C-029

アスペクト指向プログラミングによるリアルタイム OS 間でのアプリケーションの移植

Aspects for Improving the Portability of Application Programs between Real-time Operating Systems

稲本 太郎 † 兪 明連 † 横山 孝典 †
Taro Inamoto Myungryun Yoo Takanori Yokoyama

1.はじめに

組込み制御システムでは様々なリアルタイム OS が用いられているが，開発効率や再利用性向上のため，異なるリアルタイム OS 間で容易にアプリケーションを移植できる技術が求められている．

アプリケーションを移植するには API (Application Programming Interface) の呼び出しを移植先のリアルタイム OS 向けに書き換える必要がある．ただし，同じ機能でも OS により呼び出す API のインタフェース仕様が異なるうえ，ひとつの API の機能を複数の API の組み合わせで実現しなければならないこともあり，人手による書き換え作業が必要になる．ソースコード中の API の呼び出しを人手で直接修正せずに移植することができれば，ソフトウェアの再利用性を向上させることができる．

ソースコードを直接書き換えずに処理の置き換えや追加を行う方法としてアスペクト指向プログラミング [1] がある．アスペクトは置き換える処理内容を記述したアドバイスとその織り込み箇所を示すポイントカットから構成され，アスペクト指向言語処理系を通すことで，アドバイスに記述した処理を織り込んだアプリケーションを作成できる．

我々は，アスペクト指向プログラミングにより API の呼び出しを置き換えることで，アプリケーションの移植を容易化する研究に着手した．具体的には，OSEK OS[2] と μITRON[3] の自動車制御向けプロファイルを対象に，API の呼び出しを置き換えるアスペクトを開発し，ライブラリとして提供することが目的である．本論文では，OS の機能のうち，これまで検討してきたタスク管理機能とイベント制御機能に関するアスペクトについて述べる．

2.OS 間での API の対応付け

OSEK OS と μITRON 自動車制御用プロファイルで規定されているタスク管理とイベント制御の API の対応付けを表 1 に示す．同一機能を持つ API については置き換えルールをアスペクトで記述することで移植できる可能性がある．ここで，OSEK OS の自タスク終了処理とタスク起動処理を行う API である ChainTask() に直接対応する μITRON の API はないが，自タスク終了処理を行う ext_tsk() とタスク起動処理を行う act_tsk() の組み合わせで実現できる．両 OS 間で対応のない API は比較的使用頻度の低いものであり，対応する API のみの置き換えでも，ある程度の有用性が期待できると考えている．

表 1: OSEK OS と μITRON の API 対応付け

機能 \ API		OSEK OS	µITRON
タスク管理	指定タスクの起動	ActivateTask	act_tsk
	自タスクの終了	TerminateTask	ext_tsk
	自タスクを終了し指定タスクを起動	ChainTask	act_tsk + ext_tsk
	タスクの強制終了	対応無し	ter_tsk
	タスク起動要求のキャンセル	対応無し	can_act
	タスク優先度の変更	対応無し	chg_pri
	タスク優先度の参照	対応無し	get_pri
	再スケジューリング	Schedule	対応無し
	実行状態のタスクIDの取得	GetTaskID	対応無し
	指定タスクの状態を取得	GetTaskState	対応無し
イベント制御	イベントの設定	SetEvent	set_flg
	イベント待ちに遷移	WaitEvent	wai_flg
	イベントのクリア	ClearEvent	clr_flg
	イベント状況の取得	GetEvent	対応無し

なお，タスクやイベントの宣言については，OSEK OS では OIL 記述，μITRON では静的 API で記述するため，今後，OIL 記述と静的 API の記述を変換するツールを開発する予定である．

3.アスペクトの記述

3.1.記述方針

アスペクト指向言語として，C 言語を拡張した AspeCt-orientedC(ACC)[4] を使用する．call ポイントカットにより API の呼出し時を織り込み箇所に指定し，around アドバイスを用いて，一方の OS の API の呼び出しを，他方の OS の API の呼び出しに置き換えるアスペクトを記述する．これにより，API の呼び出しを置き換える．

3.2.タスク管理 API のためのアスペクト

タスク管理機能の場合の例として，指定タスクの起動するための μITRON の API である act_tsk() を OSEK

†東京都市大学

```
/* 置き換え対象のOSのAPIの型 */
#include “osek.h”
/* アスペクトで記述する元のOSのAPI仕様のパラメータ */
#include “itron_type.h”
/* タスクIDやエラーコードの変換用の関数のプロトタイプ宣言 */
#include “trans.h”

/* タスク管理機能 */
/* 指定タスクの起動 */
StatusType around (ID tskid) : call (ER act_tsk (ID)) && args (tskid) {

TaskType taskId;
StatusType errorCode;
ER ercd;
taskId = id_itron_to_osek(tskid);          /* タスクIDの変換 */
errorCode = ActivateTask(taskId);          /* API呼び出し */
ercd = er_osek_to_itron(errorCode);        /* エラーコードの変換 */

return ercd;
}
……
```

