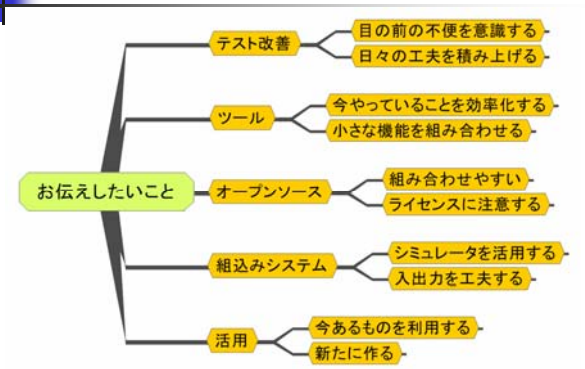
Communication and Technology Systems

オープンソース・ツールを活用した組込みシステムのテスト改善

キャッツ株式会社
ソフトウェア事業部
穴田 啓樹

本日の発表内容



テスト改善とは何か

ツールエンジニアの発達段階
1）目の前の不便に気がつかない
2）不便に気がつくが何もしない
3）手作業で工夫する
4）自分のためのツールを作る
5）組織のためのツールを作る

- 目の前の**不便を意識**することから始める

皆様の職場から、一定周期のリズムを刻む音が聞こえてきませんか？

・テストは繰り返しが多い
・「三回おなじ操作を繰り返したらやり方を考え直せ！」

- 日々の**小さな工夫**を積み重ねる

⇒ツール化すれば継続的に改善できる！

ツールで効果を上げるには

- **今やっていること**を効率化する
- 時間をかけている箇所を見つける
- **小さな機能**に分けて、**組み合わせる**

UNIX文化

基本機能にわける

・抽出する
・並べ替える
・加工する
…

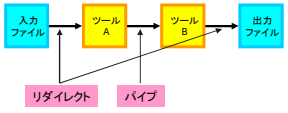
- 再利用しやすいツールになる

今回利用するオープンソースツール

- 利点 ＝ 組み合わせやすい

入出力のデータ構造が把握できる・ソースで動作を追える

- Cygwin /GNU Tools
  http://cygwin.com/
- Ruby、Ruby/Tk
  http://www.ruby-lang.org/ja/
  http://rubyinstaller.rubyforge.org/wiki/wiki.pl
- Freemind
  http://freemind.sourceforge.net/wiki/index.php/Main_Page



ツール連携（組合せ）の様子

- ライセンスの扱いに注意する

例）GPL：改変したコードは公開しなければならない
⇒自社向けにカスタマイズしたときに公開する必要がある

- 基本的に無保証である

組込みシステムのテストならでは効果を上げる工夫

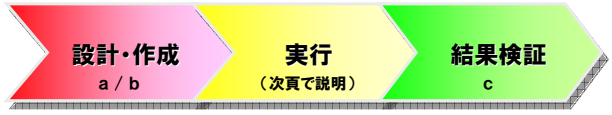
組込みシステムは、実機環境（**ターゲット**）と、開発環境（**ホスト**）
が異なることが多い ⇒ 何らかの方法で実機動作ログを残す

バグを修正するときの手がかり

- **ターゲット**に動作解析用の入出力を設ける
  - 試作機などでデバッグ用のシリアルポートを設ける
  - ログ記録用のRAMを設けてエミュレータで吸い上げる
- **ホスト**上のシミュレータを活用する
  - テストの目的を明確にすれば、高価なシミュレータを利用しなくても必要な部分を切り出して、ホスト上でテストすることもできる
  - 実機がない段階からテストを始めることができる

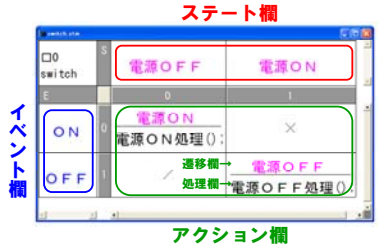
活用事例の紹介
「状態遷移表テストでの活用」

- **既存のオープンソースツールを活用する**
  - a）Freemindを利用した状態遷移表テストの設計
- **新しくツールを作る**
  - b）Ruby/Tkを利用したテスト仕様書/テストドライバの作成支援
  - c）Rubyを利用したテスト実行結果のログ解析



