4M-01

リアルタイムOSファミリ開発のためのアスペクト指向プログラミング

山内　健司†　　兪　明連†　横山　孝典†

東京都市大学†

1. はじめに

組み込み制御システムは様々な用途で使用されるため，要求されるリアルタイム OS の機能も多様である．しかし，リソース消費量の制約が大きい組み込み制御システムにおいて，それらすべての機能を持つリアルタイム OS を搭載するのは困難であることから，それぞれのアプリケーションに応じて必要最小限の機能のみを持つリアルタイム OS を搭載することが望ましい．このため，単一のリアルタイム OS ではなく，アプリケーションに応じて必要な機能のみを持つリアルタイム OS のメンバを選択することができるリアルタイム OS ファミリが求められている．

そこで我々は，ベースとなる OS に対してアスペクト指向プログラミング[1]を用いて機能の追加や変更を行うことで，リアルタイム OS をファミリ化する研究を行ってきた[2]．アスペクト指向プログラミングを用いることによって，ソースコードを直接修正することなく，機能を追加・変更することが可能になる．また追加・変更箇所を分離してアスペクトとして記述して管理できるため，リアルタイム OS ファミリのソースコードの保守性を向上できる．

しかし，これまで使用してきた C 言語ベースのアスペクト指向言語 AspeCt-oriented C(ACC)[3]は，オブジェクト指向言語ベースのアスペクト指向言語とは異なり，継承等の機能がないため，アスペクト間の重複記述が増加し，管理が複雑化してしまうという問題があった．

そこで本研究では，重複記述がなく可読性の高いテンプレート記述を可能とする C 言語ベースのアスペクト記述法を提案するとともに，それに対応する処理系を開発する．そして実際にリアルタイム OS ファミリに適用し，重複の少ないアスペクト記述が実現できることを示す．

2. アスペクト指向言語及び処理系

2.1 方針

本研究では，ACC の文法を拡張して，複数のアスペクトに共通する部分をテンプレートとして記述可能とする．そしてテンプレートアスペクトを元に，各アスペクト固有の部分のみを実装アスペクトとして記述することで，重複記述を削減する．

言語処理系については，まったく新規の処理系を開発すると保守に多くの工数を要するため，テンプレートアスペクト及び実装アスペクトを記述したファイルをオリジナルの ACC ソースファイルに変換するトランスレータを開発する．そして，変換した ACC ソースファイルを ACC 言語処理系に入力してアスペクトの織り込みを行う．これにより，従来の ACC によるカスタマイズと同じ方法で，リアルタイム OS ファミリを実現できる．

- <アスペクト>::= <ACCアスペクト>|<テンプレートアスペクト>|<実装アスペクト>
- <テンプレートアスペクト>::= template '<'<パラメータ名の並び>'>'
  aspect <識別子>{<パラメータ付きACCアスペクト>}
  [macro { <マクロ定義> }]
- <パラメータの並び>::= <パラメータ>|<パラメータ><パラメータの並び>
- <実装アスペクト>::= bind '<' <パラメータ値の並び>'>'
  aspect <識別子>{<ACCアスペクト>}
  [macro { <マクロ定義> }]
- <パラメータ値の並び>::= <パラメータ値>|<パラメータ値><パラメータ値の並び>

<パラメータ名>及び<パラメータ値>は文字列
<パラメータ付きACCアスペクト>は識別子またはその一部が<パラメータ>を含む<ACCアスペクト>
[]は省略可能を表す

図 1 構文規則

2.2 アスペクト記述法

提案するアスペクト指向言語の構文規則を図 1 に示す．テンプレート記述はパラメータを含むことができ，実装アスペクトでそれらの具体的なパラメータ値を指定する．図 1 に記載した以外の構文規則は ACC と同一である．

図 2 にテンプレートアスペクトの記述例を示す．この例では，アスペクトの名前は Template_Func_Aspect で，SYSTEMCALL，OS，MSG，RET，sOS の 5 つのパラメータが用いられている．3 行目から 12 行目までが複数のアスペクトに共通する処理の記述である．そして 14 行目以降の macro 内でマクロ関数の宣言を行っている．

図 3 に実装アスペクトの記述例を示す．この例では図 2 の Template_Func_Aspect に対応した実装アスペクトを記述している．1 行目でベースとしているテンプレート名とそこで宣言されていた 5 つのパラメータのパラメータ値を指定することにより，テンプレートアスペクトの記述のうちパラメータのみを置き換えたものが実装アスペクトが意味するものとなる．2 行目で実装アスペクトのアスペクト名を記述し，追加するアスペクト記述がある場合はそれを記述する．この例では追加する記述はない． 5 行目からは，テンプレートアスペクトで宣言されたマクロ関数に対して，この実装アスペクトにおける処理内容を記述している．

Aspect-Oriented Programming for Developing a Real-Time Operating System Family

†Kenji Yamauchi, Myungryun Yoo and Takanori Yokoyama, Tokyo City University

```
template <SYSTEMCALL, OS, MSG, RET, sOS>
aspect Template_Func_Aspect {
        after(): set(StatusType task_location) &&
        infile("SYSTEMCALL##_common_aspect.c") {
        if (tsk_location == CALL_##OS) {
            OS##_SET_FLG()
            request_sending##sOS##_service(glbtskid_store,
                                    OSServiceId_##SYSTEMCALL, MSG);
            RET = (UINT8)(get##sOS##_return value());
            AFTER_SENDING_MSG()
        }
    }
}
macro {
 AFTER_SENDING_MSG();
MULTI_SET_FLG();
DIST_SET_FLG() {
         global_syscall_flag = TRUE;
        }
}
```

