静的構造モデリングのためのアスペクト指向設計環境

添田 隆弘　横山 孝典　志田 晃一郎　兪 明連†

武蔵工業大学‡

1．はじめに

近年，組み込みソフトウェアの大規模・複雑化に伴い，オブジェクト指向による開発が普及しつつある．組み込みソフトウェア開発では，システム本来の機能である機能的側面の設計に加え，リアルタイム性や信頼性等の非機能的側面の設計が重要である．しかし，非機能的側面の処理は複数のクラスに共通する横断的関心事として現れることが多いため，オブジェクト指向のみでは非機能的側面をモジュール化できないという問題点がある．この問題を解決する手法としてアスペクト指向が注目されている．アスペクト指向を用いることにより，横断的関心事である非機能的側面をアスペクトとしてひとつのモジュールにまとめることが可能である．

効率のよいアスペクト指向による組み込みソフトウェアの開発を実現するためには，アスペクトを用いて非機能的側面をモデル化し，それをデザインパターンとしてライブラリ化できることが望ましい．しかし，アスペクト指向によるモデルをデザインパターン化し，それを織り込むための環境が整っていない．

我々は既に，非機能的側面として特にリアルタイム性に関する処理を対象にUML(Unified Modeling Language)[1]のシーケンス図とステートチャート図を用いてアスペクトを記述し，織り込みを行うモデルウィーバを開発している[2]．このモデルウィーバでは，組み込みソフトウェアでは多くの非機能的側面に分離されるため，織り込み後の図を機能的側面の図として，再び非機能的側面を織り込むことができるインクリメンタルな織り込みを可能とした．

しかし，組み込みソフトウェアの全ての非機能的側面をアスペクト記述するには，シーケンス図とステートチャート図のみでは十分ではない．例えば非機能的側面のひとつとして，周期が異なる複数のタスク間でデータの整合性を保持するためのバッファリングが挙げられる[3]．バッファリングは，異なる周期のタスクで算出されたデータを，あらかじめバッファと呼ばれるメモリ領域に保存し，それを読みだすことにより，同一周期のタスク内でのデータの整合性を保証する．このバッファはクラスとして織り込むことが適当と考えられる．そのため，クラス図を用いたアスペクトの記述とその織り込みが必要となる．

クラス図の織り込みが可能なモデルウィーバとしてUMLAUT[4]やAspectM[5]が提案されている．しかし，これらのモデルウィーバはインクリメンタルな織り込みができない，非機能的側面のモデルをパターンとして記述できない等の問題点があり，そのままでは組み込みソフトウェアの設計には対応できない．

そこで本研究の目的は，クラス図により非機能的側面のモデルをパターンとして記述し，そのパターンを織り込み可能な，組み込みソフトウェア向けのアスペクト指向設計環境を開発することである．具体的には，クラス図を用いて非機能的側面をアスペクトのパターンとして記述する方法を提案するとともに，その記述方法に対応したモデルウィーバを開発する．

2．クラス図によるアスペクトの記述

機能的側面の例として，車間距離制御システムにおけるスロットル開度(ThrottleOpening)を算出するクラス図を図1に示す．エンジントルク(EngineToruque)はエンジン回転数(EngineRevoultion)の値を読み出して算出される．そして，スロットル開度の算出ではエンジントルクの値とそのエンジントルクの算出に使用したエンジン回転数の値を読み出して用いる．これら3つの値を計算する処理を同一タスクで実行する場合は問題ないが，例えば，エンジン回転数を計算する処理を10ms周期のタスク，エンジントルクとスロットル開度を計算する処理を20ms周期のタスクで実行する場合，後者のタスクでエンジントルクの計算を行った直後に，前者のタスクによるプリエンプションが発生し，エンジントルクの計算に用いたエンジン回転数の値と，スロットル開度の計算に用いたエンジン回転数の値が異なり，データの整合性がとれなくなるという問題が発生する．

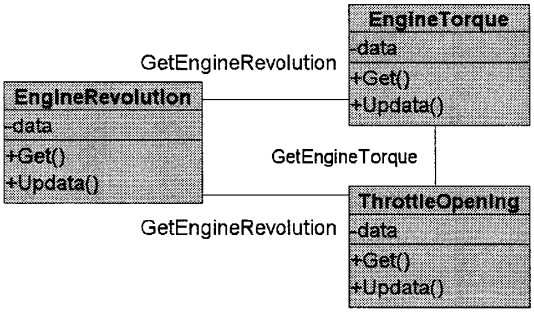

図1　機能的側面を表すクラス図

この問題を解決する手法のひとつにエンジン回転数の値をバッファリングするという方法がある．エンジントルクとスロットル開度を算出するタスクの先頭でエンジン回転数の値をバッファに記憶しておき，そのタスクの計算では，バッファからエンジン回転数の値を読みだして使用することにより，データの整合性を保持できる．バッファリングを本研究で提案するアスペクト記述でモ

