ユーザの実装逸脱度に基づくフレームワークの設計評価メトリクス

5ZC−5

黒田 隆一†, 小野 康一‡, 深澤 良彰†
†早稲田大学 理工学部, ‡日本IBM 東京基礎研究所
{kuroriu, onono, fukazawa}@fuka.info.waseda.ac.jp

1 はじめに

フレームワーク(FW)は、再利用を目的とした、アプリケーション(AP)の骨組みであり、クラス間に協調関係を持ったクラスライブラリである。FW利用者は個々のクラスではなく、ライブラリの全体を再利用してAPを構築する。AP要求に従い必要なクラスをカスタマイズし、そのまま使用できるクラスはサブクラス化せずに再利用する。クラス間の関係ごと再利用することになるので、FWの再利用は即設計の再利用となり、従来のオブジェクト指向ソフトウェア開発に比べ設計コストの大幅な削減を期待できる。

FWのカスタマイズは、フックメソッドと呼ばれる関数を他クラスとの協調関係を保つように実装することにより行なわれるが、そのためにはFW自体の理解が不可欠であり、一般にFWの理解は困難であるといわれる[1]。したがって、FW利用者は、FWを利用することで削減を期待できるAP開発コストとFW利用のために要求される教育コストとのトレードオフを慎重に検討する必要がある。

この教育コストの問題に対し、FW開発者は、理解しやすい、または理解が不十分でもある程度の保守性を保証できるFWをユーザに提供すべきであり、そのようなFWは品質が高いといえる。

本稿ではドメインに特化されたFWの品質として、「ゆるさ」を提案する。まず、FW開発者の意図と標準的実装について述べ、標準的実装からの逸脱のしやすさを表す尺度として「ゆるさ」を定義する。その後、AP業務に着目した「ゆるさ」の測定手法を定め、測定した「ゆるさ」の有効性を実例を用いて評価する。

2 本研究の概要

2.1 FWの「ゆるさ」

FW開発者は、FWを設計する際、その利用形態を考慮し、FWを用いたAPを構築する場合にどのような実装が行なわれるのが望ましいかという意図を必ず持っている。これをFW開発者の意図、また、意図に沿った実装を標準的な実装と呼ぶことにする。

開発者の意図はFWソースコード、API、仕様から読みとれることもあるが、FW内部に暗に示されていて外部からは分からなかったり、あるいは全く意図がFWに反映されていない場合もある。このようなFWを用いてAPを構築すると、非標準的な実装が行なわれてしまい、FWの保守やAP要求の変更に対処できなくなる恐れがある。

従って、非標準的実装に対する制約をFWの設計に含めることはAPの保守性の向上に有効である。

我々は開発者の意図から外れた実装を許してしまうようなFWは品質が低いと考え、FWの設計が開発者の意図からどれだけ乖離しているかを表す尺度を「ゆるさ」と定義する。本稿ではFWの「ゆるさ」を測定する方法を述べる。

2.2 前提

2.2.1 対象とするFW

以下では、特定ドメインに向けて開発されたFWを対象とする。また、そのFWを利用した複数のAPがあるとする。

2.2.2 FW開発者の意図

本研究ではFW開発者の意図として、次に挙げるようなメソッド呼出しの規則を考える。

- あるAPドメインにおいて定義される業務を行なうメソッドの指定
- メソッド呼出しに関する依存

この点においてFWがゆるい場合、ある業務を行なう際にFW開発者が標準的と考えるものから外れたメソッド呼出しを行なっても、APが正常に動作してしまう可能性がある。

3 「ゆるさ」の測定

FWにおける業務の大部分はオブジェクトの状態変化であると考えることができる。そこで本手法では対象

A Metric for the Framework Design Based on User's Deviation from Standard Implementation
Ryuichi Kuroda†, Kouichi Ono‡, Yoshiaki Fukazawa†
†Department of Computer & Information Science, School of Science & Engineering, Waseda University
‡Tokyo Research Laboratory, IBM Japan, Ltd.

