JaSST '09 Tokyo（テクノロジーセッションA3）

静的解析技術の当社取組みについて

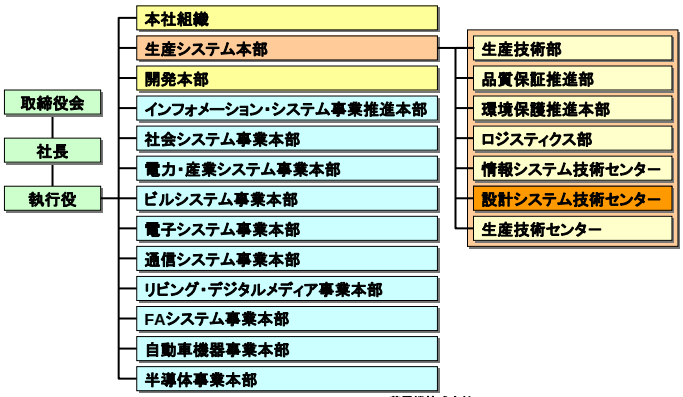
2009年1月28日
三菱電機株式会社　設計システム技術センター
ソフトウエアエンジニアリング部　設計実装検証グループ

藤本　卓也

三菱電機株式会社

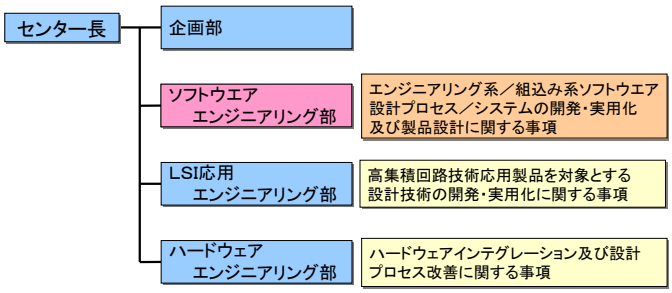
1. 設計システム技術センター（紹介）

三菱電機（株）技術センター組織

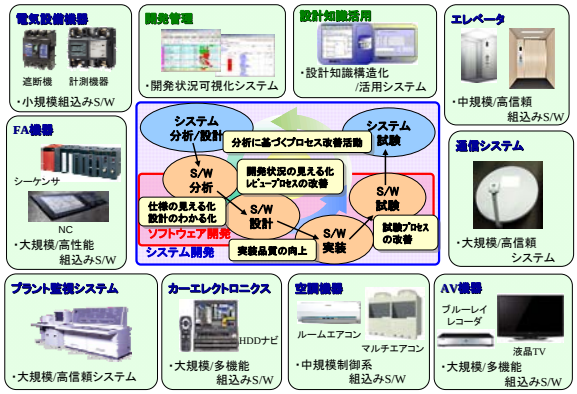


設計システム技術センター組織

所掌範囲：全社および関係会社のH/W，S/Wの設計
生産性向上技術に関する事項



ソフトウエアエンジニアリング部の活動



背景

- 流用開発の割合が高い。
  - 製品ソフトの品質は流用元ソフトの品質に左右される。
  - 流用元ソフトの品質レベルを確認，確保する必要がある。
- 製品差別化へ対応
  - 高機能化：新規開発ソフトの規模増加
  - 低価格化：開発コストの圧縮
  - 迅速な市場投入：開発期間の短縮
- 実装 ⇒ 製品に組み込まれるコードの記述
  - 実装以降はコードの世界となる。
  - 設計が良くても最終的なソフト品質はコードで決まる。

導入の目的

- 実装品質の向上による開発コスト，期間の圧縮
  - 現状のソフトウェア品質の把握
  - 誤り混入の未然防止
  - 誤り検出作業の効率化
  - 誤り検出率の向上

