組込みソフトウェアの性能要件充足のための コーディングパターンの考察

2009SE171 水野早也香
指導教員：野呂昌満

1　はじめに

近年，ユーザ要求の多様化などに伴い，ソフトウェアの大規模化や複雑化が著しい．組込みシステムの開発においては，商品開発サイクルの短縮化により，さらなる生産性と信頼性の向上が求められている．大規模化したソフトウェアの開発期間を短縮する方法としてモデル変換による自動生成がある．また，複雑なソフトウェアはアスペクト指向技術を用いて構造を整理する．組込みシステムは，状態遷移機械の集合なので，オブジェクトとの親和性が高く，オブジェクト指向で開発することが適している．組込みシステムにおいて，性能要件を充足することは重要であるしかし，オブジェクト指向で開発した際に，性能要件を充足するためのコードは複数のオブジェクトに散らばって存在する．散在しているそれらのコードをアスペクトとして分離・モジュール化する．

一般的に前提，課題，解決策の関係を表現したものをパターンという．本研究では，組込みシステムを前提とし，性能要件の充足に関する課題とその解決法の場合の，コードの雛型をコーディングパターンと呼ぶ．

ソフトウェアは一般的に要求定義，仕様記述，設計，実装の順で開発される．しかし，ソフトウェア開発の要求，仕様，設計，コード間の関係は多対多となる．それぞれの関係を整理する必要がある．

本研究の目的は，設計のパターンととコードのパターン間の関係を整理することである．仕様と設計の関係からコーディングパターンを特定する．

本研究では，仕様と設計の関係からコーディングパターンを定義し，カタログとして提案する．

ISO9126[3] の品質特性と対応するコーディングパターンのカタログを提案する．品質特性と対応した GoF デザインパターン [2] を，関係を整理している Quality Map[1] により選択する．デザインパターンのサンプルコードを基にコーディングパターンを定義する．また，品質特性を複数考慮した場合のコーディングパターンも定義する．定義したコーディングパターンと品質特性の関係をカタログ化する．カタログより品質特性とデザインパターンを選択すれば，それに対応した性能要件を充足するコードが求まる．作成したカタログを組込みシステムに適用可能か事例検証をして考察し，評価する．

2　背景技術と問題

背景技術として ISO9126，デザインパターン，Qulaity Map について説明する．

ISO9126 は，品質の評価に関する国際規格である．要求と品質特性と照らし合わせることで，網羅的に確認して性能要件を定義することができる．組込みシステムでは信頼性，機能性などの品質特性が考慮されるべきである．

GoF デザインパターンはオブジェクト指向設計の頻出問題の解決方法である．性能要件に関する問題を解決するパターンも存在する．

Quality Map は品質特性とデザインパターンの関連を示したものである．品質特性同士のプラスとマイナスの影響の有無が示されている．

ソフトウェア開発の，要求，仕様，設計，コード間の関係は多対多の構造をとり，複雑に関連している．要求とコードの関係が，直接的には不明瞭なので，それぞれの関係を整理する必要がある．

3　コーディングパターン

Quality Map より，品質特性と対応したデザインパターンを基に，コーディングパターンを定義する．その後，事例検証を用いたカタログの利用方法を述べる．

3.1　コーディングパターンの定義

GoF デザインパターンのサンプルコードを基にしたコーディングパターンの定義を示す．サンプルコードとはデザインパターンの一般例であり，デザインパターンにおいて主な構造となる部分が記述されている．本研究ではコーディングパターンとして，メッセージ通信，条件分岐処理，その他に必要なコードを定義する．

障害許容性を例としてコーディングパターンを定義する．Quality Map より障害許容性には Chain of responsibility, Command, Facade, Strategy パターンが対応する事が分かる．Chain of responsibility パターンについて示す．

障害許容性と関連する性能要件の充足に必要な部分を，コーディングパターンとして定義したものの一部を以下に示す．

```
   resolve(value)
IF
   done(value)
   next != null
ELSE IF
   Handler(value)
ELSE
   fail(value)
END-IF
```

Handler クラス内で，通常処理を行えず後続オブジェクトがある場合は処理を送信し，後続オブジェクトが無けれ

ば例外処理オブジェクトに送信する処理を抽出した．

3.2 複数の品質特性を考慮した定義

複数の品質特性を考慮した性能要件について考察する必要がある．他の品質特性についても，品質特性それぞれとの組み合わせにより，違うコーディングパターンが定義できると考える．

