組込みシステムにおけるコード自動生成に関する研究
―生成プロセスとツールについての考察―

2009SE296 若山泰知
指導教員：野呂昌満

1　はじめに

1.1　背景

組込みシステムでは例外処理は重要な要素である．本研究では例外処理について焦点を当てる．

我々の研究室では E-AoSAS++[1] に基づいたアプリケーション開発について研究している．近年システムの開発期間の短縮等，開発労力の削減が求められてている．E-AoSAS++ に基づくシステム開発を行うことでこれらの問題を解決を試みている．

E-AoSAS++ に基づいたアプリケーション開発をする場合，コード生成ツールである「MDA[2] に基づくコード生成ツール」が利用可能である．コード自動生成より手作業でコードを記述する手間が軽減され，システムの開発期間の短縮等，開発労力が削減できる．

1.2　問題点

生成ツールの自動生成では限界が存在する．E-AoSAS++ に基づくシステム開発ではシステムを状態遷移機械の集合と規定されている．また状態遷移機械は遷移とそれに伴うアクション等で構成されている．そして，自動生成に UML を利用するが，E-AoSAS++ ではアクションコードに対応する自動生成の元になる記述が存在しない．よって，一般的にコードの自動生成には難しい問題が存在する．

また，自動生成するコードの元となる情報を変更するとアーキテクチャの変更が必要である．よってツールの再利用することができない．したがってツールを作りなおす必要がある．

1.3　目的

課題としてアクションコードに対応する記述が存在しない事について，我々は組込み開発で重要な要素である例外処理に焦点を当て問題の解決に取り組む．これにより，例外処理についてコードを自動生成することで，例外処理とそれに関連するアクションコードも一部生成可能であると考える．

目的は，既存の技術を応用することにより，自動生成可能な領域を拡大することである．これにより自動生成することへの優位性が向上が期待できる．そして例外処理を自動生成するには必要な情報を追加する必要がある．

1.4　アプローチ

JML[3] 記述の追加する事を想定している．それによって Java でコードを書くことに限定される．ただし，生成ツールが生成したコードに手を加える形式であるので，ツール自体に手を加える必要がない．

アクションのみにしぼる事で自動生成に条件が加わり，自動生成可能ではないかと考える．

JML 記述の条件からコードを自動生成する．コードに条件を追加する形で行えるので，コードの再利用が可能である．

2　背景技術

chapter 背景技術本研究で利用する背景技術を以下に示す．

2.1　E-AoSAS++

本研究室では組込みシステムのためのアスペクト指向ソフトウェアアーキテクチャスタイルである E-AoSAS++ を提案している．

E-AoSAS++(Aspect-oriented Software Architecture Style for Embedded systems) はシステムを平行に協調動作する状態遷移機械 (以下，CSTM) の集合として定義されている．CSTM は遷移と遷移に伴うアクションで構成される．

E-AoSAS++ に基づく開発支援環境が整備されている．E-AoSAS++ に基づくアーキテクチャ記述からコードを自動生成を行うコード生成系が既に制作されている．コード生成系については 2.3 節に示す．

2.2　モデル駆動型アーキテクチャ

モデル駆動型アーキテクチャ (以下，MDA) とは国際標準団体である Object Management Group が 2001 年に発表したシステム開発手法である．MDA では UML 等のモデリング言語を用いてアプリケーションをモデル化した情報にもとづいてコードを自動生成する．MDA はプラットフォームに依存せずコードの自動生成をすることが可能である．

2.3　MDA に基づくコード生成ツール

本研究で用いられるコード生成ツールは E-AoSAS++ にアーキテクチャ記述を入力することでコードを生成するツールである．特に断りが無い限り，本論文では E-AoSAS++ に基づいたコードを出力する MDA に基づくコード生成ツールを MDA に基づくコード生成ツールと呼ぶことにする．本研究では入力するモデルとしてとして Astah 形式の UML[4] ファイル，出力として Java を用いる．

2.4 JML

本研究ではアクションコードの自動生成の入力として JML 記述を利用する．JML とは Java モジュールの動作の条件を指定する事を目的として用いられる言語である．本研究では JML を利用しアクションコードを自動生成する．