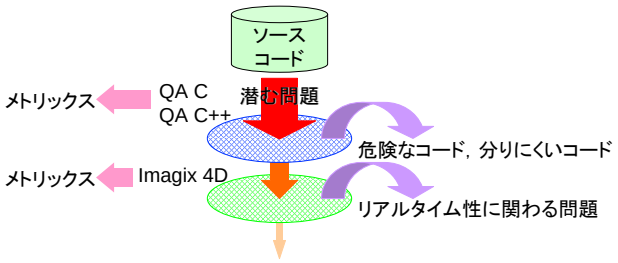
静的解析ツール
・QA C/QA C++
・Imagix 4D
の活用

➔ 不具合に繋がる危険コード記述の早期検出及び修正
➔ 潜在ロス及び手戻り工数の削減

これまで以上のソフトウェア品質の確保へ

静的解析ツールの活用

- 検出できる問題の異なるツールと合わせて使用（補間）し，問題検出，品質のフロントローディングを強化



ツール導入の時期

- '90年代後半：QA C/QA C++（静的解析ツール）を導入
  - 試使用，カタログによる比較評価の結果，機能的に最も優れていたQA C/QA C++を選択
  - 静的解析技術活用の実装品質向上の活動を本格化
- '00年代前半：Imagix 4D（構造解析ツール）を導入
  - マルチタスクシステムにおける問題点の手がかりを与えてくれる機能を持っていることから選択
  - タスク間の問題を可視化するための環境を増強

静的解析ツール（QA C/QA C++）とは？

- 英国Programming Research社の製品。日本では東陽テクニカ（株）が販売
- QA Cは7.1J，QA C++は2.3.1Jが最新（2009年1月現在）
  - ソースコードの特性の測定 ➔ メトリックス測定
    - QA Cは69，QA C++は26種類の測定が可能
  - ソースコード中の危険性の検出 ➔ 警告出力
    - QA Cは1,348，QA C++は 1,035種類を出力
- 危険性の高い箇所を検出，特定できる。

構造解析ツール（Imagix 4D）とは？

- 米国Imagix Corporation社の製品。日本では東陽テクニカ（株）が販売
- 6.4.0が最新バージョン（ 2009年1月現在）
- 大別して，次の4種類の情報を出力する。
  - ソフトウェア構造を示す各種の図
  - ソフトウェア特性を示すメトリクス
  - 変数，関数，タスク，イベントの危険な使用状況に関するレポート
  - 危険なソースコード記述に対する警告
- タスクに跨った問題を指摘する機能がある。

実施してきたこと

- 解析支援
- 解析結果分析支援
  ➔ ソフトウェア品質の把握，誤り検出作業の効率化
- 解析環境の整備，提供
  ➔誤り検出作業の効率化
- 警告の危険度による分類，ピックアップ
  ➔誤り検出率の向上
- MISRA-Cへの対応
  ➔誤り検出作業の効率化，誤り混入の未然防止
- コード品質の点数化
  ➔ ソフトウェア品質の把握
- コーディング規約との連携
  ➔ 誤り混入の未然防止

解析，解析結果分析支援

- 社内向け静的解析センターとしての活動
- QA C/QA C++，Imagix 4Dを含む静的解析技術に関するサポートセンターとしての活動
- 解析レポートの作成，提示
- 品質改善のための処方箋の作成，提示
- マルチタスクプログラムでの問題点抽出

社内向け静的解析センターとしての活動

- 開発部門からの依頼で解析作業及び解析結果分析を弊センターで実施する。
- 短期間で問題となる可能性の高い箇所をピックアップした解析結果を開発部門に提出する。
- 開発部門には，指摘内容の確認と問題性有無の判断，対応が必要となる箇所の修正に専念してもらう。

- 開発部門:　限られた開発期間での効率的な品質向上
- 弊センター:　全社のソフト品質情報の収集
　品質，生産性向上施策へのフィードバック

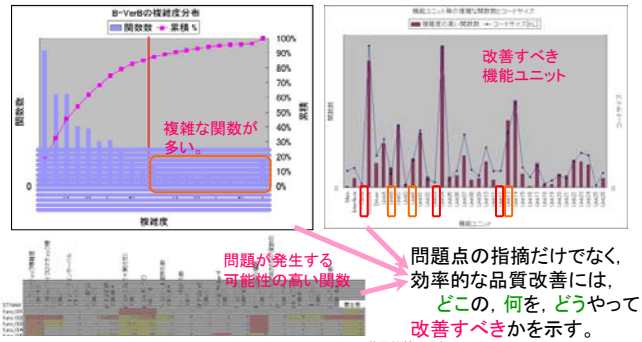
を行うことができる。

解析レポートの作成，提示

- 解析レポート(例)

