C++レビュー支援システム

2C－4

堀田勇次　中島哲　直田繁樹
(株)富士通研究所

1. はじめに

ソフトウェア品質を向上させるためには、レビューは不可欠なものである。一方、C++言語では特有のイディオム[2][3]が存在し、プログラムの理解を困難なものにしている。我々はこれまでパターンの自動適用ツール[1]を開発し、イディオムやデザインパターンの適用を自動化する試みを行った。そこで今回は、これらのイディオムやパターンの中で形式的にチェック可能なものを自動的にチェックし、自動適用ツールで正しいコードを生成することによって、レビュー作業を支援するシステムを開発した。このシステムでは合わせて品質評価用のメトリクスの計測も行い、総合的にレビュー作業を支援する。

本稿では、レビュー支援システムの概要と、その中のコーディング規約チェック部について述べる。

2.レビュー支援システム概要

本システムは、次のような構成となる。*図 1* はレビュー支援システム全体の構成を示す。

(1)入力解析部は入力 C++プログラムを構文解析し、解析した情報をプログラムオブジェクトモデル(以下 POM)という構造に格納する。POM の情報を元に各種のレビュー支援のためのツールを構築している

(2)メトリクス計測部は、POM の情報を元に各種メトリクス情報を生成する。

(3)規約チェック部は、POM の情報とコーディング規約を元に入力プログラムをチェックし、規約に反するものがある場合は結果を出力する。ここでの規約チェックは言語仕様レベルのエラーチェックではなく、ユーザ部門で規定したコーディング規約やイディオムに対する違反チェックである。

(4)コード生成部は、従来のパターン適用ツール[1]を用いたものであり、POM の情報／規約チェックの結果／コード生成ルールを元に、規約に沿ったコード例を生成する。

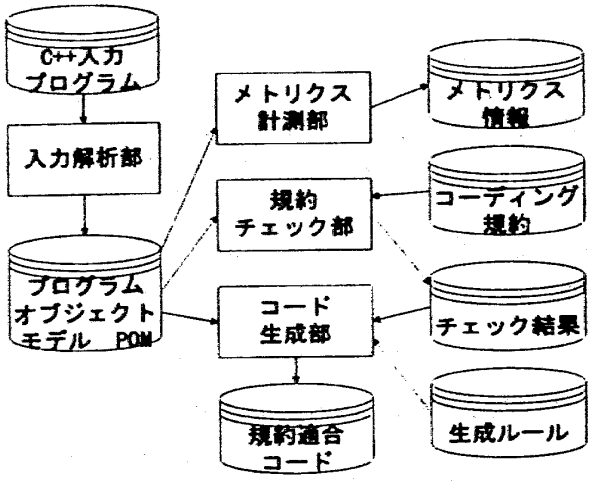

図1 レビュー支援システムの構成

以下では規約チェック部に焦点を当てて説明する。

3.規約チェック部

3.1.規約の指定

チェック部による規約適合チェックでは、チェックする対象とチェックの内容を組合わせて指定することで、規約を柔軟に設定することを可能にしている。

例えば、チェック対象としては*図 2* に挙げたものなどを用意する。チェック内容としては*図 3* に挙げたものなどを用意する。これらの対象とチェック内容を使って、*図 4* のように規約を指定する。

図 4 の第 1 の規約は「関数呼出しでオブジェクトの値渡しをしてはいけない」というものを表現している。このために対象として「メンバ関数」や「大域関数」でなく、「全ての関数」を指定している。

第 2 の規約は「データメンバにポインタ型を has-a 関係で持つクラスはデストラクタが存在しなければならない」というものを表現している。この

C++ Program Review Support System
Yuuji Hotta, Satoshi Nakashima, Shigeki Suguta
Fujitsu Laboratories, ltd.
4-1-1 Kamikodanaka Nakahara-ku, Kawasaki 211-8588, Japan

ため、and を使ってチェック内容の論理積を指定している。

```
データメンバ
メンバ関数
大域変数
全ての関数
クラス
全てのシンボル
```

図2　チェック対象

関数に関するもの	関数の仮引数でポインタ渡しチェック オブジェクトの値渡しチェック ...
クラスに関するもの	ポインタ型のデータメンバ存在チェック コピーコンストラクタなしチェック デストラクタなしチェック 代入演算子なしチェック ...
命名規約に関するもの	接頭文字列チェック ...

図3　チェック内容

```
全ての関数: オブジェクトの値渡しチェック
クラス: ポインタ型のデータメンバ存在チェック
        and has-aのデータメンバチェック
        and デストラクタなしチェック
```

図4　規約

3.2.規約チェック例

図5 に入力プログラム例を示す。このプログラムにはクラス A に char へのポインタ型のデータメンバのみが存在する。データメンバのコメント部分では、メンバを has-a 関係で持つことを指定している(生成ツール[1]で使用した方式)。

```
// file A.h
class A {
private:
    char *label; // !!has-a
};
```

図5 入力例

このプログラムに対して*図4* で示す規約のチェックを実行した場合、*図6*の出力が得られる。

```
ファイル A.h: 4行目
クラスAはポインタ型のデータメンバを持つ
かつ4行目
has-a関係のデータメンバを持つ
かつ2行目
クラスAはデストラクタを持たない
```

図6 出力例

*図5*の入力では、*図4*の第2の規約(ポインタ型の has-a 関係のメンバを持つクラスがデストラクタを持たない)に反しているため、*図6* の出力ではその具体的な内容の通知を行う。

3.3.生成部との結合

規約に沿ったコードの自動生成を行う場合は、*図7*のように規約を指定する。

*図7*の規約により、has-a 関係のポインタ型のメンバを持ちデストラクタを持たないクラスについては、そのメンバをデストラクタにより削除するコード例を生成する(*図8*)。規約中の「gen_destructor」は、コード生成部のために指定する生成ルール名である。

規約チェック部は、コード生成のための付加情報を伴う一時ファイルを出力し、コード生成部はそれらの情報と生成ルール等を元に、規約に適合したコードを生成する。

```
クラス: ポインタ型のデータメンバ存在チェック
        and has-aのデータメンバチェック
        and デストラクタなしチェック
        : gen_destructor
```

図7　生成の指定を含む規約

```
// file A_gen.h
class A {
public:
    ~A() {  // デストラクタ追加
        if (!label) delete [] label;
    }
private:
    char *label; // !!has-a
};
```

図8 コードの生成例

4.まとめ

コーディング規約への適合を自動チェックし、正しいコードを生成することでレビュー作業を支援するツールを実現した。

現在の試作では、規約チェック部とコード生成部との結合が、一部の単純な生成を除き未実現である。今後はこの機能を拡張し、適用範囲を広げる。

参考文献

[1] 直田, 堀田: デザインパターンの適用自動化手法, 情報処理学会研究報告 96-SE-112, pp. 41-48, 1996.

[2] S. Meyers: Effective C++. Addison-Wesley, 1992. 岩谷訳, C++の 50 の急所. ソフトバンク, 1993.

[3] J.O.Coplien: Advanced C++ Programming Style and Idioms. Addison-Wesley, 1992. 安村, 大谷, 湯原訳, C++プログラミングの筋と定石 1994.