ソフトウェアテストシンポジウム 2004
JaSST'04: Japan Symposium on Software Testing 2004

XML 形式の仕様書作成によるソフトウェア机上チェックの効率化
－机上デバッグ支援ツール Prio（プライオ）－

山田 信幸　　鈴木 幹雄　　坂 守

アイシン精機株式会社信頼性技術部　〒448-8650 愛知県刈谷市朝日町 2-1
E-mail:　nyamada@rd.aisin.co.jp,　{mikio, mban}@rd.aisin.co.jp

あらまし　仕様書に記述されている情報とプログラムに記述されている情報に含まれる単純な不具合を発見することを目的として，仕様書のテキストにプログラム情報を埋め込んだ XML 文書を作成し，不整合部分を自動的に発見するツール「Prio(プライオ)」を作成した．このツールを用いた机上チェックを行うことにより，仕様書とプログラムの不整合ミスを評価工程前に大幅に減少することができた．また，得られた XML データを後工程で活用し評価作業時間を短縮することができた．

キーワード　XML，仕様書，テスト

Automatic code check with using spec document in XML format
－Code check support tool　'Prio'－

Nobuyuki YAMADA　　Mikio SUZUKI　and　Mamoru BAN

PRODUCT RELIABILITY ENGINEERING DEPT., AISIN SEIKI CO.,LTD
2-1 Asahimachi Kariya Aichi 448-8650 JAPAN
E-mail:　nyamada@rd.aisin.co.jp,　{mikio, mban}@rd.aisin.co.jp

Abstract　In this paper, automatic code check with using specification document in XML format is discussed. We developed the software check support tool. This tool composed of the following modules

- Editor-module　input program informations to specification document parts with easy operation
- Check-module　inspects specification document inconsitencies with program

With using this tool, we could decrease the consistency errors in the specifications and the program. And, we could shorten test time by using XML data.

Keyword　XML, Specification, Test, Document

1. 概要

本報では，仕様書とプログラム間の関係を設定することにより，仕様の不整合とプログラムの不整合を自動的に検出することを目的としたオリジナルツールを作成し，ソフトウェアチェックを効率化した事例について紹介する．

2. 当社のソフトウェア開発の特徴

当社では自動車用システム製品の設計・開発・生産を行っており，各製品のコントローラーであるＥＣＵ(Electronic Control Unit)に搭載するソフトウェア開発を行っている．

近年，チップの高機能化とユーザー要求の高度化により，図.1 に示す通りソフトウェアによる自動車の制御は高機能化し，プログラムの規模が加速度的に増加し，複雑度も増してきている．

今後も ITS などの普及により，さらに高度な制御を組み込んだシステム開発が求められ，ソフトウェアの開発への負担は今後も増大することが予想される．

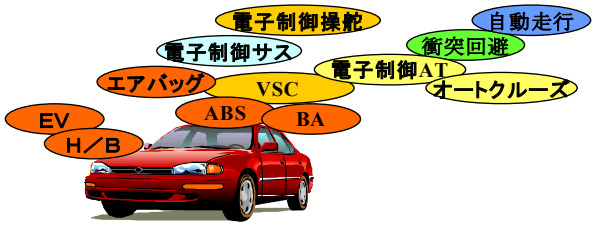

図 1 車載用システムの例

このような中でも，プログラムの不具合が車の安全

に関わるため，プログラム品質の低下は許されない．そこで，当社ではソフトウェアの開発部署とは異なる第三者評価部門による製品の最終テストを行っているが，多くの時間を費やしている．

3. ソフト開発の流れ

当社におけるソフトウェア開発の流れを図 2 に示す．

図 2 ソフトウェア開発の流れ

開発ステップでは机上でのアルゴリズム検討の後，仕様書を作成し，プログラムを作成する．

評価ステップではまずコードの机上チェックを行いＥＣＵに擬似的なアクチュエータ・センサを接続して動作チェックを行うベンチ評価の後，実際の車両に搭載して車載評価を行い，出荷される．

