Prof. Mike Hennell
Technical Director

LDRA Testbed® and IEC 61508

内容

- IEC 61508 **スタンダード**
- MISRA **ガイドライン** for C and C++
  - **組込みシステムの信頼性向上**
- CERT C **セキュアコーディングスタンダード**
  - **あらゆるシステムの安全性向上**
- **日本発** IPA/SEC **コーディング作法ガイド**
  - **あらゆるシステムの信頼性、メンテナンス性、ポータビリティ、スタイル**
- LDRA Testbed **でできること**

IEC 61508 Requirements

IEC 61508 **は主にプロセス志向であるが**V&V**に対するガイドラインもある**

その [IEC98, Part 3, **条項** 7.9.2.7]**には、**V&V**実施で実行されるべき、コード検証、ソフトウエアモジュールテスト、インテグレーションテストなどが含まれる**

Table III. IEC 61508 recommendations for code verification

- The recommendations cover five main categories:
  IEC61508**の5つの推奨**:

1. Formal proof **形式的証明**
2. Probabilistic testing **確率論的テスト(ランダムテストなど)**
3. Static analysis **静的解析**
4. Dynamic analysis and testing **動的テストとその解析**
5. Software complexity metrics **ソフトウエア複雑度の尺度**

これらいくらかのテクニックに関してはスタンダード内に追加の表がある

IEC 61508

- SIL（**安全性レベル**）**によって各テクニックは、以下のように分類される**
  - Neither recommended nor not recommended
    推奨、推奨しないのどちらでもない
  - Not recommended (NR) **推奨しない**
  - Recommended (R) **推奨**
  - Highly recommended (HR) **強く推奨**
- **問題なのは、特定のアプリやSILに対して、どのテクニックが推奨されるのか、されないのかを決定できる仕組みが無いこと**

Software faults **ソフトウエアの障害**

- **IECスタンダードの目的は、残存するソフトウエア不良を、信頼度要求事項（SILのレベル）に応じて保証すること**
- **ここでソフトウエア障害とは、要求される振る舞いに対するソフトウエアの逸脱**

Software Faults ソフトウエアの障害

- ソフトウエアの障害は以下のいずれかに分類:
  - アプリに特化した障害
    - 特定のアプリケーションドメインのみに関連する障害
      (ブレーキ作動の失敗)
  - テクニカルな障害
    - 基本的にどのようなアプリでも障害となる
      (配列境界エラー、不当な配列参照)

Static Analysis and Dynamic Analysis

- 静的解析ツールはテクニカルな障害検出に向く
  - これらの障害は言語、プログラミング上の問題、欠陥である
- 動的解析は基本的にアプリケーション障害の検出に向く
- 動的解析といくつかの形式的手法のみアプリケーション障害を検出することが出来る
- これはアプリケーションの仕様である要件を明確に用いることから

Static Analysis Limitations

- どのような静的解析でも、いくつかの形式手法でも全ての障害を検出することは保証できない
- これは有名な‘halting theorem’（halting problemチューリングマシンの停止性問題）の帰結 （「(任意の) プログラムにバグがないか判定できるプログラム」は存在しない)
- 全ての障害を検出できない、あるいは偽陽性メッセージの生成というような結論になる。また不完全でもある

Dynamic Analysis

- 動的解析は全ての起こり得る障害を検出できる能力を持つ得る
- 問題は、障害が露呈される条件が複雑となり、通常計り知れない
- 一般に動的解析テクニックは、多様なカバレッジ尺度の到達要求により実際には制限される

LDRA Testbed について

- LDRA Testbed は、静的解析、動的解析の両方を提供
- 形式手法を用いた解析も
  - System wide data flow analysis システムワイドなデータフロー解析
  - System wide pointer analysis システムワイドなポインタ解析
  - File operation analysis システムワイドなファイル操作の解析
  - Array bound analysis システムワイドな配列境界解析
  - Infeasible code detection 実行不可能なコードを検出
  - Ineffective code detection 無効なコードを検出
  - Information flow analysis インフォメーションフロー解析

LDRA Testbed について

- 広範なプログラミングガイドラインをサポート
  - MISRA C (1998 and 2004) and C++
  - JSF++ AV
  - CERT C Secure Coding Standard
  - SEC C (from Japan)
  - Others
- システムワイドな複雑度解析
  - Control flow complexity コントロールフローの複雑度
  - Programming construct complexity プログラミング構造の複雑度
  - Data flow complexity データフロー複雑度

Dynamic Analysis 動的解析

- 動的解析の完全な要件に対するトレーサビリティをサポート
- 以下のカバレッジレベルに対応
  - Statement coverage
  - Branch coverage
  - MCDC coverage
  - LCSAJ coverage
- ソースコードレベル、オブジェクトコードレベルのカバレッジ解析

Conclusion まとめ

- LDRA ツールスイートはIEC 61508 で求められる主要なテスト手法を全てサポートできる唯一のツール
- 全SILレベルに対するエビデンスを完全に提供できる唯一のツール
- セーフティクリティカル、ミッションクリティカルなソフトウエアで30年以上広く用いられてきている

Finally

- SECのコーディングスタンダードは、一般的にシリアスな20のプログラミングエラーを公開
- LDRA Testbed スイートはこれら全てを検出可能

MISRA-C++:2008
MISRA-C:1998/MISRA-C:2004
CERT C, JSF++AV, HIS, HI C++, DERA C
LM Train Control Program, JPL, GJB
LDRA UML C/C++, ISO9000/9001:2000
IPA/SECコーディング作法ガイド
IEC 61508(SIL 4-1), DO-178B(Levels A〜E)