6 ロボット開発環境

6.1 ドキュメンテーションフレームワーク

6.1.1 ドキュメントの必要性

Ｔ．Ｂ．Ｄ

6.1.2 reStructured Text / Sphinx

概要

- 以下の文書を記述

  http://www.planewave.org/translations/rst/quickstart.ja.html

  http://sphinx-users.jp/

  reStructured Textとは、首尾一貫したパターンを使う "Relaxed Text"（形式ばらないテキスト）とでも 呼ぶべきものです。

導入

1 Sphinxのインストール

- Linux
- Windows

1.1 Pythonのインストール

- http://python.org からWindows Installerをダウンロードし、インストールする。
- 環境変数のパスに追加する。
  - マイコンピュータのアイコンを右クリックして、プロパティダイアログを開きます
  - 詳細タブの下の方にある環境変数ボタンをクリックします
  - システム変数のPATHに以下を追加

```
C:\Python27
C:\Python27\Scripts
```

- インストール確認

  スタートメニューから、 コマンドプロンプト を起動するか、「名前を指定して実行」で cmd と入力して みましょう。ウィンドウが表れたら、 python[Enter] とタイプします。インストールしたPythonの バージョンを表す文章に続いて、 >>> という文字が表示されればインストールは成功です。 Ctrl+Z キーを押して終了しましょう。

1.2 easy_installコマンドのインストール

Pythonには easy-install という、外部ライブラリをインストールするのに便利なコマンドがあります。
何かインストールしたいプログラムやライブラリがあったとすると、コマンドを一つ入力するだけで、
実行するのに必要なものも一緒にダウンロードしてくれます。

http://peak.telecommunity.com/dist/ez_setup.py

上記のリンクを右クリックして保存します。URLを開くと、ブラウザによってはそのままダウンロードできます。
ファイルの中身が見えても、おちついて、右クリックで保存をすれば大丈夫です。
ダウンロードしたら、コマンドラインを起動し、該当のファイルのあるところまで移動してから、以下のように
実行します。

```
> python ez_setup.py
```

これで easy_install コマンドがインストールされます。ここまで行けば次はとうとうSphinxのインストールになります。

1.3 Sphinxのインストール

setuptoolsまでインストールされていれば、後は一瞬です。コマンドラインから以下のようにタイプします。

```
> easy_install sphinx
```

これで完了です。インストールが終わったら、コマンドラインから、 sphinx-quickstart[エンター] とタイプしてみます。以下のように表示されていればインストールは成功です。Ctrl+Cキーを押して中断しましょう。インストール作業は以上です。次は プロジェクトを作る に進んでください。

1.4 Sphinxのプロジェクト作成

http://sphinx-users.jp/gettingstarted/make_project.html

6.2 テスティングフレームワーク

6.2.1 テストの必要性

- 以下の文書を記述
  http://gihyo.jp/dev/feature/01/hudson/0001

Continuous Integration，以下CI）のおさらいをしましょう。CIは，Extreme Programmingに端を発し，Martin Fowlerによって広められた概念で，狭義には，別々に開発された部品を持ち寄ってお互いの動作を検証する「統合テスト」を早い段階から恒常的に行うことを指します。この当初の概念には必ずしも統合テストの自動化という考え方は含まれていませんでしたが，最近では，CIは単に統合テストだけではなく，広くビルド及びテスト全般を恒常的に行うことを指すようになり，またこれを現実的な工数で実現するための必須の手段として，ビルド・テストの工程を極力自動化する，という事が重要なポイントの一つになってきました。

この考え方の背景の一つには，コンピュータの高性能化・低価格化する一方，人件費はむしろ高くなっているという経済的な現実があります。この流れの結果，今日では，技術者の生産性の向上に少しでも寄与するならばコンピュータを湯水のように無駄遣いしても元が取れる，ということになってきました。こういう考え方に立てば，ソースコード管理システムに投入される変更一つ一つに対してビルドとテストを行って変更の質を確認する，というプロセスも決しておかしくはない事になります。

もちろん，最終的な目的は技術者の生産性を向上させる，つまり我々エンジニアが楽をする，という事なわけですから，CIを導入するのに手間が掛かるようでは本末転倒です。この点について，CruiseControlを始めとする初期のツール群には色々な問題がありましたが，ここ数年の間に登場した第二世代のCIツールによって， CIは開発の現場で実用可能なレベルに到達してきました。

