか
   - リファクタリングとメトリクス
2. JDT を使ってリファクタリングする
   - JDT のリファクタリング
   - 名前変更
   - 移動リファクタリング
   - メソッド・シグニチャの変更
   - 匿名クラスをネストに変換する
   - ネストされた型をトップレベルに変換
   - プッシュダウン
   - プル・アップ
   - インタフェースに抽出
   - 使用可能な場合にスーパータイプを使用

- インライン化
- メソッドの抽出
- ローカル変数の抽出
- その他のリファクタリングについて

3. JUnit を使う

- 自動テストの重要性
- JUnit の設定
- JUnit プラグインでテストケースを書く
- テストの実行

（3）Eclipse でチーム開発

1. CVS サーバの設定とユーザの作成
2. チーム開発

- CVS を使ったチーム開発フロー
- CVS リポジトリの場所を登録する
- Eclipse のプロジェクトを CVS リポジトリのモジュールに（インポート）
- CVS リポジトリのモジュールを Eclipse のプロジェクトに（チェックアウト）
- リポジトリとの同期化（同期化ビュー）

3. CVS のより便利な使い方
4. Java プロジェクトとしてチェックアウトする

以上