業務要件を用いたフレームワークの動的振る舞いに関するカスタマイズ方式の提案 *

4C－2

花舘 蔵之¶
NTT 情報通信研究所†
e-mail : hanadate@ isl. ntt . co. jp

1. はじめに

迅速かつ低コストで業務アプリケーション（AP）開発を実現するものとして、業務フレームワーク（FW）が注目されている。FW を利用するにあたって、FW 利用者（以下、利用者）は、まず、FW の「設計、実装」を理解し（FW の学習）、「これから作ろうとする AP（TAP：Target AP）」の業務分析を行う。そして、FW 内の「変更可能個所（HS：Hot-spot）」から実際に変更する「変更個所（CP：Customize-point）」を特定し変更する（カスタマイズ）。

利用者が「カスタマイズ」を行う際の問題点として、カスタマイズ前に行われる「TAP の業務分析」と「FW の学習」で得られた知識が大きく異なることが挙げられる[1]（問題点１）。これによって、「迅速かつ低コストで AP を構築すること」が損なわれる可能性がある。この問題を支援する技術として、ツールを利用する方法[1]や開発フェーズの作業をを利用フェーズで再利用する方法[2]がある。しかし、FW の動的振る舞いを直接カスタマイズすることは困難である[2]（問題点２）。

本稿では、FW の動的振る舞いに着目したカスタマイズ支援技術の中で、特にメソッドの内部記述の変更を FW の業務要件を用いて変更する方法を提案する。第 2 章では、FW の静的構造と動的振る舞いについて触れ、２つの問題点を分析し提案手法を述べる。第 3 章では、動的振る舞いのカスタマイズ方式を提案し、提案手法のカスタマイズ適用領域とその詳細について述べる。

2. 動的振る舞いのカスタマイズ

本稿で想定している「静的構造」は「クラス」「属性」「メソッド・インタフェース」といった「クラス図」で表現される。「動的振る舞い」は、あるクラスに属するオブジェクトがインスタンス化された際に、どのように動くかについて「メソッドの内部実装」や「メソッドの呼出手順」として記述される。

（問題１）について、HS の業務との関連から以下の分類ができる。

(a) 業務フローなど業務と関係した記述
(b) 設計パターン、実装言語制約、エラー処理など業務と無関係な記述

（問題２）について、その理由を以下のように考えた。。

・静的構造の方が動的振る舞いよりも観測しやすく[3]、FW の静的構造から CP を特定するのは容易である。
・FW の動的振る舞いから特定できる CP は、「TAP の構築過程において特定される CP」と「TAP の構築以前で特定される CP」と分散するため、特定困難である。

このことから、動的振る舞いのカスタマイズは、カスタマイズが行われるタイミングについて以下の２つの分類が考えられる。

(1) 既存のメソッドを組み合わせるカスタマイズ。
（TAP 構築過程におけるカスタマイズ）
(2) 既存のメソッドの内部を変更するカスタマイズ。
（TAP 構築以前におけるカスタマイズ）

(1)は FW に含まれないまったく新しい TAP 用のクラスとそのメソッドを、FW 内のオブジェクトのメソッドを利用して作成するカスタマイズである。(2)は FW 内のオブジェクトの動的振る舞いが TAP の業務と異なる際に、クラス継承を用いたメソッドのオーバーライドによって、異なる部分を修正するカスタマイズである。

以上から、表１の組み合わせが考えられる。

	(1)メソッド組み合わせ	(2)メソッド内部変更
(a)業務に関係	TAPの業務に依存	HSの提示
(b)業務に無関係	FWの利用制約	カスタマイズ不可

表 1 カスタマイズの分類とカスタマイズ手法

本稿では、(1)を行う過程で(2)を行う可能性が生じていると考え(2)の検討を行う。(2)において、クラス継承を用いてカスタマイズした場合、親クラスの実装をサブクラスから見る必要[4]がある。この結果、利用者は(a)、(b)の双方を考慮することになり、2.に挙げた問題点と同様の問題が生じる。

この問題を解決するために、(a)に関係した部分について業務要件として与えカスタマイズを行う。(b)は、業務要件との対応はとれないためカスタマイズは不可能である。ただし、(a)を変更することで(b)にも影響がでる可能性も十分に考えられ得る。この点について、「変更操作情報」として管理し(b)を含めた形で業務要件からカスタマイズを行うものとする（図１）。

---

* Customization of Framework Dynamic Behaviour taking into account of Business Policies

¶ Masayuki HANADATE

† NTT Information & Communication Systems Labs.