An Aspect-Oriented Design Environment for Static Structure

† Soeda, Takahiro, Yokoyama , Takanori, Sida, kouitiro and Yoo, Myungryun

‡ Musashi Institute of Technology

デル化したものを図２に示す．図２に示すように，アスペクトをパッケージといわれる要素で囲み，そのパッケージ名に<<aspect>>と記述することでアスペクトであることを宣言する．次にポイントカットとして，〈ClassA, ClassB, ClassC, RelationA, RelationB〉 (EngineRevolution, EngineToruque, ThrottleOpening, GetEngineRevolution, GetEngineRevolution) という形式で記述する．この記述は ClassA, ClassB, ClassC, RelationA, RelationB を機能的側面の EngineRevolution, EngineToruque, ThrottleOpening, GetEngineRevolution, GetEngineRevolution に対応付けることを意味する．そして，この対応付けされた要素と繰り込む要素の関係をクラス図で記述する．図２では ClassA_buffer というクラスと buffering という関連を織り込むことをモデル化している．すなわち，このクラス図はバッファリングのためのアスペクトのパターンを表現している．このように本手法では，非機能的側面をパターンとしてモデル記述できる．

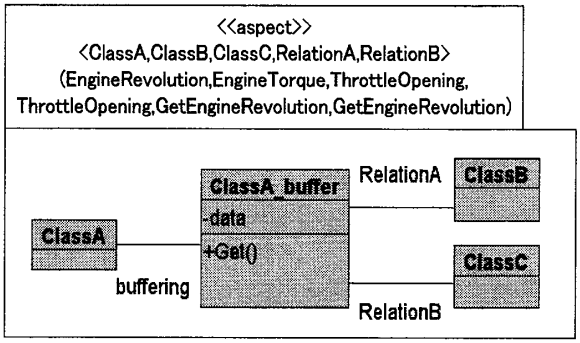

図２ アスペクトの記述

３．モデルウィーバ

開発したモデルウィーバは，機能的側面のクラス図とアスペクト記述の XMI(XML Metadata Interchange)ファイルを入力とし，提案したアスペクトの記述方法に基づき解釈し，織り込みを行った後，織り込み後のクラス図の XMI ファイルを出力する．

図１のクラス図と図２のアスペクト(パターン)をモデルウィーバに入力することで，織り込み後のクラス図として，スロットル開度の算出時にエンジン回転数のバッファリングを行う図３のクラス図を得ることができる．

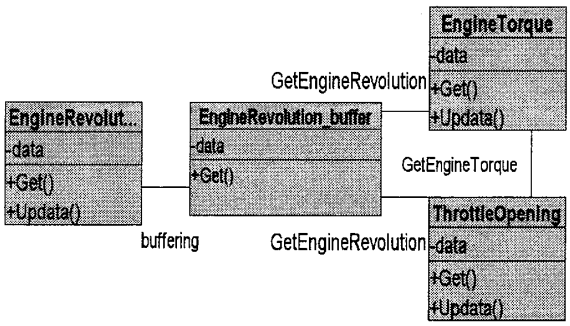

図３ 織り込み後

４．関連研究との比較

本研究とクラス図の織り込みが可能なモデルウィーバである UMLAUT[4]とアスペクト指向モデリング言語 AspectM[5]の比較を行う．

UMLAUT は，ポイントカットを指定したクラス図とアドバイスとして記述されたステートチャート図を入力とし，織り込み後のソースコードとクラス図を出力する．UMLAUT では，織り込み後のクラス図を再び入力として使うためにはポイントカットを指定しなければならない．そのため，インクリメンタルな織り込みをする場合の開発効率が低下する．それに対して，本研究で作成したモデルウィーバでは，織り込み後のクラス図を変更することなく，入力として使うことが可能であるため，開発効率を下げることなく，インクリメンタルな織り込みが可能である．

AspectM では，記述したアスペクトを元にクラス図 の変更を行うことができる．AspectM は MDA(Model Driven Architecture)のためのモデルコンパイラを対象としているため，クラス図の要素をアスペクトとして扱い，各々の要素の織り込みを行う．しかし，要素単位の記述であるため，本研究が目的としているような，非機能的側面を複数の要素を関連づけたパターンとして記述し，それを織り込むことには適していない．

５．おわりに

本研究では，組み込みソフトウェアを対象に，非機能的側面をクラス図によりモデル記述し，織り込むことが可能なアスペクト指向設計環境を開発した．

今後の課題としては，組込みソフトウェアの非機能的側面をアスペクトのパターンとしてライブラリ化することで組み込みソフトウェアの開発を支援し，本モデルウィーバの有用性を増すことである．

謝辞

本研究の一部は科研費 20500037 の助成を受けたものである．

参考文献

[1] Object Management Group, Unified Modeling Language Specification Version 2.1.1, 2007, http://www.uml.org/

[2] 柳館雄太，山形晃人，横山孝典，モデルレベルでの織り込みを実現するアスペクト指向設計環境，情報処理学会研究報告，2007-SE-155, pp.17-24, 2007.

[3] 吉村健太郎，宮崎泰三，横山孝典，オブジェクト指向組み込み制御システムのモデルベース開発法，情報処理学会論文誌 vol.46, No6, pp.1436-1445, 2005.

[4] Ho, W.-M., Jezequel, J.-M., Pennaneac' h, F. and Plouzeau, N., UMLAUT: A Framework for Weaving UML-based Aspect-Oriented Designs, Proc of 33rd International Conference on Technology of Object-Oriented Languages, pp.324-334, 2000.

[5] 鵜林尚靖，佐野慎治，前野雄作，村上聡，片峯恵一，橋本正明，玉井哲雄，アスペクト指向に基づく拡張可能な MDA コンパイラ，情報処理学会組込みソフトウェアシンポジウム，pp.104-107, 2004.