アスペクト指向的振舞インターフェース記述言語Moxaによるスケーラブルな仕様記述*

渡部卓雄† 橋本康範‡ 山田聖§
†‡ 東京工業大学・大学院情報理工学研究科・計算工学専攻
§ 産業技術総合研究所・情報セキュリティ研究センター

概要

MoxaはJavaを対象とした振る舞いインターフェース記述言語であり，表明アスペクトと呼ばれる機構の導入によって仕様記述のモジュール化を促進している．本稿ではMoxaの表明アスペクトを用いた仕様記述について概観し，さらに表明アスペクトを拡張して仕様記述のモジュール化をより促進する方法について議論する．

1 表明記述量の爆発

表明(assertion)はプログラムの各点における実行状態に関する仮定を記述するための機構であり，実行時検査によりテストやデバッグに役立てることの他に，(主にホーア論理を用いた) 静的検証のための検証条件やドキュメンテーションに利用される．Meyerが提唱した開発方法論である「契約による設計(Design by Contract, DbC)」では，クラスの外部仕様の記述に表明（各メソッドについての事前条件と事後条件およびクラス不変条件）を用いており，プログラミング言語Eiffelはその方法論を直接サポートする言語機構を持っている．JavaやC#を拡張して同様の機構を導入したものにLeavensらが設計したJava Modeling Language (JML)[2]やMicrosoft ResearchによるSpec#があり，それぞれ表明の実行時検査や表明にもとづく静的検査を行うためのツールが実装されている．

表明はプログラムが満たすべき性質を直接記述するものであり，表明検査や表明を用いた検証は，実際に稼働しているプログラムの信頼性や安全性の向上に寄与する．我々は過去に安全メール[3]の一部コンポーネントの仕様をJMLによって記述し，デバッグや安全性に関する性質の検証に用いたことがある．

一般に安全性の検証といった目的のためにはある程度詳細な仕様を記述する必要があり，プログラムの規模が大きくなるにつれて表明記述量が爆発的に増大する傾向がある．上で述べた安全メールのコンポーネントの場合，約2500行程度のあるコンポーネントについて検証に必要な表明記述量が(ソースコードとは別に)約3500行になったことがあった．そのような場合，複数のモジュールにまたがる条件が各モジュールの表明中に出現する傾向が非常に高くなる．このような傾向は，(我々が書いたもの以外にも) JMLによるいくつかの記述例に見られるが，それによって表明記述量とモジュール(とそれに付随する表明)間の依存性が増大し，結果として保守性が大きく低下してしまう．

2 Moxa

前節で述べた問題を解決するために，我々はJMLを拡張した仕様記述言語Moxaを設計し，そのためのツールを実装した[4]．Moxaを設計するにあたり，複数のプログラムモジュールに横断する表明記述に対処するために，アスペクト指向の考え方を用いた．MoxaではJMLと同様の仕様記述方法に加え，表明アスペクト(assertion aspect)という単位で仕様を記述するこ

*Scalable Specification in Aspect-Oriented Behavioral Interface Specification Language Moxa

†WATANABE Takuo (takuo@acm.org0, Department of Computer Science, Tokyo Institute of Technology
‡HASHIMOTO Yasunori (jubekun@psg.cs.titech.ac.jp), Department of Computer Science, Tokyo Institute of Technology
§YAMADA Kiyoshi (yamada-kiyoshi@aist.go.jp), Research Center for Information Security, National Institute of Advanced Industrial Science and Technology

表 1: プログラムの変更が表明記述に与える影響の比較

	JML		Moxa	
	Class1	Class2	Class1	Class2
変更箇所	42	53	6	4
変更した行数	190	149	54	40

とができる．AspectJ におけるアスペクトはモジュール間を横断するコードをモジュール化したものであるが，Moxa の表明アスペクトはモジュール間を横断する性質に関する表明をモジュール化したものである．

我々は現在までに，表明アスペクトを用いることで表明記述量を大幅に減少できることに加え，コードの変更が表明記述に与える影響範囲を限定できることを明らかにした．表 1 は実際に仕様記述を行ったコンポーネントの 2 つのクラスについて，その中で呼び出されるメソッドを変更したときの影響についてまとめたものである．表では変更の量のみが示されているが，変更箇所を表明アスペクトの織り込み先から容易に求めることができるため，実際の変更作業は非常に容易であった．

3 プロトコル記述の導入

旧来のクラス・メソッド単位の表明記述から表明アスペクトを抽出することで，複数のオブジェクトで構成されるコンポーネントの状態遷移に関する制約条件など，プログラムの抽象的な動作に関する仕様を明確にすることができる．ただし従来の論理式による（フラットな）表明記述ではそのような動作は把握しづらく，検証の際にも付帯条件が必要となり扱いづらい．そこで，我々はメソッド呼び出しシーケンスやコンポーネントの状態遷移を直接記述できるような表明アスペクトの拡張記述方式（プロトコル記述）を提案した．これは Cheon らによる JML へのプロトコル記述拡張 [1] を Moxa 用に拡張したものであるが，表明アスペクトを記述のベースとしたことにより，複数のクラスで構成されるコンポーネント内を横断する状態遷移に関する記述が可能になっている．

我々はさらに，プロトコル記述の合成の定式化を完成し，それにもとづいて表明アスペクトの織り込みに相当する操作を定義した．また，プロトコル記述によって導入される状態遷移系の性質をモデル検査器 SPIN を用いて検証する方法を提案した．以上によって，状態遷移を伴う複雑な仕様を簡潔に記述することが可能になり，仕様記述のスケーラビリティ向上に寄与することが期待できる．

4 今後の展望

Moxa による表明記述を DbC 的なものからより広範囲に拡張することを考える．例えば前節で述べたプロトコル記述では，対象となるメソッド起動までの実行履歴が表明の真偽に関わるような記述が可能になる．これは振る舞いサブタイプ関係を満たさなくなるため，もはや DbC の意味での表明ではなくなるが，表明の実行時検査を積極的に使う立場[1]として意義がある．今後はこの仕組みを Moxa に導入し，分散プログラムの仕様記述，テストへの応用を考えている．

参考文献

[1] Cheon, Y. & A. Perumandla, Specifying and Checking Method Call Sequences in JML, *Intl. Conf. on Software Engineering Research and Practice*, pp. 511–516, 2005.

[2] Leavens, G. T., A. L. Baker & C. Ruby, JML: A Notation for Detailed Design, *Behavioral Specifications for Businesses and Systems*, chap. 12, pp. 175–188, 1999.

[3] Shibayama, E., S. Hagihara, N. Kobayashi, S. Nishizaki, K. Taura & T. Watanabe, AnZenMail: A Secure and Certified E-mail System, *Software Security: Theories and Systems*, LNCS, 2609, pp. 201–216, 2003.

[4] Yamada, K. & T. Watanabe, An Aspect-Oriented Approach to Modular Behavioral Specifications, *Aspect-Based and Model-Based Separation of Concerns in Software Systems*, ENTCS, 163 (1), pp. 45–56, Sep., 2006.

[1] イリノイ大学の G. Rosu らによるモニタリング指向プログラミング (Monitoring-Oriented Programming, MOP) など