処方箋の作成，提示

- 処方箋(例)



問題点の指摘だけでなく，効率的な品質改善には，どこの，何を，どうやって改善すべきかを示す。

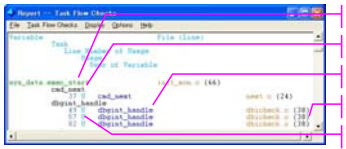
マルチタスクプログラムでの問題点抽出

次に示すような情報に着目し，危険な項目を検出

- 複数タスクに書き込まれる変数
  どのタスクがいつ値を書き込むが分かりにくく，タスク間共有変数の中でも危険度が高い。
- 書込み前に読み込まれる変数
  関数の呼出し関係の誤りや排他制御の不備により，初期化されていない状態で読込みが行われている可能性がある。
- タスク間共有変数
- ・・・

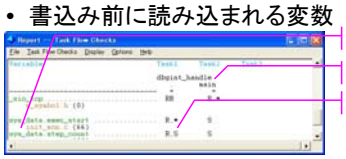
マルチタスクプログラムでの問題点抽出(2)

- 複数タスクに書き込まれる変数



書込みが発生している変数
書き込んでいるタスク
書き込んでいる関数
書き込んでいる位置
書込み保護(排他制御)の有無
U:保護無し！

- 書込み前に読み込まれる変数



チェックすべき変数
アクセスしているタスク
アクセス順序(R:Read，S:Write)
アクセスしているタスクは二つのみだが，S(et)の前にR(ead)が発生する可能性がある。

解析環境(QA Cビューア)の整備

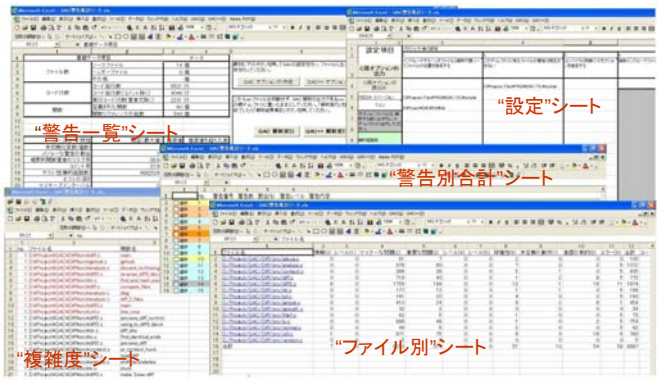
- QA C/QA C++は強力だが，使いにくい点もある。
  - 警告付きソースリスト，メトリックスレポートの出力機能はあるが，解析結果に対する集計・分析機能が弱い。
  - 膨大な情報が出力され，真に重要な情報が抽出しにくい。
- ExcelのVBAを使用したフロントエンドを開発
  - GUIによる操作性を確保
  - Excelの機能を利用した解析結果の集計及び分析機能を強化

➔ 静的解析技術の展開，定着のために社内にも展開

解析環境(QA Cビューア)の整備(2)

- QA Cビューアの機能
  - 設定機能
    - 解析時のオプションの設定
    - 警告メッセージの出力／抑制の選択
    - メトリックスの推奨値の設定
  - 解析機能
    - ディレクトリ下の全ソースファイルの解析
    - 指定したファイルのみの解析
  - 分析支援機能
    - 警告の分類，種類別合計
    - 指定，重要警告の抽出
    - 重要メトリックス情報の表作成

解析環境(QA Cビューア)の整備(3)



警告の危険度による分類，ピックアップ

- QA C/QA C++の問題点
  非常に多くの警告が出力される。
  - どの警告が重要なのか分かりにくい。
  - 重要な警告が埋もれてしまう。
  - 全ての警告をチェックし切れない。
  - 問題のない箇所で警告が検出される場合(誤認識)がある。
- 危険度により，全警告を5段階に分類
- 検出されると確実にシステムに重大な影響を及ぼす可能性のある警告をピックアップ