テストのフェーズで活用部を分類

事前知識：状態遷移表の読み方
状態遷移表テストとは



状態遷移表の読み方

- 電源OFF状態のときに、ONイベントが発生すると、電源ON処理を行い、電源ON状態へ遷移する。
- 電源OFF状態のときに、OFFイベントが発生すると、無視「/」する。
- 電源ON状態のときに、OFFイベントが発生すると、電源OFF処理を行い、電源OFF状態へ遷移する。
- 電源ON状態のときに、ONイベントイベントが発生してはいけないことを設計者が「×」で表現している。

状態遷移表テスト

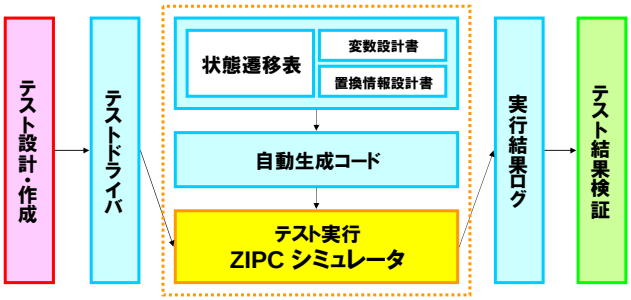
・状態遷移表パステスト　　イベントを順番に起こして正しく遷移するか確認するテスト
・状態遷移表マトリクステスト　状態遷移マトリクスでもれなく遷移を確認するテスト

『Open SESSAME Seminar 初心者向けテキスト』西 康晴

実行環境の説明

- **シミュレータ（ZIPC）※を活用する**

（※）状態遷移表ベースの組込みCASEツール

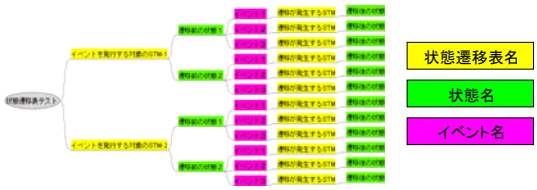


a）Freemind(※)を利用した状態遷移表テストの設計

※Tony Buzan氏が考案したマインドマッピングというメモ記述技法に近いものを実現したオープンソース・ソフトウェア。

- **状態遷移パステストをツリーで設計する**

マインドマッピングの発想法に基づいて、テストケースを描いていく。
⇒「状態遷移表名」・「イベント」・「状態」の並べ方、色分けのルールを決めておくと頭が整理される（**マトリクスとツリーの相乗効果**）。



状態遷移表テストの設計にマインドマッピングツールを利用する

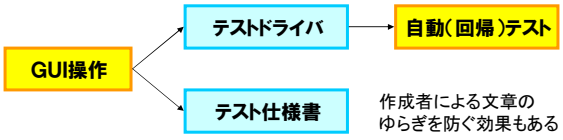
デモ

b）Ruby/Tk(※)を利用したテスト仕様書・テストドライバの作成支援

※Rubyは、まつもと ゆきひろ氏が開発しているオブジェクト指向スクリプト言語。Ruby/Tkは、RubyからTcl/TkのToolkitを利用できるようにした環境。

- テストにおいて、テスト仕様書（テストシーケンス）作成、テスト実施の時間が占める割合は高い。
- テスト仕様書は定型、かつ、同一単語が多数出現。
- Ruby/Tkで簡単にGUIを構築できる

