2B-1

ソースコードを記述しない アプリケーションフレームワークの提案

小池 晃弘†

株式会社日本レジストリサービス†

1. 背景

効率的にソフトウェアを改修するためにはドキュメントとソースコードの対応が取れていることが重要である。しかし、ドキュメントがソフトウェアの仕様を表していないことがある[1]。

ドキュメントとソースコードを同一ファイルに記述する手法[2]や別々のファイルとして作成されるドキュメントとソースコードの関連付けを行う手法[3-5]が存在するが、ドキュメントとソースコードの双方を適切に更新していかなければ記述内容の乖離が発生する。

本論文では、ドキュメントを元にしてソースコードを記述するのではなく、ドキュメントをソースコードとみなして実行することでソースコードとの乖離を抑制する手法を提案する。具体的には、プログラミング言語の知識を必要としない可読性の高い書式でドキュメントを規定し、そのドキュメントに従って動作するアプリケーションフレームワーク SLAF(Sourcecode-less Application Framework)を提案する。この手法によりソースコードの記述量が削減され、ドキュメントとソースコードの乖離が抑制される。

2. 提案手法

本論文で提案する SLAF はオンラインリアルタイム処理を対象としており、画面遷移の定義、ビジネスロジックの定義、永続化の定義でアプリケーションを構成する。

2.1. 実現方式

画面遷移は状態遷移表と画面テンプレートによって定義する。

A proposal of new application framework which does not need Source code

†Akihiro Koike

Japan Registry Services Co., Ltd.

ビジネスロジックは「処理」と「例外処理」を 1 行に記述することで定義する。「処理」にはその概要を示す簡潔な文字列とパラメータを記述し、「例外処理」には中断の有無とエラーコードを記述する。これに反復と分岐を加えた構文で曖昧さを排除した仕様を記述する。「処理」は別のビジネスロジックでその詳細を記述する。また「処理」は Java のメソッドにマッピングすることができ、最終的には全てのビジネスロジックが Java のコードとして実行される。

永続化はデータベースのテーブルと DTO(Data Transfer Object)のマッピングを定義し(図 2)、定義に従って SQL 文を生成し永続化を行う。

2.2. バッチ処理への適用

バッチ処理には画面遷移なく状態遷移表と画面テンプレートが不要となる。これらの代わりに並列に処理する単位を記述することでバッチ処理に適用可能なバッチ型 SLAF も提案する。

3. 評価

提案手法の有効性を確認するため、商用利用を前提としたインターネット・ドメイン名登録管理システム(図 3)開発への適用を行った。

制御文	処理	result	object	param	error break
#data_read	domain 情報取得	domain		SRSDB	
	非存在確認			domain	yes
#if	存在確認			In.domain.period	
	有効期限設定		In.domain	1	
#end					
#data_create	domain 情報登録			SRSDB	

図 1 シナリオの記述例

基本情報

ID	type
domain_read	db_read

データマッピング

table	column	class	param	field
domain	roid	EPPDomain	domain	roid
domain	name	EPPDomain	domain	name
domain	exdate	EPPDomain	domain	exDate

