ソフトウェアテストシンポジウム2010
JaSST'10: Japan Symposium on Software Testing 2010

組込みリアルタイムOSのAPIテストの実施

2010年1月28日

名古屋大学大学院 情報科学研究科
附属組込みシステム研究センター
共同研究員　鴫原 一人

アジェンダ

オープンソースのリアルタイムOS(RTOS)のテストスイートを開発・実施した結果と，得られた知見について述べる

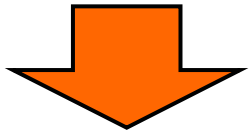
1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

組込みシステム開発におけるオープンソースの利用

オープンソースの利用

- ライセンス料金が不要
- スタンダード化による信頼性
- 自由にカスタマイズ可能
- 自らバグを解決できる

オープンソースを製品として使用
するためには品質の確保が重要



・検証作業を利用する企業で独自に実施する
・オープンソースの検証を行うベンダへ依頼する
・公開されているテストスイートを使用する

組込みシステムにおいても
オープンソースの利用が進んでいる

オープンソースの組込み向けRTOS(TOPPERS)

TOPPERSプロジェクト

TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

http://www.toppers.jp/

- TOPPERSカーネルを開発/公開しているNPO法人
- ITRON仕様のRTOS，ミドルウェア等を開発

  (ITRON：日本の組込みシステムで使用されているOSのデファクトスタンダード)
- オープンソースで公開
- 実製品にも利用されている

応用製品の例

TOPPERS新世代カーネル

- 信頼性、安全性、ソフトウェアポータビリティを向上させるため ITRON仕様を改良
- 改良した仕様は**“TOPPERS 新世代カーネル統合仕様書”**に明記している
- シングルプロセッサ対応**ASPカーネル**
- マルチプロセッサ対応**FMPカーネル**

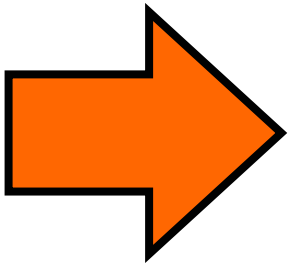
TOPPERS新世代カーネルのテストに関する問題点

包括的なテストは未実施

- RTOSのテストは規模が膨大で工数が大きい
- 大学では仕様検討/性能評価などの研究を中心としている
- RTOSを実製品へ組み込む際，利用者側でテストを行っている

マルチプロセッサ環境

- マルチプロセッサ対応RTOSのテスト開発に対する経験が少ない
- テスト開発/実施の工数がシングルプロセッサより肥大化する
- プロセッサ間の実行タイミングに依存したテストを考慮する必要がある



名古屋大学 組込みシステム研究センター(NCES)が
2009年度よりコンソーシアム型の研究組織を立ち上げ、
上記の問題を解決するためのテストスイートの開発を行っている

開発成果は一定期間後にオープン化する

アジェンダ

1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

コンソーシアム型研究組織の参加企業と体制

TOPPERS新世代カーネルの利用や，マルチプロセッサ対応RTOSに関する知見を得たい複数の企業・団体が参加

参加企業(五十音順)

三洋電機株式会社
株式会社デジタルクラフト
株式会社東芝 セミコンダクター社
日本電気通信システム株式会社
富士ソフト株式会社
有限会社松浦商事
宮城県産業技術総合センター
株式会社ルネサステクノロジ

体制

- 名古屋大学へ常駐：4名
- 自社内での作業：4名
- オブザーバ：3社

→月1回のミーティングに参加

2009年度の実施内容

RTOSに対するテストの分類

(A) API(静的API含む)に着目したテスト → <2009年度実施>
全APIが仕様通りに
動作することをテストする

(B) 処理単位に着目したテスト
(C) ターゲット依存部単独でのテスト
(D) 割込み禁止区間に注目したテスト
(E) タイマ割込み処理のテスト
(F) ロック区間に注目したテスト
(G) スピンロック中の割込みのテスト
(H) マイグレーションに関するテスト