4. ソフト開発改善の取り組み

プログラムの複雑化への対応として，当社でもソフト品質の向上と開発のスピードアップを目的として，ソフト開発方法の改善を検討中である．

ＵＭＬの導入や制御モデルによる仕様の検討，コードの自動生成も進めており，特にモデルについては the MathWorks 社の MATLAB/Simulink をカスタマイズしてコードと仕様書を自動生成するオリジナルのツール Schetch/M(スケッチエム)[(1)(2)(3)(4)]を開発している．

5. 現状の問題点

しかし，このような活動でも現状ではいくつかの課題が存在する．

①開発されるソフトは既存ソフトに変更を加えた物が多いため，既存の開発方法を大きく変えることは，品質確保の面からリスクが大きい．

②製品の仕様・性能は人間の官能による部分が大きく，プログラムを修正しながら仕様を作り込むことになる．この結果，仕様書とプログラムは同時にできあがるため，単純なウォーターフローやＶ字開発にはならない．

③ツールの導入コストに対して効果が未知数である．

このような状況から，当面は日本語による仕様表記された仕様書が使われ続けることが予想される．

6. 方策

そこで，できるだけ現状の開発スタイル，仕様書を変えないまま開発の効率化を進める方法を検討した．

ベンチ評価工程で発見される不具合を調査した結果，仕様書とプログラム内容の不整合やラベル名の間違いといった机上で十分なチェックを行えば，事前に発見できるものが多くあった．

この不具合を事前にチェックする方法として仕様書に着目した．

仕様書はワードプロセッサソフトを用いて作成されているが，その内容については人間が読んで判断するもので，電子化されていることによるメリットは，きれいな字で印刷できるだけにすぎない．

これを意味を持ったデータにすることによって，プログラムのチェックに活用できると考えた．

文章を意味を持ったデータにする手段として，ＸＭＬ(eXtended Markup Language)がある．

しかし一般的なＸＭＬエディタはユーザーがＸＭＬの構造や仕組みを理解している必要があり操作も複雑である．

そこでテキストエディタの感覚でＸＭＬを作成し，内容の自動チェックができるツール Prio(プライオ)を独自開発した．

7. ツールの紹介

Prio を用いることで，仕様に対応したプログラムの情報を入力すれば，その矛盾点を発見することが可能である．

図 3 に示すとおりユーザーは意味付けを行いたい文字列を範囲指定し，右クリックメニューで開かれた表の中にその文字列がプログラム内のどのラベルに相当するかといったプログラム情報を設定する．

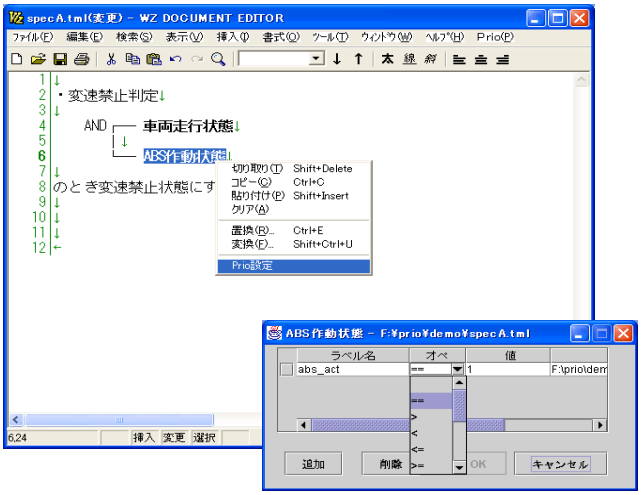

図 3 Prio によるデータ入力画面

このようにして対応するすべての状態に対してプログラム情報を設定する.設定された情報は図 4 に示した形式の XML(XHTML))[5][6][7]形式の文書として保存される.

```
><prio:condition↓
   ><prio:caption↓
      >ABS作動状態</prio:caption↓
   ><prio:refer↓

      lvar = "abs_act"↓
      op = "=="↓
      rvar = "1"↓
      src = "F:&#165;prio&#165;demo&#165;srcA.c"↓
      line = "7"↓
      col = "30"↓

   /></prio:condition↓
><prio:text↓
   ><br↓
   /><br↓
   />のとき</prio:text↓
```

図 4 XML ファイルのデータ

その後，図 5 に示すチェック機能により，整合性をチェックしたいファイル群を指定すると，不整合部分を指摘した HTML ファイルが生成される.