3. 提案手法

3.1. 提案手法の検討内容

本方式を検討するにあたって、以下が必要であると考えられる（図1）。

① 「設計、実装」からHSとして記述要素を抽出
② HSに対応した「業務要件」の記述を確立
③ 「設計、実装」と「業務要件」の変更操作の確立
④ ③の変更操作の対応関係の確立
⑤ 「変更操作情報」の管理方式の確立
⑥ ③～⑤を扱うカスタマイズツールの作成

このうち、本稿では①～②について検討した。

3.2. 「設計、実装」の記述要素と業務要件の記述

本章では、業務要件と対応させるメソッド内部にHSを与えるために、メソッド内部から記述要素を抽出し、抽出した記述要素に業務要件を対応づける。考慮した点は以下である。

1. 動的振る舞いの特徴として実行時に与えられた「値」によってオブジェクトがどのように振る舞うかが異なるため、業務要件の記述は、メソッド引数や条件式に与える「値」によって変える必要があると考えた。このため、「値」を記述要素として与え、業務要件を「値」との組み合わせで記述するとした。
2. 業務要件の記述内容は、「(A)どのような業務を行うのか？(what)」「(B)どのように業務を行うのか？(how)」に分類できる。しかし、C++やJavaといったオブジェクト指向プログラム言語では、(A)、(B)に相当する「メソッド呼び出し」や「制御手順」がすべて1つの実装コード内に記述されている。そのため、そのまま業務要件と対応させた場合、(A)と(B)が同時に業務要件として記述される。その結果、利用者は、業務要件で記述した実装コードの実行手順を単に目で追う結果となってしまう。この問題を避けるため、記述要素で「制御手順」を抜き出し業務手順と対応させ、(A)、(B)について明確に分離し、変更操作を別々に与えるとした。

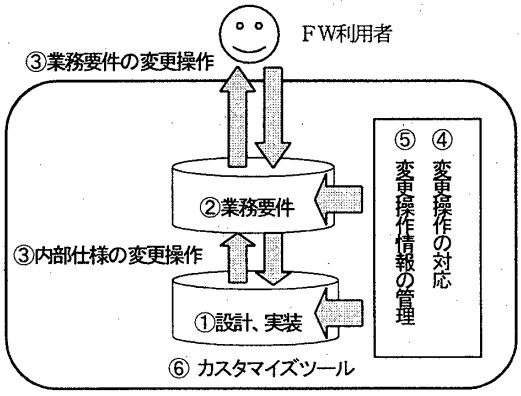

図1 提案手法

以上をふまえ、記述要素と業務要件を次のように与えた。

まず、記述要素として以下を抽出した。

・メソッド呼び出し（メソッド名＋引数）
あるメソッド内部で他のメソッド呼び出している要素。
・制御条件（条件文＋条件式）
ifやwhileのような条件式によって制御を行う要素。
・制御手順
「メソッド呼出し」「制御条件」の実行順序に関する要素。
・値
値は、FWの設計時点で特定できる「10」や「1999」といった具体的な「数値」として与えられている場合（定数）、また、FWの設計時点で特定できずに整数や年月日、契約者といった「型」や「クラス」として抽象化されて与えられている場合（変数）がある。

次に、上記の記述要素を組み合わせ業務要件は以下を与えた。

・業務機能 ＝「メソッド呼び出し」＋「値」
・業務条件 ＝「制御条件」＋「値」
・業務手順 ＝「制御手順」

以上の記述要素、業務要件に変更操作を与え、ツールを用いてカスタマイズを行う。

4. まとめと今後の検討課題

本稿では、FWの動的振る舞いにおけるカスタマイズ方式を提案し、FWを構成するオブジェクトのメソッド内部について記述要素を抽出し、そこから業務要件をあたえた。

本提案手法では、FW内部の業務に関係した動的振る舞いについてカスタマイズを行うことができる。業務に関連しない箇所の直接のカスタマイズは不可能であるが、影響がある場合は、ツールによってカスタマイズされる。また、メソッドを組み合わせるカスタマイズ（TAPの構築、FWのテスト等）は行うことはできない。

今後は、③～⑥について検討を行い、実際のカスタマイズへ適用、評価を行う。また、メソッドを組み合わせるカスタマイズについても検討を行う。

参考文献

[1] 大嶋,内川,関根,「業務要件に基づくビジネスオブジェクトカスタマイズツール」,情処学会第56回全国大会,p.p.125-126
[2] 名取,本位田,「オブジェクト指向フレームワーク利用手法の提案」,情処学会OO'97シンポジウム,p.p.100-107
[3] E.Gamma et al.,"Design Patterns",Addison-Wesley,1995