検索条件

table	column	class	param	field	field	operator	join
domain	name	String	#1			=	

図 2 永続化層のマッピング記述例

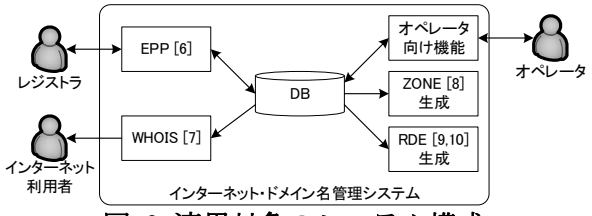

図 3 適用対象のシステム構成

表 1 ドキュメントとソースコードの比率

	行数(千行)	割合(%)
ドキュメント	60.1	87.2
ソースコード	8.8	12.8

表 2 従来のシステムとの記述量比較（千行）

	従来	提案手法
C	18.0	-
Java	-	0.4
ドキュメント	-	3.5
合計	18.0	3.9

表 3 WHOIS の処理時間（ミリ秒）

従来	提案手法
2	3

表 4 バッチ処理時間（秒）

Java	提案手法
168	201

3.1. ソースコード削減効果

提案手法を適用して開発したシステム全体のドキュメントとソースコードの割合を表 1に示す。Java のソースコードは全体の 12.8%に抑えることができた。

表 2は同等機能の従来システムが存在するWHOIS[7]サブシステムの記述を比較である。プログラム規模が大幅に縮小されていること、ドキュメントが 3500 行であるのに対してソースコードが 400 行程度に抑えられることが確認できた。

ソースコードを必要としたのはクライアントから受信したデータをフレームワークの内部形式へ変換する部分や DTO などであり、ビジネスロジックはソースコードを記述することなくシステムを実現できている。

3.2. 性能評価

オンラインリアルタイム処理におけるレスポンスタイムの比較を表 3に示す。比較に用いたのは前述した WHOIS サブシステムである。有効数字が小さくどの程度の差が生じているか判断できないが、提案手法で性能低下が見られる。

バッチ処理における処理時間の比較を表 4に示す。比較に用いたのはデータベースからZONE[8]生成を行うサブシステムである。ただし、従来のシステムを同一環境で評価できていないため同等処理を Java で記述した場合との比較を行った結果である。提案手法で 20%程度の性能低下が見られる。

4. まとめ

本論文ではドキュメントをソースコードとみなして実行する手法の提案とその評価について報告した。

提案手法を適用することで性能は劣化したが、ソースコードを大幅に削減可能でありドキュメントとソースコードの二重管理から開放されることが期待できる。

今後はリフレクションで実現しているビジネスロジックの実行をコード生成に変えるなどし、性能を向上する。

参考文献

[1] Andrew Forward, Timothy C. Lethbridge, “The relevance of software documentation, tools and technologies: a survey”, In Proceedings of the 2002 ACM symposium on Document engineering, pp.26-33, ACM Press, 2002.

[2] D.E.Knuth, 有澤誠 訳, 文芸的プログラミング, アスキー出版局, 1994.

[3] 後藤英斗, 大久保弘崇, 粕谷英人, 山本晋一郎, ”文脈に基づいたソースプログラムとドキュメント間の識別子対応付け手法”, 情報処理学会研究報告.ソフトウェア工学研究会報告, Vol.2005, No.29 (20050317) pp.41-48.

[4] 川平航介, 長田晃, 海谷治彦, 北澤直幸, 海尻賢二, “要求追加によるインパクトの分析に基づく組込みソフトウェア開発の効率化,” 情報処理学会研究報告.ソフトウェア工学研究会報告 2008(29), 17-24, 2008.

[5] 大場勝, 権藤克彦, ”アスペクト指向を用いたドキュメント整理法の提案”, 日本ソフトウェア科学会 第 7 回プログラミングおよび応用のシステムに関するワークショップ, 2004.

[6] S. Hollenbeck, "Extensible Provisioning Protocol (EPP)", 2009, <http://www.ietf.org/rfc/rfc5730.txt>.

[7] L. Daigle, "WHOIS Protocol Specification", 2004, <http:// www.ietf.org/rfc/rfc3912.txt>.

[8] P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", 1987, <http://www.ietf.org/rfc/rfc1035.txt>.

[9] F. Arias, G. Lozano, S. Noguchi, " Registry Data Escrow Specification draft-arias-noguchi-registry-data-escrow-06 ", 2013.

[10] F. Arias, G. Lozano, S. Noguchi, J. Gould, C. Thippeswamy, " Domain Name Registration Data (DNRD) Objects Mapping draft-arias-noguchi-dnrd-objects-mapping-05 ", 2013.