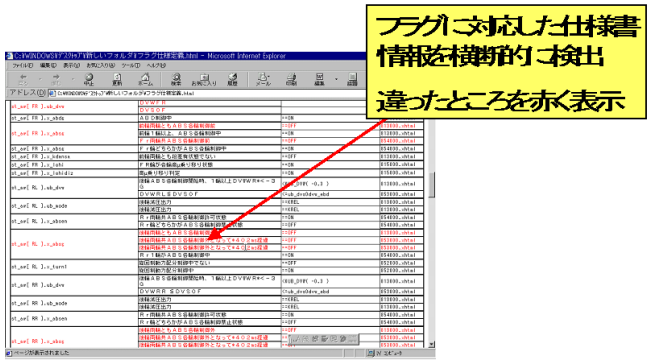

図 5　データチェック画面

8. 事例紹介

Prio を用いて，テキスト形式に変換した仕様書の特定の状態をあらわす語句に対して，対応するプログラム情報（どのラベルがどのような状態を表すか）を設定していく.

すべてのデータを入力後，全体のデータの整合性を自動的にチェックすることにより，例えばモジュール A では「車両走行状態」を ”vhicle_runnnig = 1 ” と設定し，モジュール B では「車両走行状態」を” car_running = 1” と設定していると言った不具合を事前に発見することができる．以下に発見可能な不具合の事例を示す.

8.1. 仕様書の条件がプログラムに抜けている例

仕様書の条件がプログラムに抜けている例を図 6 に示す．データ設定が完了した文字列はボールド表示されるので，プログラム情報のデータ入力を完了した時点でボールド表示されていない文字列は，その仕様に該当するコードがプログラム上に存在しないと気づくことができる.

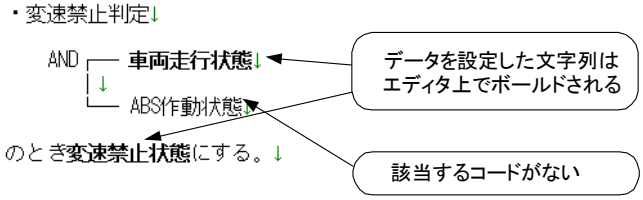

図 6 エディタによるボールド表示

8.2. プログラムの条件が仕様書に抜けている例

プログラムの条件が仕様書に抜けている例を図 7 に示す．データ入力完了後に，入力データのチェック機能により,ソース内の設定済文字列を赤色で表示する.この時点で赤色で表示されていないコード部分は，そのコードに該当する仕様が仕様書上に存在しないと気づくことができる.

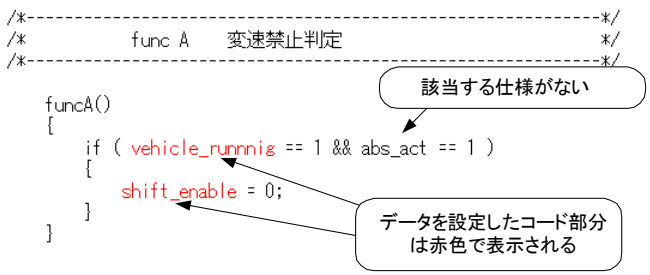

図 7 入力データのコードチェック機能

8.3. 仕様書の記述が間違った例

仕様書の記述が間違った例を図 8 に示す．データ入力完了後に，入力データのチェック機能により，すべてのプログラム上のラベルに対してどのような仕様が定義されているかを一覧表示できる．このとき，同じラベルに対して異なった仕様を定義したラベルを赤色で表示する．この時点で赤色で表示されているラベルは，その仕様書の仕様記述が他の仕様記述と異なった表現をしているか，間違った設定をしていると気づくことができる．