2010年度以降
実施予定

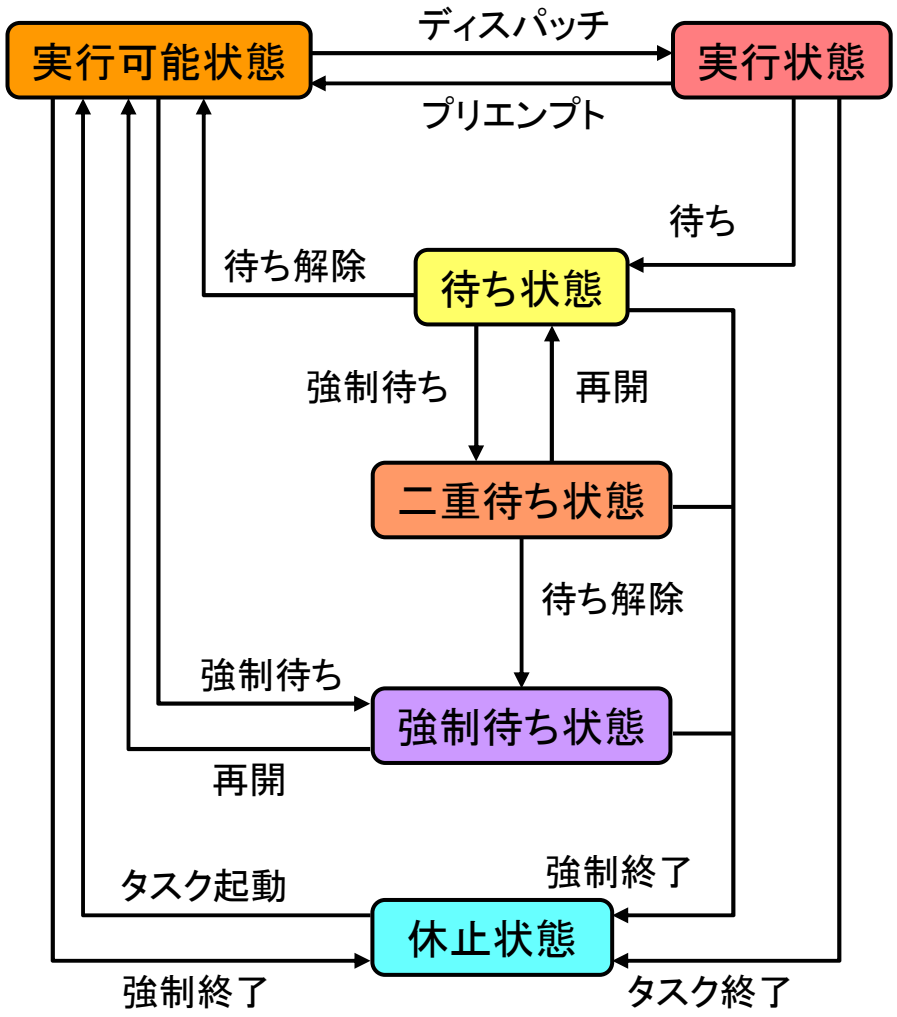
マルチプロセッサ対応FMPカーネルの前段階として，
シングルプロセッサ対応ASPカーネルに対してテストを行った

アジェンダ

1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

ASPカーネルの仕様

タスクの状態遷移



タスクの優先度に従った
プリエンプティブな
優先度ベースのスケジューリング

実行可能状態のタスクのうち，
最高優先度のタスクが**実行状態**
となる

カーネルのシステム状態として，
ディスパッチ許可状態と**禁止状態**
があり，**ディスパッチ禁止状態**では
優先度の高いタスクが起動しても
ディスパッチしない

各APIの発行によりタスクの状態が遷移する(APIに対称性がある)

ASPカーネルのAPI種別

機能名	概要	API数
タスク管理	タスクの起動や終了、優先度変更など	8
タスク付属同期	タスクの起床、起床待ち、遅延など	10
タスク例外処理	タスク例外の許可、禁止、要求など	6
タスク状態参照	タスク状態の参照	1
セマフォ	セマフォ資源獲得、開放、初期化など	7
イベントフラグ	イベントフラグ設定、クリア、初期化など	8
データキュー	データ送信、受信、初期化など	11
優先度データキュー	優先度付きデータ送信、受信、初期化など	9
固定長メモリプール	固定長メモリプール獲得、開放、初期化など	6
メールボックス	メッセージ送信、受信、初期化など	6
アラームハンドラ	アラームハンドラ起動、終了、参照など	5
周期ハンドラ	周期ハンドラ起動、終了、参照	3
割込み管理	割込みの許可、禁止など	4
CPU 例外管理	CPU例外からのシステム状態参照	2
システム状態管理	ディスパッチ禁止などのシステム状態参照	16
システム時刻管理	システム時刻の参照	2
静的API	システムコンフィギュレーション用API	17
合計		**121**