3 JML の構文評価

3.1 JML 記述から Java コードに変換の対応関係

JML 独自の演算子について Java で書き表す．これにより JML の演算子は変換可能であることが分かる．

JML Operator	Java Operator
A ==> B	!A || B
A <== B	!B || A
A <=!=> B	A != B
A <==> B	A == B

3.2 JML から Java への変換

ensure, require, invariant による条件の記述方法を示す．JML 記述と出力されるコードの対応関係を図 1 で示したように変換を行う．

図 1 左の JML 記述は，右の Java コードに変換される．これにより JML 記述から変換することができるといえる．

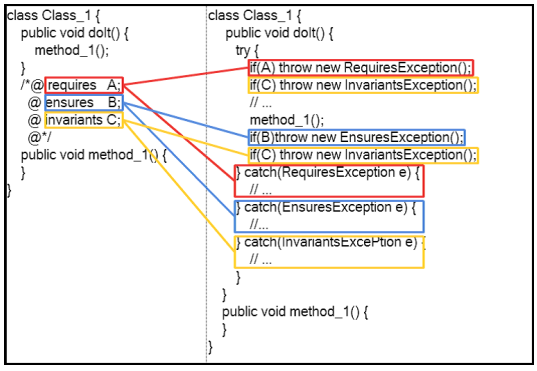

図 1 E-AoSAS++ に基づく開発プロセス

4 JML 記述の一定のパターンからの自動生成

4.1 パターンの概要

パターン化することで手作業で書く部分を減らし自動生成の優位性を出す．

4.2 パターンによる自動生成の適用

パターン化の 1 つとして宣言の自動生成が考えられる．以下の例では，条件の値の宣言を自動生成している．

```
/* @ require obj. getX() > 0; @ */
public void method() {
}
```

上記の JML 記述は以下の Java コードに変換される．

```
public void method() {
  int x = obj. getX();
  // …中略
}
```

また変数名はメソッド名から命名する．

5 事例検証

実例検証として飛行船制御ソフトウェアに対して自動生成を行った．これによりコード生成によりアクションコードの一部が生成可能であることが分った．自動生成で出力されたコードは，手作業でプログラムを書く箇所が減少し，優位性があることが分った．

6 考察

JML から Java にコードを変換する方法を示した．これにより例外処理，パターン化した部分である宣言文が自動生成することが可能である．例外処理では事前条件，事後条件，不変条件に対しての条件による分岐を行う箇所を自動生成した．パターン化では条件に関連性のある値について変数宣言を自動う生成した．

本研究の自動生成の問題点としてコードのパターンを示すことが困難であることがあげられる．本研究ではパターンとして変数宣言部を生成したが，その他のパターンを発見することが今後の課題である．

また，ESC/Java といった定理証明をすることによる自動生成による精度の改善が可能であると考えられる．ESC/Java では JML 記述に Java コードが則しているか静的解析を用いて事前検証することで今以上の精度あがると考える．

今回の研究では JML を利用したが，代替手段として executable UML や Action Semantics が存在する．これらとは違い JML で記述することでツールの再設計が不要であり，コードの再利用が可能な優位な点が確認できた．

参考文献

[1] 加藤大地，蜂巣吉成，沢田篤史，野呂昌満，**“アスペクト指向に基づくソフトウェアアーキテクチャ文書化方式”**，“知的ソフトウェア工学研究会 (KBSE)．vol. 108，no. 449，pp. 44-60，2005.

[2] K. Anneke, W. Jos, and B. Wim, *Explained:The Model Driven Architecture: Practice And Promise*, Addison-Wesley 2003.

[3] G. Leavens, A. Baker, and C. Ruby: *“JML: A notation for detailed design, ”* In Behavioral Specifications of Businesses and Systems (H. Kilov, B. Rumpe, and I. Simmonds, editors), pp. 175—188, Kluwer Academic Publishers, Boston, 1999.

[4] Jacobson, Ivar; Booch, Grady; Rumbaugh, James. *“The Unified Software Development Process. Addison Wesley Longman. ”* 1998.