➔ 効率良く警告をチェックできる目安を作成，展開

警告の危険度による分類，ピックアップ(2)

- 全警告を危険度により次に示す5段階に分類
  - 危険度A:最優先でチェック/対応すべき警告
  - 危険度B:チェック/対応すべき警告
  - 危険度C:チェック/対応を推奨する警告
  - 危険度D:できれば，チェックした方が良い警告
  - 危険度E:無視しても問題となることが少ない警告
- 確実に不具合の原因となる警告をピックアップ
  - 危険度3:約40種類の警告
  - 危険度2:危険度3警告を含めた約80種類の警告
  - 危険度1:危険度3，2警告を含めた約160種類の警告

MISRA-Cへの対応

- 欧州の自動車業界発信のC言語コーディングガイド
- '98年版(127ルール)及び'04年版(141ルール)の二つが制定されている。
- 自動車業界以外でも広く認知され，使用されている。
- MISRA-Cチェッカパッケージが販売されており，QA Cでルール違反チェックを行える('98年及び'04年版に対応)。
- 電子化された手頃なルールの一覧(表)がない。
- QA Cビューアでの対応の要望があった。

MISRA-Cへの対応(2)

- MISRA-C('04年版)のルール一覧

MISRA-Cへの対応(3)

- QA Cビューア:“MCM解析”ダイアログ


MISRA-C1('98年版)対応　MISRA-C2('04年版)対応

コーディング規約との連携

- 問題の効率的な除去よりも，最初から問題の可能性を入れ込まないことが重要
  - ➔コーディング規約を策定し，これを遵守することで問題となる可能性のあるコードを記述させないようにする。
  - ➔しかし，十分に遵守される保証はない。

- 従来のプログラムの保守性を向上させる。
  - 命名規約
  - スタイル規約
  - コメント規約

＋

- 問題となる可能性の作りこみを抑制する。
  - 実装規約

を持つコーディング規約を策定し，静的解析ツールで違反をチェックする。

コーディング規約との連携(2)

- 実装規約の一覧と例
  - 関数
  - 演算子
  - 変数
  - 定数
  - データ型
  - 演算及び文
  - 制御構造
  - プリプロセス
  - メモリ管理
  - 移植性及び保守性
  - 効率性
  - メトリクス

[演算子の評価タイミング]
条件式における，値を変更する(副作用を起こす)演算子の使用は禁止する。
値を変更する演算については別の独立した文で行い，その結果を条件式で使用すること。

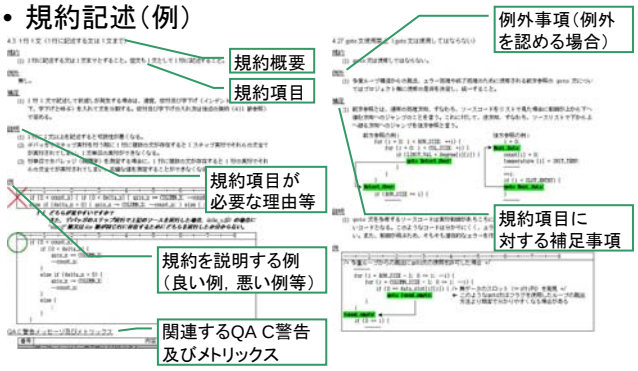
```
(×) if ((++cnt) == MAX) {
(○) ++cnt;
    if (cnt == MAX) {
```

[マクロ定義]
#define文で関数マクロを定義する場合，定義内で使用される引数は必ず小括弧で囲むこと。

```
(×) #define square(x) x * x
(○) #define square(x) ((x) * (x))
```


コーディング規約との連携(3)

- 規約記述(例)



効果，今後の予定

- 効果
  - 手戻り抑制による生産コスト低減
  - 保守性，可読性の高いコードの生産性向上に寄与
- 今後の予定
  - 施策内容のブラッシュアップ
  - QA C/QA C++，Imagix 4Dの更なる活用
  - 新しい技術の導入，活用
  - グループを含む全社への静的解析技術の展開，定着

MITSUBISHI
三菱電機
Changes for the Better