全API(121個)のテストを行う

アジェンダ

1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

APIテスト概要

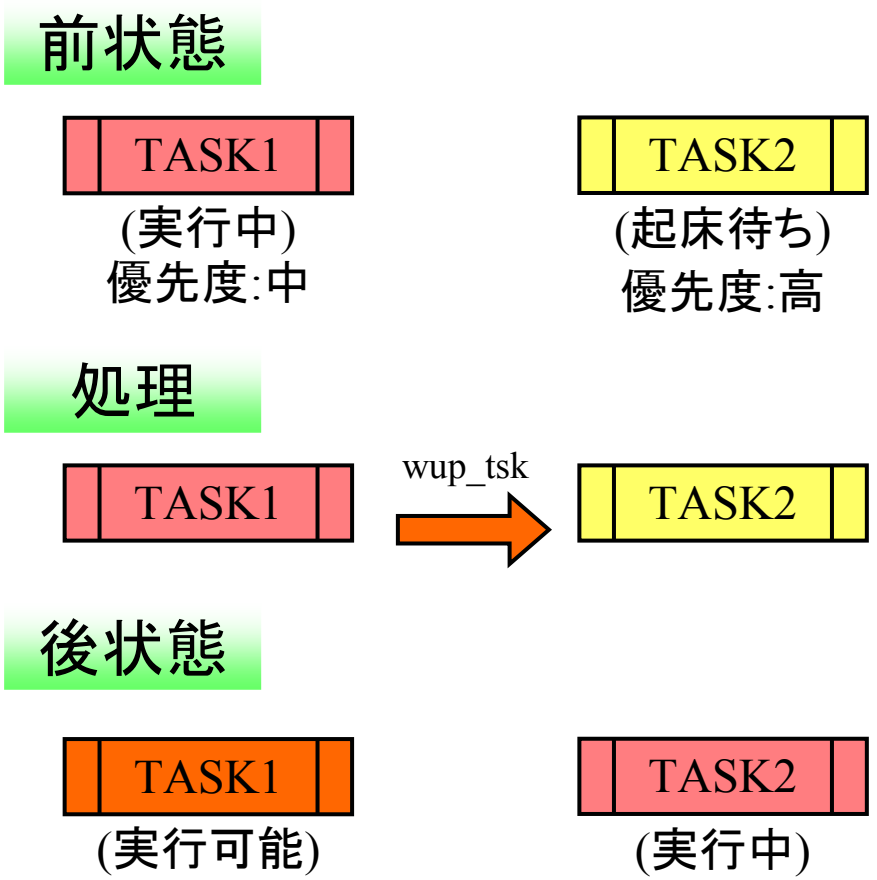
統合仕様書に基づいてAPI発行前後のシステム状態の変化を確認する

wup_tskの仕様抜粋

```
wup_tsk       タスクの起床
                :
【機能】
tskidで指定したタスク(対象タスク)を起床する.
対象タスクが起床待ち状態である場合には,
対象タスクが待ち解除される.
```

