プログラム品質検証ツールの運用考察

中泉雅行†　　田部美智昭　　玉本亮

日本電気通信システム株式会社‡

１．はじめに

ソフトウェアの開発において、問題を開発初期工程で検出し修正することはソフトウェアの品質向上およびトータルコストの削減に結びつくことが知られている。また、その手法としてはプログラム品質検証ツールである静的チェックツールを使用することが有効とされている。

本論文では、プログラム品質検証ツールを運用してソフトウェア開発を実施した結果と問題点およびその対処方法を紹介する。

２．市販ツールによる試行

ソフトウェアの開発において、開発の工程の初期段階である製造工程の時点の品質が悪いと、後工程の検査に要する工数が膨大になってしまう。そこで社内プロジェクトの製造直後のプログラム（表１）に対して市販の静的チェックツールを導入して試行してみることにした。結果を以下に示す。（表２）

表１　プログラム規模

ファイル数	655　ファイル
関数の数	5430　関数
コード行数	759　ＫＬ

表２　試行結果

警告の総数	24502　件
警告の種類	163　種類
警告の密度	32.2　件／ＫＬ

その結果、以下の問題があった。

（１）　出力される警告の数があまりにも大量。

（２）　開発経験の浅い開発者にとって警告の重要度の違いがわからない。

（３）　各開発担当者がすべての警告をチェックすることは時間的に不可能。

そこで、Linux 標準コマンドである splint のチェック機能をベースとした静的チェックツールを開発することにした。

Employment consideration of a Program quality verification Tools

†Masayuki Nakaizumi, Michiaki Tabe, Akira Tamamoto

‡NEC Communication Systems,Ltd

３．プログラム品質検証ツールの開発

市販ツールによる試行結果を踏まえて、以下の点をポイントにプログラム品質検証ツール開発をおこなった。

（１）　警告のランク分け

検出される警告を、過去の経験および参考文献[1]を参考にして重要度別に特Ａランク・Ａランク・Ｂランク・Ｃランクのように４段階にランク分けを行った。（図１）

（２）　警告のフィルタリング

ＢおよびＣランクの警告に対しては、大量の警告が出力されることを抑止するために、１種類あたりの警告の出力数に上限を設け、一定数以上の警告を出力しないようにした。

（３）　警告のソーティング

同一ランクの警告は、１種類あたりの警告数の少ないものから出力するようにソートした。また、同一内容の警告はファイル毎に行番号をキーに逆順に出力する機能を追加した。

（４）　問題集中箇所の絞込み

解析結果を統計的に分析し問題が集中しているファイルをピックアップして表示するようにした。

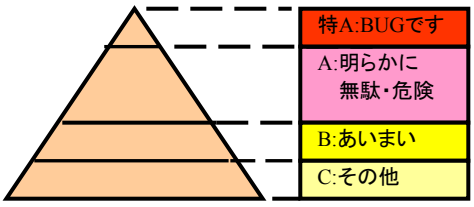

図１　警告出力イメージ

４．プログラム品質検証ツールの導入

今回開発したプログラム品質検証ツールを市販ツールの試行を行ったプロジェクトに導入してみた。（表３）

表３　導入結果

警告の総数	1998　件
警告の種類	31　種類
警告の密度	2.63　件／ＫＬ

その結果は、警告の総数が１９９８件となり約１／１２となった。また、警告がランク分け

されより重要な問題から結果が出力されるようになり、各開発担当者も限られた時間で警告をチェックできるようになった。

５．品質検証ツールの運用

導入結果より、品質検証ツールの効果的運用が期待できるため、別プロジェクトにて本格的に運用することにした。

該当プロジェクトは１つのシステムが複数の機能ブロックから構成されるもので、複数のグループにより全国各地の拠点にて開発しているものであった。

ツールを運用するにあたっては以下のルールを設定した。（図２）

- 各拠点では担当者がファイル単位で随時検証を実施する。
- 毎週１回、金曜日の夜にネットワーク経由ですべての機能ブロックの最新ファイルを一ヶ所のサーバマシンに集め、定期的にまとめて一括検証を実施する。
- 検証結果は上位スキル者がチェックを行い、警告の中で担当者がわかりにくいものについては解説を追記して各開発担当者に渡す。
- 各開発担当者は、検証結果のうち特Ａ・Ａランクの警告に対しては必ず修正することとし、修正が不要な場合はその理由を明確にするためにソースコード中にコメントで記入することにした。

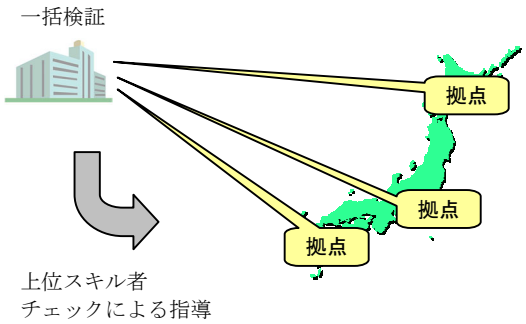

図２　運用イメージ

さらに、運用を促進および定着させるために以下の対策を実施した。

- 各開発者向けに、警告された箇所に対応するソースコードにジャンプできるようにした。
- 警告番号に対応する詳細解説と修正例を記述したＷＥＢ解説ページ（図３）へジャンプできるようにした。
- 管理者向けにテキスト形式の検証結果をＣＳＶ形式に変換し、データ管理を簡単にできるようにした。

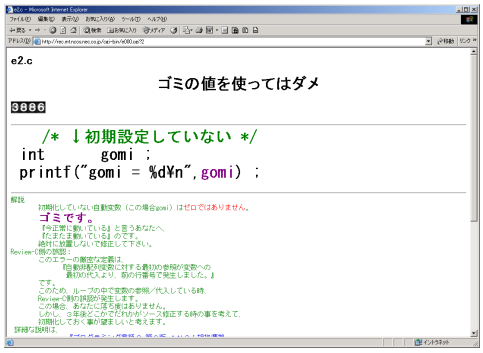

図３　ＷＥＢ解説ページ

これらのルールや対策を実施して品質検証ツールを運用した結果、以下のような効果を得ることができた。

- 検証結果の問題点がピックアップされて出力されるため、効率よくレビューや修正を実施することができ、トータルの工数を削減することができた。
- Ａランク以上の警告を無くすルールにより、グループおよび拠点間の品質を一定にすることができた。
- 定期的に一括検証を実施することにより品質の推移状況を目視管理できるようになった。
- ＷＥＢ解説ページの作成により、特にスキルの浅い開発者のスキル向上を補助することができた。

６．品質検証ツールの今後

今回のプロジェクト開発でのプログラム品質検証ツールを効果的に使用することができた。今後は、より検証の精度を高めて有効な警告を出力するようにアプローチしていきたい。

参考文献

[1] Brian W.Kernighan and Dennis M.Ritchie:
“The C Programming Language”
(邦訳：石田晴久訳プログラミング言語Ｃ
共立出版)

[2] Steve Summit:
“C Programming FAQs Frequently Asked Questions”
(邦訳：北野欣一訳Ｃプログラミング FAQ
トッパン)