Ｔ．Ｂ．Ｄ

6.2.2 Jenkins

- 以下の文書を記述
  https://wiki.jenkins-ci.org/display/JA/Meet+Jenkins

概要

Jenkinsは、ソフトウェアのビルドやcronで起動するジョブなどの繰り返しのジョブの実行を監視します。
これらのうち、Jenkinsは現在次の2つのジョブに重点を置いています。

1 継続的な、ソフトウェアプロジェクトのビルドとテスト:
つまり、CruiseControlやDamageControlが行うこと。

一言で言えば、Jenkinsは、容易ないわゆる「継続インテグレーションシステム」を提供し、開発者が変更をプロジェクトに統合でき、ユーザーがより新しいビルドを容易に取得できるようにします。自動化された継続的なビルドは、生産性を向上させます。

2 外部で起動するジョブの実行監視: cronによるジョブやprocmailのジョブで、リモートマシンで動作するものも含みます。例えばcronについて言えば、出力をキャプチャーした定期的なメールだけ受信し、こつこつとそれを見ます。おかしくなっていることに気がつくかどうかは、すべてあなた次第です。Jenkinsは出力を保存し、 いつおかしくなったのか容易に把握することができるようになります。

特徴

1 簡易なインストール: java -jar jenkins.war
を実行するか、サーブレットコンテナにデプロイします。

追加のインストールも、データベースも不要です。

2 簡易な設定:
豊富な入力時のエラーチェックとヘルプを備えたわかりやすいWebGUIを使用して、Jenkinsを

設定できます。もう手でXMLをいじる必要はありません。いじりたいのならそうすることもできますが。

3 差分のサポート:
Jenkinsは、CVSやSubversionからビルドへの変更の一覧を生成することができます。

これは、リポジトリの負荷を削減するとても効率的な方法で行われます。

4 永続リンク:
どこからでも簡単にリンクできるように、"最新のビルド"や"最新の安定ビルド"のような永続

(固定)リンクを含む、多くの画面は、クリーンでわかりやすいURLを持ちます。

5 RSS/Eメール/IM との連携:
失敗時にリアルタイムに通知をうけるために、RSSやEメールでビルド結果を

監視します。

6 ビルド後のタグ: ビルドが完了した後に、ビルドにタグを付与できます。

7 JUnit/TestNGによるテスト結果のレポート:
JUnitのテスト結果を、一覧表示および要約し、いつから失敗

しているのかなどの履歴情報とともに表示します。履歴の傾向はグラフ化されます。

8 分散ビルド:
Jenkinsは、複数のコンピュータで分散ビルド/テストを実行できます。このおかげで、開発者

の机の下に横たわっている何もしていないワークステーションを利用することができます。

9 ファイル指紋:
Jenkinsは、どのビルドがどのjarを生成したのか、どのビルドがjarのどのバージョンを使用

しているのか等々、追跡できます。この機能は、Jenkinsが管理しないjarでも機能します。そして、プロジェクトの依存性を管理するのにも有用です。

10 プラグインサポート: Jenkinsをサードパーティのプラグインで拡張できます。
開発チームが使用するツール

や処理をサポートするプラグインを書くこともできます。

導入

- 以下の文書を記述
  https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu

  1 インストール

```
wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo aptitude update
sudo aptitude install jenkins
```

※ apacheの記述も必要

6.3 ソースコードリポジトリ

6.3.1 ソースコードのバージョン管理

- 以下の文書を記述
  http://ja.wikipedia.org/wiki/バージョン管理システム

バージョン管理システム（バージョンかんりシステム）とは、コンピュータ上で作成、編集されるファイルの変更履歴を管理するためのシステム。特にソフトウェア開発においてソースコードの管理に用いられることが多い。バージョン管理システムの最も基本的な機能は、ファイルの作成日時、変更日時、変更点などの履歴を保管することである。これにより、何度も変更を加えたファイルであっても、過去の状態や変更内容を確認したり、変更前の状態を復元することが容易になる。更に、多くのバージョン管理システムでは、複数の人間がファイルの編集に関わる状況を想定している。商業的なソフトウェア開発やオープンソースプロジェクトなどでは、複数の人間が複数のファイルを各々編集するため、それぞれのファイルの最新の状態が分からなくなったり、同一ファイルに対する変更が競合するなどの問題が生じやすいが、バージョン管理システムは、このような問題を解決する仕組みを提供する。ただし、バージョン管理システムを個人のファイル管理に使用することも可能であるし、ソフトウェアのソースコードだけでなく、設定ファイルや原稿の管理などにも使うことも可能である。