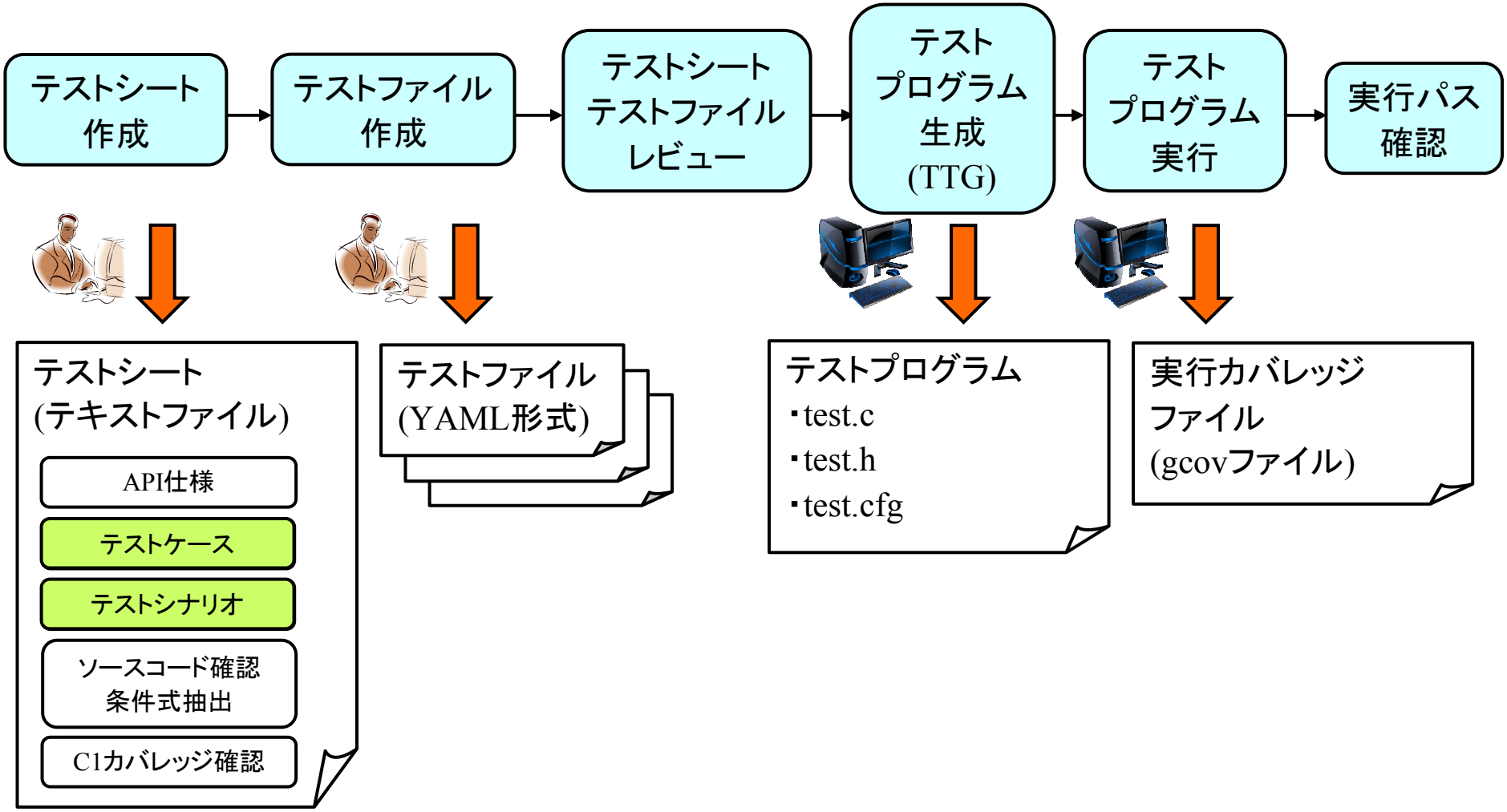
仕様の振舞いを確認するテストを実施する

- 自タスク以外のタスクを指定して呼び出す.
- 起床待ち状態のタスクを指定する.
- 対象タスクの優先度が，実行状態のタスクより高い場合，対象タスクが実行状態になること.



APIテスト開発の作業フロー

API毎に以下のフローを実施する



APIテスト - テストケース -

wup_tskのテストケース

1.1. エラー条件のテストケース

(a) 非タスクコンテキストから呼び出して，E_CTXエラーが返ること．
(b) CPUロック状態で呼び出して，E_CTXエラーが返ること．
(c) tskidが不正の時にE_IDが返ること．
　(c-1) tskidが許容される最小値-1(-1)の時にE_IDが返ること．
　(c-2) tskidが許容される最大値+1(TNUM_TSKID+1)の時にE_IDが返ること．
(d) 対象タスクが休止状態の時にE_OBJが返ること．
(e) 対象タスクの起床要求キューイング数がTMAX_WUPCNT(=1)に一致している時に呼ぶと，E_QOVRが返ること．
　(e-1) 対象タスクが自タスクの場合．
　(e-2) 対象タスクが他タスクの場合．
　　：