例えば障害許容性に関する性能要件は，同時に他の品質特性と関連する場合がある．品質副特性を同時に関連したコーディングパターンが作成可能か，表としてまとめる．障害許容性と他の品質副特性の関係の表を図 1 に示す．

	合目的性	正確性	相互運用性	セキュリティ	成熟性	回復性	理解性	習得性	運用性	魅力性	時間効率性	資源効率性	解析性	変更性	安定性	試験性	環境適応性	設置性	共存性	置換性
障害許容性	−	×	−	○	○	−	−	−	−	−	○	×	○	○	○	−	○	−	−	−

図 1 障害許容性と他の品質副特性の関係

3.3 事例検証

定義したコードについて事例検証を行なう．例として飛行船のセンサー取得システムについて考える．障害許容性を考慮した性能要件として，センサーが異常値を取得しても計算処理を続けることを定義する．使用するデザインパターンは，いづれかのオブジェクトで必ず処理をする構造である Chain of responsibility パターンを選択する．自動生成できた実行結果コードは以下のようになる．

```
public final void Handler(Value value){
  if(resolve(value)){
    done(value);
  }else if(next != null){
    next.Handler(value);
  }else{
    fail(value);
  }
}
```

自動生成したコードを基にした実装コードの構造は，センサーの値に障害が起った時に，処理オブジェクトの連鎖として後続オブジェクトがあるかどうかを確認する．ある場合は次の処理オブジェクトへと処理を渡し，無ければ例外処理をするというものになる．このコードは，どのような障害が起っても処理をするので，障害許容性に関する性能要件を充足しているといえる．

4 考察

4.1 コーディングパターンの妥当性の確認

事例検証により，定義したコーディングパターンを用いることで性能要件充足を考慮したコードの自動生成ができることが分かった．自動生成したコードを基にした実装コードの構造は，障害許容性に関する性能要件を充足しているといえる．よって定義したコーディングパターンは妥当であると考える．しかし，定義したコーディングパターンだけでは，必要なコードが足りないと考える．本研究では組込みシステムを対象としてコーディングパターンを定義したが，アプリケーションの前提を絞った場合に，他にも自動生成できるコードがあると考える．事例検証を重ねて，足りないコードを付け加える努力が必要である．

また，複数の品質特性に関連した性能要件を充足するコーディングパターンの定義が必要である．品質特性同士の影響を調べ，品質特性を同時に考慮できる性能要件について，全ての品質特性の組み合わせについてコーディングパターンを定義する必要がある．また，二つ目の品質特性の条件によって定義できるだけでなく，同じ品質特性とデザインパターンからコーディングパターンが複数定義できる場合もある．定義したコーディングパターンと品質特性とデザインパターンの関係を整理する必要がある．

4.2 先行研究との比較

三宅らの研究 [4] との比較を行う．この研究では頻出コードを自動抽出し，コーディングパターンとして定義する方法を提案している．本研究では性能要件を考慮してデザインパターンからコーディングパターンを定義している．性能要件を充足する点では本研究の方法の方が，性能要件の充足を考慮したコーディングパターンを定義に優位であると言える．

また先行研究はコーディングパターンを絞り込みのための手法の検討を今後の課題として挙げている．本研究は品質特性と対応したコーディングパターンを定義しているので，性能要件充足に関するコーディングパターンの整理ができている．

5 おわりに

本編では，Quality Map とデザインパターンから，品質特性と対応したコーディングパターンを定義した．また，コーディングパターンの有用性の評価を行なった．

今後の課題として，複数の品質特性に関連した性能要件を充足するコーディングパターンの定義がある．

参考文献

[1] A. Sawada, M. Noro, H. Chang, Y. Hachisu, and A. Yoshida, “A Design Map for Recording Precise Architecture Decisions,” *Proceedings of the 18th Asia-Pacific Software Engineering Conference*, pp. 298-305, 2011.

[2] E. Gamma, R. Helm, R Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[3] ISO/IEC 9126-1:2001, *Software Engineering - Product Quality - Part 1: Quality Model*, 2001.

[4] 三宅 達也, 石尾 隆, 谷口 考治, 井上 克郎, “メソッド呼び出しパターンとして現れる横断的関心事の検出,” 電子情報通信学会技術研究報告，Vol. 107，No. 99，SS2007-9，pp. 1-6，2007.