6.3.2 Subversion

- 以下の文書を記述
  http://ja.wikipedia.org/wiki/Subversion

  http://subversion.apache.org/packages.html

Subversion（サブバージョン、サバージョン； SVN）はプログラムのソースコードなどを管理する集中型バージョン管理システムの一つ。2009年11月7日にApache Incubatorプロジェクトのひとつとなり、2010年2月17日よりApacheのトッププロジェクトとなった。ライセンスはApache Licenseに準じたものとなっている。
歴史的には広く使われているバージョン管理システムの一つにCVSがあった。CVSはよくできているが、ディレクトリの移動の管理やネットワーク対応の点、不可分な更新などでやや難があった。これらCVSの問題点を解決すべく開発されたのがSubversionである。
古くからオープンソースソフトウェアの開発においてはCVSが多く使われていたが、近年ではSubversionを使用するオープンソースプロジェクトも多くなりつつある。
Subversionは集中型（クライアント・サーバ型）であるが、その後、GitやMercurialやBazaarなどの分散型のバージョン管理システムが登場するようになった。例えば、Linuxカーネルの管理にはGit、Mozilla Firefoxの 管理にはMercurial、MySQLの管理にはBazaarが使われている。

6.3.3 Sourceforge

- 以下の文書を記述
  http://ja.wikipedia.org/wiki/SourceForge.JP

SourceForge.JP（ソースフォージドットジェーピー）は、日本のオープンソースソフトウェアプロジェクト向けのホスティングサイト。SourceForge.netの姉妹サイトで、OSDN社が運営している。
SourceForge.netの日本語版サイトとして、VA Linux Systems JapanのOSDN事業部によって2002年に設立（2002年3月ベータ公開、2002年4月正式運用開始）。現在は2007年9月にVA LinuxからスピンオフしたOSDN株式

会社によって運営されている。提供されているサービスはSourceForge.netとかぶる部分が多いが、コンパイルファームのようにSourceForge.JPにしかないサービスもある（詳細はサービスの節を参照）。

SourceForge.JPではホスティング費用は発生しないが、オープンソースプロジェクトホスティングサイトなので、開発成果はオープンソースとして公開する必要がある。ライセンスはOSIにオープンソースライセンスとして承認されているものが利用可能。

企業によるオープンソース活動の拠点としても利用されており、登録開発者には個人のほか、それらの企業に所属する開発者も多い。

2008年8月現在の登録プロジェクト数は3,263、登録ユーザ数は30,035。

6.3.4 Git

- 以下の文書を記述

  http://ja.wikipedia.org/wiki/Git

  http://git-scm.com/download

Git（ギット）はプログラムなどのソースコード管理を行う分散型バージョン管理システム。動作速度に重点が置かれている。Linuxカーネルのソースコード管理を目的として、リーナス・トーバルズによって開発された。現在のメンテナンスは濱野純（Junio C Hamano)が担当している。

Gitではワーキングディレクトリがリポジトリの全ての履歴を含んでいるため、中央サーバへのアクセスが不可能な状態であってもリビジョン間の履歴を調査することができる。

Linuxカーネルの開発では、Linux Kernel Mailing Listに投稿される多数のパッチをメンテナーたちがソースコードに適用するという形式を採用している。これらの作業を効率的にするため、当初BitKeeperというバージョン管理システムを用いていたが、このソフトウェアは商用ソフトウェア（クライアントはバイナリのみ無料で、サーバは商用だがBitMover社の好意で無料で使えていた）であった。この状況を快く思わない人々がBitKeeperのクローンを実装したことからこの環境が使えなくなってしまい（BitKeeper#ライセンス問題やBitKeeper#価格変更を参照）、その代替として2005年にGitが開発された。

Linuxのカーネルでは、相当量のソースコードを扱うため、変更点の抽出やリポジトリ操作に時間がかかっていては困るという状況になっていた。他の様々なバージョン管理システムをあたったが十分なものがなかった。
そのため、このような問題もできるかぎり解決できるよう、いくつかの案が導入されている（この部分は、他のバージョン管理システムにも同様の機能が導入されるようになった）。