図 2　テンプレートアスペクト

```
bind Template_Func_Aspect <ActivateTask, MULTI, 0xFFFFFFFF, ercd,_multi>
aspect dist_multi_ActivateTask{
}
macro {
        AFTER_SENDING_MSG() {
         if (ercd != E_OK) {
           _errorhook_par1.tskid = TSKID(tskid_store);
           call_errorhook(ercd, OSServiceId_ActivateTask);
         }
    }
}
```

図 3　実装アスペクト

2.3 言語処理系

開発するトランスレータは，テンプレートアスペクトを入力し，テンプレートの記述内容を C のマクロ記述に変換したヘッダファイルを出力する．また実装アスペクトを入力し，テンプレートマクロを含む ACC アスペクトのファイルを出力する．

図 4 に本トランスレータを用いたカスタマイズの流れを示す．トランスレータの出力ファイルを C プリプロセッサに通すことで，テンプレートマクロが展開され，従来と同じ ACC アスペクト記述が得られる．ACC コンパイラは最初に C プリプロセッサを通すため，トランスレータが出力したファイルをリアルタイム OS のソースファイルと共に ACC コンパイラに入力することで，カスタマイズしたリアルタイム OS コードを得ることができる．

3. 評価

提案したアスペクト指向言語により，どの程度アスペクト記述量を削減できたかを評価するため，本研究室でこれまで開発した OSEK OS[4]準拠のリアルタイム OS である TOPPERS/ATK1[5]を並列・分散リアルタイム OS にカスタマイズするためのアスペクトを，本研究で開発したアスペクト指向言語とオリジナルの ACC を用いて記述し，それらのファイル数，総行数及び総文字数を比較する．表 1 に示すように，重複記述をなくすことで，総行数は 3/4 に，総文字数は 7 割程度に削減できた．
実行性能については，提案したテンプレートアスペクトは従来と同じ ACC の記述に変換されるため，両者に差はない．

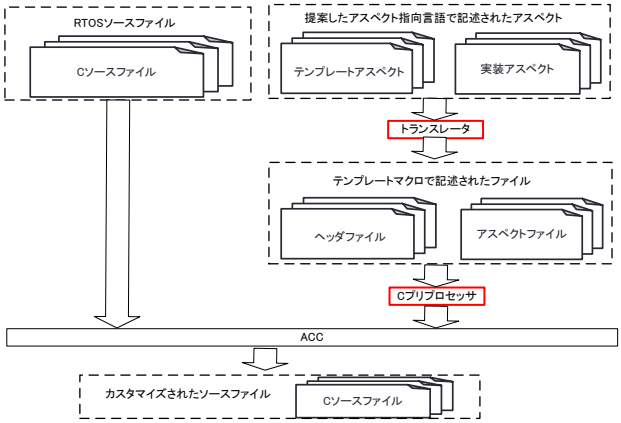

図 4　トランスレータを用いたカスタマイズの流れ

表 1　評価結果

アスペクト	本手法			ACC		
	ファイル数	行数	文字数	ファイル数	行数	文字数
dist_multi_Template	1	20	401			
com_Template	1	36	710			
dist_ActivateTask	1	11	216	1	12	340
dist_ChainTAsk	1	13	246	1	15	371
dist_GetEvent	1	3	86	1	8	246
dist_GetTaskState	1	3	91	1	8	237
dist_SetEvent	1	3	83	1	8	236
multi_ActivateTask	1	11	229	1	11	329
multi_ChainTAsk	1	13	259	1	14	360
multi_GetEvent	1	3	99	1	7	235
multi_GetTaskState	1	3	104	1	7	226
multi_SetEvent	1	3	96	1	7	225
com_ActivateTask	1	17	418	1	32	719
com_ChainTask	1	30	621	1	45	970
com_GetEvent	1	8	327	1	26	677
com_GetTaskState	1	6	318	1	25	672
com_SetEvent	1	14	476	1	32	818
合計	17	197	4780	15	257	6661

4. おわりに

重複記述を削減できるテンプレートを用いた C ベースアスペクト指向言語及び処理系を開発し，その有効性を確認した．今後，本処理系を用いて，より多機能のリアルタイム OS ファミリを実現する予定である．

謝辞

本研究で使用した TOPPERS/ATK1 の開発者と ACC の開発者に感謝する．本研究は JSPS 科研費 JP18K11225 と JP15K00084 の助成を受けたものである．

参考文献

[1]Kicazales, G. et al.: Aspect-Oriented Programming, Proceeding of the 11th European Conference on Object-Oriented Programming, pp.220-242 (1997)
[2]原田祐輔，阿部一樹，兪明連，横山孝典:アスペクト指向プログラミングによるリアルタイム OS スケジューラのカスタマイズ，情報処理学会論文誌 Vol.57,No.8,pp.1752-1764 (2016)
[3]Gong, M. Z. C. and Jacobsen, H. A.: Systems Development with AspeCt-oriented C (ACC), Connections 2007 (ECE Graduate Symposium, University of Toronto) Talk 5.6 (2007).
[4]OSEK/VDX:Operating System, Version 2.2.3 (2005)
[5]TOPPERS/ATK1: https://www.toppers.jp