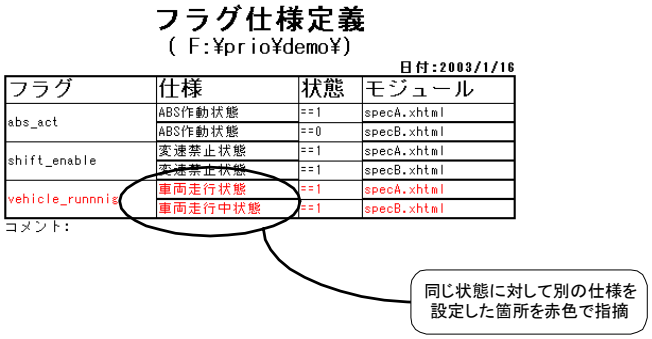

図 8 仕様記述チェック機能

8.4. プログラムの記述が間違った例

プログラムの記述が間違った例を図 9 に示す．データ入力完了後に，入力データのチェック機能により，すべてのプログラム上の仕様に対してどのようなラベルが定義されているかを一覧表示できる．このとき，同じ仕様に対して異なったラベルを定義した仕様を赤色で表示する．この時点で赤色で表示されている仕様は,間違ったラベルをしていると気づくことができる．

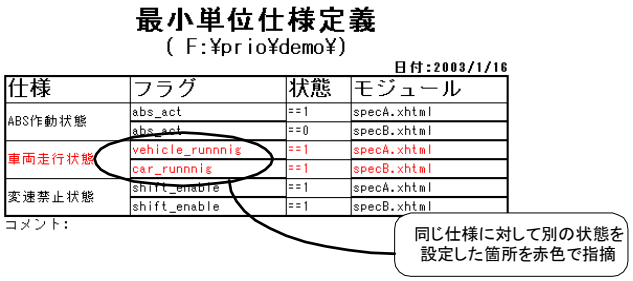

図 9 プログラム仕様チェック機能

9. 効果の説明

机上チェック時にチェックシートによるチェックを行っているが,これを補完するものとして活用した．

実際に導入した結果，図 10 に示す通り従来のベンチ評価の約 3 倍の効率でソフトウェアの不具合を発見することができた．

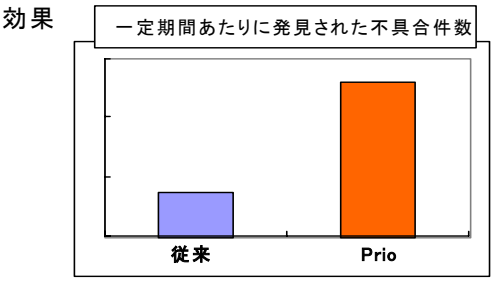

図 10 従来の評価方法との比較

Prio の開発において，テキスト表示には市販エディタである「WZ-Editor」を用い，「TX-C 言語」[8]により実装している.また，XML 処理部には Java の XML 用ライブラリを用い,チェック機構については XSL[9]を用いてほとんどの機能を実装したことにより短期間でかつ安価に開発できた．

10. チェック済みデータの活用

Prio で入力／チェックを完了した XML データは，仕様書とプログラムを 1 対 1 で結びつけた情報を持っている．この情報は開発部署のチェック工程だけでなく，その後の工程でも利用することにより，多くの作業が効率化できる．現在は，レビュー工程および第三者評価部門による最終テスト工程に Prio のデータを活用する仕組みを構築し，これを活用中である．

10.1. レビュー用資料の生成

レビュー工程では図 11 に示す通り各製品のレビュースタイルに合わせたスタイルシートを作成し，出力した HTML ファイルを活用している．

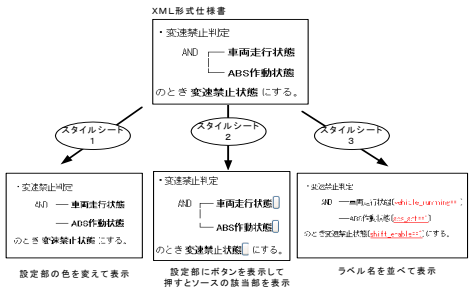

図 11 スタイルシートによるレビュー資料生成

10.2. テスト用データベースとのリンク

最終テスト工程では，ベンチまたはシミュレーションで仕様書通り動作しているか RAM 値の動きを確認している．しかし，テストごとにチェックする RAM 値は異なり，テスターは毎回どの RAM 値をチェックするのかを設定する必要があった．

図 12 に示す通り Prio で作成した XML データとテスト用データベースとをリンクさせることにより，テスターは HTML 形式で表示された仕様書から計測する RAM 値のラベル名をドラッグ＆ドロップをするのみでテスト用設定作業を行えるようになった．