1.2. 正常条件のテストケース

(g) 起床要求キューイング数が0である自タスク以外のタスクを指定して呼び出す．
　(g-1) 起床待ち状態のタスクを指定する．
　　(g-1-1) 対象タスクの優先度が，実行状態のタスクより高い場合，
　　　(g-1-1-1) 実行状態になること．
　　　(g-1-1-2) ディスパッチ禁止状態の場合，実行可能状態になること．
　　　(g-1-1-3) 割込み優先度マスクが全解除でない場合，実行可能状態になること．
　　(g-1-2) 対象タスクの優先度が，実行状態のタスクより低い場合，実行可能状態になり，同じ優先度のタスクの最後につながれること．
　　(g-1-3) 対象タスクの優先度が，実行状態のタスクと同じ場合，実行可能状態となり，同じ優先度のタスクの最後につながれること．
　　：

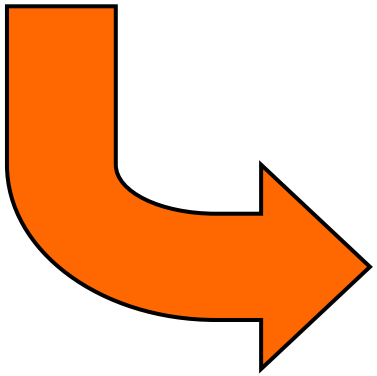
API毎の仕様から作成可能なテストケース

統合仕様書全体からAPIの振舞いを考慮する必要があるテストケース

APIテスト - テストシナリオ -

テストケース毎にテストを実施するための
シナリオを前状態/処理/後状態に分けて記述する

(g-1-1-1)の例
・自タスク以外のタスクを指定して呼び出す.
・起床待ち状態のタスクを指定する.
・対象タスクの優先度が，実行状態のタスクより
高い場合，対象タスクが実行状態になること.