とする業務をステートチャートとして表現し、ここから「ゆるさ」を求めることにする。

3.1 AP 業務に基づく「ゆるさ」測定の手法

本手法で使用するステートチャートのノードには対象業務を最も象徴する状態変化が起きるオブジェクト(キーオブジェクト)の状態を、アークには、FW のメソッド呼出しを描く。キーオブジェクトが 2 つのメソッドによって状態を遷移する場合、ステートチャートは図 1のように描けばよい。ある業務に関して開発者の意図する状態遷移が描かれたステートチャートを SS と呼ぶことにする。またその FW を用いた n 個の AP の実装についても、同じ業務に関するステートチャートを描き、これらを $AS_1, AS_2, \ldots AS_n$ とする。

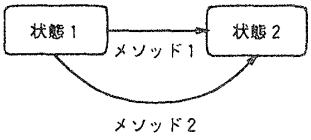

図 1: 業務表記ステートチャートの例

ステートチャート上でのその業務における FW の「ゆるさ」を i とし、以下の手順に従い、i を求める。

1. $1 \leq j \leq n$ の全ての j に対して SS と AS_j を比べ、不一致アーク数を m_j とする
2. $i = \frac{1}{(SSのアーク数)} \times \frac{1}{n} \sum m_j$ を計算する

3.2 FW ライフサイクル上の「ゆるさ」測定の意味

FW の「ゆるさ」を求めるという工程は、「ゆるさ」が小さくなるよう再設計することで FW をより洗練させるためのものであり、FW の成熟化プロセスの一環と考えることができる。

「ゆるさ」は FW の部分によって異なる値をとる。本手法では FW の業務毎の「ゆるさ」を測り、「ゆるさ」が大きな業務箇所から再設計を行なうものとする。

4 検証

ここでは、標準的実装からの逸脱のしやすさと、前節の提案手法で測定される「ゆるさ」との関係について検証する。検証に用いたのは図書館システムドメインの FW である。このドメインにおいて定義される主な業務には、図書及び利用者の登録・削除、貸出・返却、予約・予約取消、紛失・補充などがある。

この FW では貸出、予約、補充について実装が逸脱しやすくなるよう設計した。貸出業務では、標準的な貸出を行なうインタフェース `Rental()`、貸出の前処理を行なう関数 `PreRental()`、実際に図書の状態を「貸出中」にする `RentalDash()` などのメソッドが用意されており、本来なら仮想関数にするべきではない `Rental()` などのメソッドを仮想関数化することで、ユーザが改変可能な箇所を多く作っている。これにより、例えば、`Rental()` の中に `PreRental()` が行なうべき貸出の前処理を記述することで、`PreRental()` を実行しなくとも AP が正常動作する、といった非標準的な実装が可能になる。

図 2に各業務の「ゆるさ」の測定結果を示す。図に挙げられていない業務では非標準的実装は行なわれておらず、「ゆるさ」は 0 である。

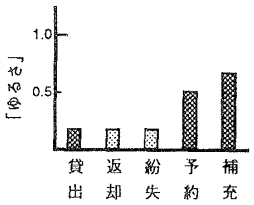

図 2: 各業務の「ゆるさ」(サンプル数 : 7)

逸脱しやすく設計された 3 つの業務の内、予約、補充業務では他の業務に比べ「ゆるさ」が大きいが、貸出業務は返却業務などと大差ない。この理由として、ユーザによる仮想関数のオーバーライドが可能であっても大部分のメソッドではデフォルトの実装が用いられること、また、返却など、逸脱しにくい業務の実装を行なうことによりユーザが教育効果を受け、貸出業務の標準的な実装方法を理解してしまうことが考えられる。

5 おわりに

本稿では FW の保守性を高めるためには標準的実装を促すように FW が設計される必要があることを述べ、設計の質を測るために「ゆるさ」という尺度を定義、その測定手法を提案した。また、本手法により測定される「ゆるさ」と AP への非標準的実装の可能性との間に相関があることを実例を用いて示した。

本手法により得た「ゆるさ」を FW にフィードバックさせれば、特定の業務に関する実装方法が FW 設計者の意図に近くなるような FW が得られることが期待できるが、ステートチャートを作成するコストは考慮しなければならない。この点に関しては、FW 設計者が AP 実装者にチャートを描くための情報を提供することによりコスト低減を望める。

現時点ではメソッド呼出し間の依存に関する「ゆるさ」を測ることはできない。今後はこの問題点に対する取り組みを進めようと考えている。

参考文献

[1] 小林隆志, 佐伯元司「フレームワークに基づいた変更支援法について」情報処理学会 ソフトウェア工学研究会報告, 122-16, pp.117-124, March 1998.