⇒入力作業を効率化



作成者による文章のゆらぎを防ぐ効果もある

一度の操作でテスト仕様書とテストドライバの両方を作成するツール

テストシーケンス作成の工夫

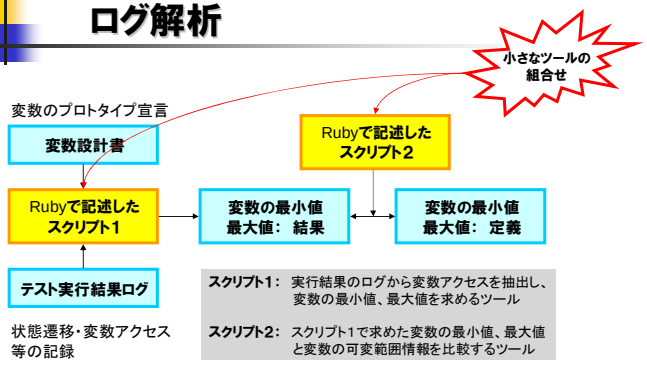
- **テストシーケンスの構成要素を分類してタグにする**
  - **事前準備**　このテストを始める前に整えておくこと
  - **事前条件**　このテストを始める前にあるべき状態
  - **イベント**　このテストで発生させる事象
  - **事後条件**　このテストを終えた後にあるべき状態
  - **事後処理**　このテストを終える前に後始末しておくこと
- **「<タグ>データ」でテストシーケンスを構成する**

⇒状態遷移表を修正したときに、テストシーケンスの修正が必要な箇所の**追跡支援**ができる。

例）あるセルの遷移先が変更された場合、「事前条件」「イベント」で特定されるテストシーケンスの事後条件を修正する。

デモ

c）Rubyを利用したテスト実行結果のログ解析



変数が設計通りに利用されているかを検証するツール

Rubyをログ解析（＝テキスト処理）に利用する基本

- **テキスト処理の基本構造**

```
while line=gets          # 1行読み込む → 繰り返す
  if /正規表現/ =~line   # マッチする行を探す
    処理                 # 必要な処理を行う
  end
end
```

Rubyが有用なケース
- 複雑な検索をしたい
- 部分的に文字列を取り出したい
- 文字列を整形したい
- 文字列を置き換えたい

- **ポイントは正規表現**

文字列の中から特定パターンを探すときのルールを表現する方法。文字とメタ文字で構成する。

例）文字列「aaaaa test aaaaa」が正規表現 /¥s¥w+¥s/（空白で囲まれた1文字以上の英数字）にマッチするのは、「 test 」である。

代表的なメタ文字

¥w	英数字
¥W	非英数字
¥s	空白文字
¥S	非空白文字
¥d	数字
¥D	非数字
*	直前の表現の0回以上の繰り返し
+	直前の表現の1回以上の繰り返し
()	グループ化

ログ解析ツールでの利用例

ZIPCは次のようなログを出力する

「28 SME 67000000 RUN “CD” “CD” “ubSpeed” “1” “2“」

これは変数「ubSpeed」が「1」から「2」へ変化したことを示している。ここで、変数名「ubSpeed」と変数値「2」を抜き出して、利用するには、

/¥d+¥s¥w+¥s¥d+¥s¥w+¥s”¥w+”¥s“¥w+”¥s“(¥w+)”¥s“¥d+”¥s“(¥d+)”/

にマッチさせると、$1に変数名、$2に変数値（の文字列）が入る。これを利用して、変数毎の最小値、最大値を求めている。

ツール活用の効果

- **a）Freemindを利用した状態遷移表テストの設計**
  当初から利用していたので比較できないが、状態遷移パステストのイベント列を考えるときに、頭が整理される効果は高い。
- **b）Ruby/Tkを利用したテスト仕様書・テストドライバの作成支援**
  あるテスト仕様書で3日かけていたのが、同規模で1.5日に短縮した。
- **c）Rubyを使ったテスト実行結果の解析**
  1回あたり手作業で約15分かけていたのが、約3秒に短縮した。

ツール活用のポイント

- **GUIとスクリプト処理の良さを使い分ける**
  過度なGUIは生産性を低下させる。
  必要に応じてスクリプト処理を使いこなせ！
- **正規表現を使いこなす**
  単純検索だけであきらめないこと。
  正規表現で欲しいデータを作り出せ！
- **小さな機能単位でツール化する**
  1つのツールであれこれ作りこまない。
  やりたいことの本質をシンプルなツールに実現せよ！

お伝えしたいこと（まとめ）

目の前の不便を意識しよう

オープンソースツールを使いこなそう

小さな工夫を積み上げよう

節約した時間で、テストの質と技術を向上させよう！