前状態
タイプ　：タスク
ID　：TASK1
優先度　：中
状態　：実行

タイプ　：タスク
ID　：TASK2
優先度　：高
状態　：待ち
待ち要因：SLEEP

処理
TASK1がwup_tsk(TASK2)を発行する

後状態
タイプ　：タスク
ID　：TASK1
優先度　：中
状態　：実行可能

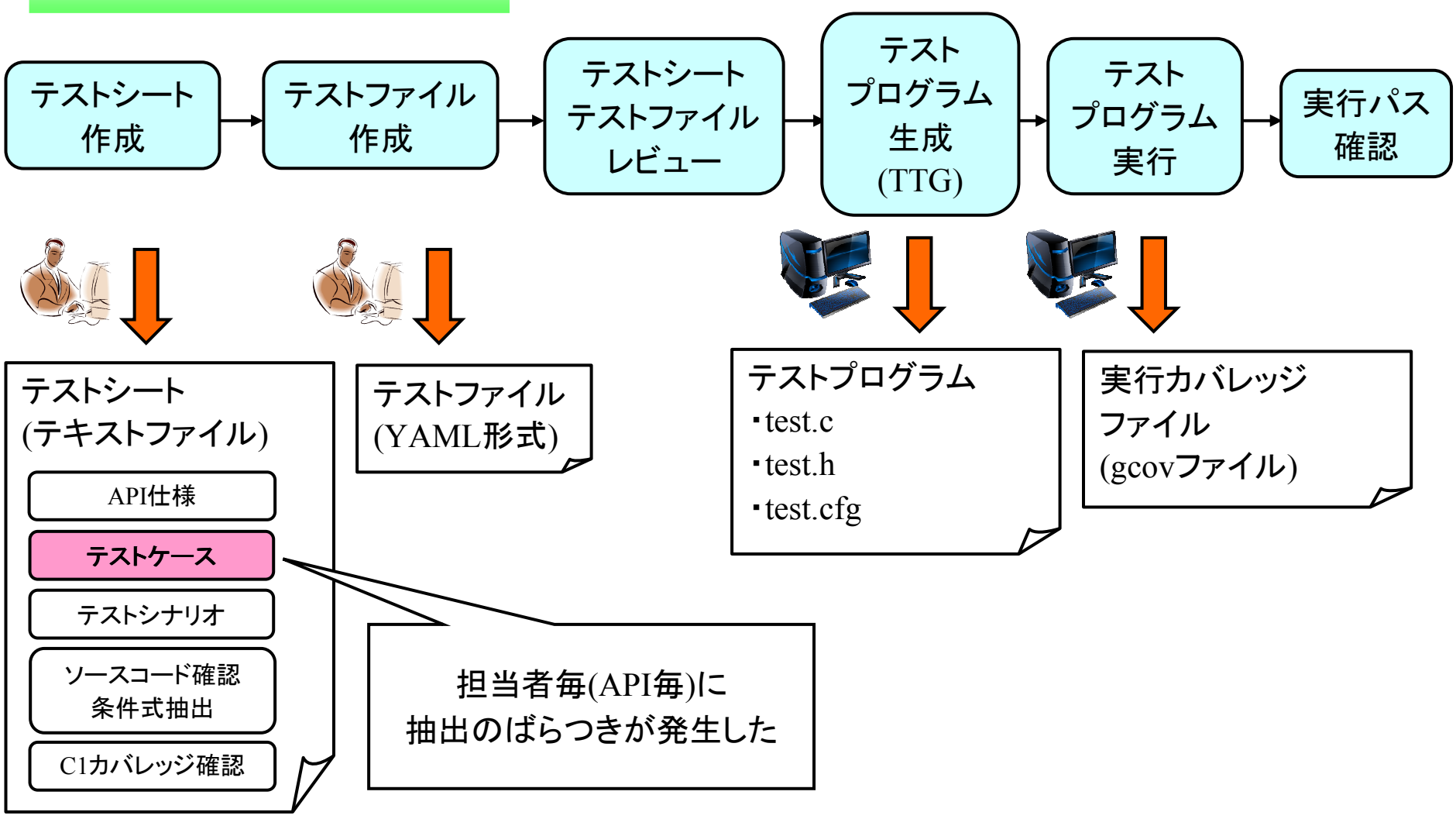
タイプ　：タスク
ID　：TASK2
優先度　：高
状態　：実行

アジェンダ

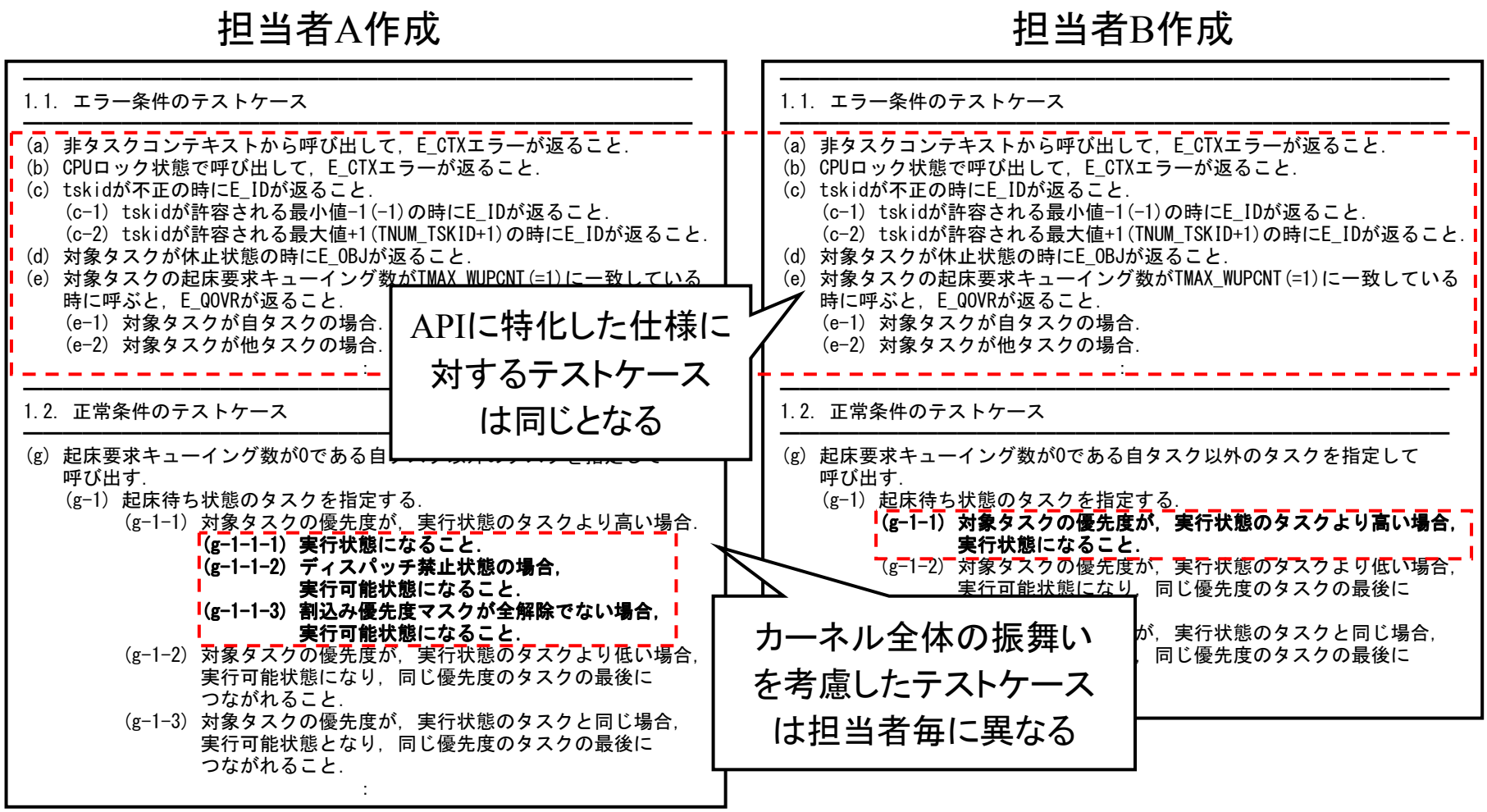
1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

実施の際に発生した問題点

APIテスト作業フロー



発生したばらつき① - テストケース抽出レベル -



担当者のカーネル理解度によってテストケースがばらついてしまう

ばらつき①が発生する要因(1) 統合仕様書の構成

統合仕様書からAPI毎の
仕様部分を引用して
テストシートへ記載している

統合仕様書全体を考慮すると
対象APIの振舞いとして
テストすべき仕様が存在する

wup_tsk　　タスクの起床

【C言語API】
　ER ercd = wup_tsk(ID tskid)

【パラメータ】
　ID tskid 対象タスクのID番号

【リターンパラメータ】