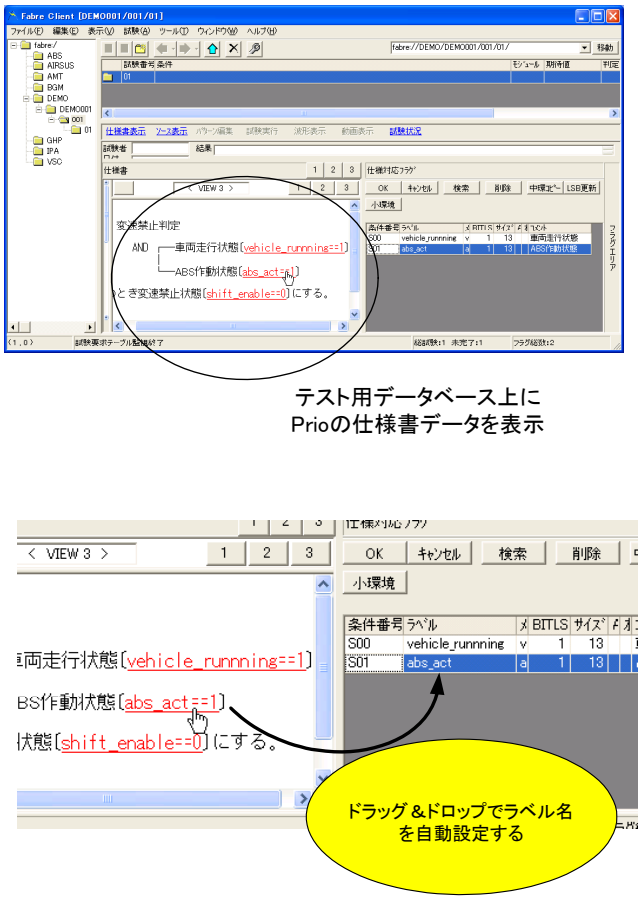

図 12 データベースとの連携動作

この仕組みによって計測すべき RAM 値の検索時間が大幅に短縮され，設定ミスもなくなった．

11. 今後の計画

Prio を導入することにより仕様書とプログラムとの不整合ミスは評価工程前に大幅に減少することができた．また，Prio によって得られた XML データを後工程で活用する仕組みを構築できた．

現在は，Prio で入力できないチャート，数式，図などの仕様を XML 化し，Prio と同様の効果を狙ったツールの検討を進めている．

文　献

[1] 稲森豊，和田錦一，手嶋茂晴, “ECU ソフトウエア仕様記述言語とプログラム自動生成”，自動車技術会 1996 年春季学術講演会

[2] 相馬秀茂，“「次世代ソフトウェア開発支援ツール」の開発と活用”，自動車技術会 2001 年春期大会 20015228

[3] サイバネットシステム，“自動車分野への応用を９目指した統合ソフトウェア開発環境の構築”，http://www.cybernet.co.jp/matlab/support/event/autoconf/auto_aisin.pdf

[4] サイバネットシステム，“シャシー系 ECU ソフトウェア設計への適用のため SIMULINK のカスタマイズ”，http://www.cybernet.co.jp/matlab/support/event/autoconf/auto_tytlabs.pdf

[5] “XML WORLD 第一弾” IDG ジャパン

[6] ジャストシステム，“一太郎 Ark で始める XML/XHTML” http://www.justsystem.co.jp/ark/genba/03.html

[7] 丸山宏，田村健人，浦本 直彦＝著／訳”“ＸＭＬとＪａｖａによるＷｅｂアプリケーション開発” ピアソン・エデュケーション社

[8] WZ エディタ ビレッジセンター http://www/ villagecenter.co.jp/soft/wz40/

[9] W3C XSL http://www.infoteria.com/jp/contents/xml-data/REC-xslt-19991116-jpn.htm

[10] 山田信幸，鈴木幹雄，坂守，“XML 形式の仕様書作成によるソフトウェア机上チェックの効率化”，ソフトウェアテストシンポジウム２００３，http://blues.se.uec.ac.jp/swtest/jasst03/yamada_doc.pdf