図 1: タスク起動 API を置き換えるアスペクト

```
/* 置き換え対象のOSのAPIの型 */
#include “osek.h”
/* アスペクトで記述する元のOSのAPI仕様のパラメータ */
#include “itron_type.h”
 /* タスクIDやエラーコードの変換用の関数のプロトタイプ宣言 */
#include “trans.h”

/* イベント制御機能 */
/* イベントのセット */
StatusType around( ID flgid,FLGPTN flgptn) :
       call (ER set_flg (ID,FLGPTN)) && args (flgid,flgptn ){

TaskType taskId;
EventMaskType mask;
StatusType errorCode;
ER ercd;
taskId = flgid_itron_to_osek(flgid);     /* フラグIDをタスクIDに変換 */
mask = event_itron_to_osek(flgptn);  /* イベント情報の変換 */
errorCode = SetEvent(taskId,mask);    /* API呼び出し */
ercd = er_osek_to_itron(errorCode);  /* エラーコード変換処理 */

return ercd;
}
……
```

図 2: イベントセット API を置き換えるアスペクト

OS の API である ActivateTask に置き換えるアスペクトを図 1 に示す．このアスペクトでは，call ポイントカットで “act_tsk() の呼び出し時” を指定し，タスク ID を表す引数 tskid を args ポイントカットを使用して参照する．around アドバイス内では，まずタスク ID 変換用の関数 tskid_itron_to_osek() により，μITRON でのタスク ID である tskid を OSEK OS でのタスク ID である taskId に変換する．そして OSEK OS の API である ActicateTask() の呼び出しを行う．最後にエラーコード変換処理を行う関数 er_osek_to_itron() により，OSEK OS のエラーコードを μITRON のエラーコードに変換し，リターンする．ただし，変換できるのは共通のエラーコードのみで，各 OS 独自のエラーコードについては現時点では対応していない．

3.3.イベント制御 API のためのアスペクト

イベント制御機能の例として，イベントをセットをする μITRON の API である set_flg() を，OSEK OS の API である SetEvent() に置き換えるアスペクトを図 2 に示す．OSEK OS のイベント制御の API ではタスク ID を指定するのに対し，μITRON のイベント制御の API はイベントフラグ ID を指定するため，タスク ID とイベントフラグ ID を対応付けるデータを作成し，記憶する．今回は手作業でデータを作成したが，今後は OIL 記述と静的 API を変換するツールにより自動生成する予定である．

このアスペクトでは call ポイントカットで “set_flg() の呼び出し時” を指定し，フラグ ID を表す引数 flgid と，イベント情報を表す flgptn を args ポイントカットを使用して参照する．around アドバイス内では，まず，フラグ ID をタスク ID に変換する関数である flgid_itron_to_osek() により変換処理を行う．そして変換で得たタスク ID を指定して，OSK OS の API である SetEvent() を呼び出す．最後に er_osek_to_itron() によりエラーコードを変換してリターンする．

なお，μITRON のイベント待ちの API である wai_flg() には，イベント状態に対し AND で待つ機能があるが、OSEK OS の API である WaitEvent() にはない．そこで WaitEvent() を繰り返し呼び出すことで AND 待ちを実現するが，実行効率は低下する．

4.適用実験と考察

今回対象とした OSEK OS と μITRON の API を用いて記述したテスト用アプリケーションを作成し，OSEK OS として TOPPERS/ATK1[5]，μITRON として TOPPERS/JSP[5] を用いてアスペクト適用後の動作確認を行った．OIL 記述と静的 API の記述については，今回は人手で変換した．タスク管理機能はほぼ制約なく置き換えを行うことができたが，イベント制御機能については前述のように実行効率が低下することがある．

5.おわりに

OSEK OS と μITRON のタスク管理機能とイベント制御機能に関する API の呼び出しを置き換えるアスペクトを提案した．今後は，排他制御機能等，残された API に関するアスペクトを提案するとともに，OIL 記述と静的 API の記述を変換するツールを開発することで，有用性のある開発支援環境を実現したいと考えている．

参考文献

[1] Kiczales, G et al., Aspect-Oriented Programming, Proc. of ECOOP ’ 97, pp. 220-242, 1997.

[2] OSEK/VDX: Operating System Version2.2.3, Feburary 17th, 2005.

[3] 坂村健監修，高田広章編:μITRON4.0 仕様 Ver. 4.02.00, トロン教会，2004.

[4] AspeCt-oriented C，https://sites.google.com/a/gapp.msrg.utoronto.ca/aspectc/

[5] TOPPERS プロジェクト，http://www.toppers.jp/