　ER ercd 正常終了(E_OK)またはエラーコード

【エラーコード】

E_CTX	コンテキストエラー(非タスクコンテキストからの呼出し:wup_tskの場合，タスクコンテキストからの呼出し:iwup_tskの場合，CPUロック状態からの呼出し)
E_ID	不正ID番号(tskidが不正)
E_NOEXS〔D〕	オブジェクト未登録(対象タスクが未登録)
E_OACV〔P〕	オブジェクトアクセス違反(対象タスクに対する通常操作1が許可されていない:wup_tskの場合)
E_OBJ	オブジェクト状態エラー(対象タスクが休止状態)
E_QOVR	キューイングオーバフロー(起床要求キューイング数がTMAX_WUPCNTに一致)

【機能】
tskidで指定したタスク(対象タスク)を起床する．具体的な振舞いは以下の通り．
対象タスクが起床待ち状態である場合には，対象タスクが待ち解除される．
待ち解除されたタスクには，待ち状態となったサービスコールからE_OKが返る．
対象タスクが起床待ち状態でなく，休止状態でもない場合には，対象タスクの起床要求キューイング数に1が加えられる．起床要求キューイング数に1を加えるとTMAX_WUPCNTを超える場合には，E_QOVRエラーとなる．
対象タスクが休止状態である場合には，E_OBJエラーとなる．
tskidにTSK_SELF(=0)を指定すると，自タスクが対象タスクとなる．

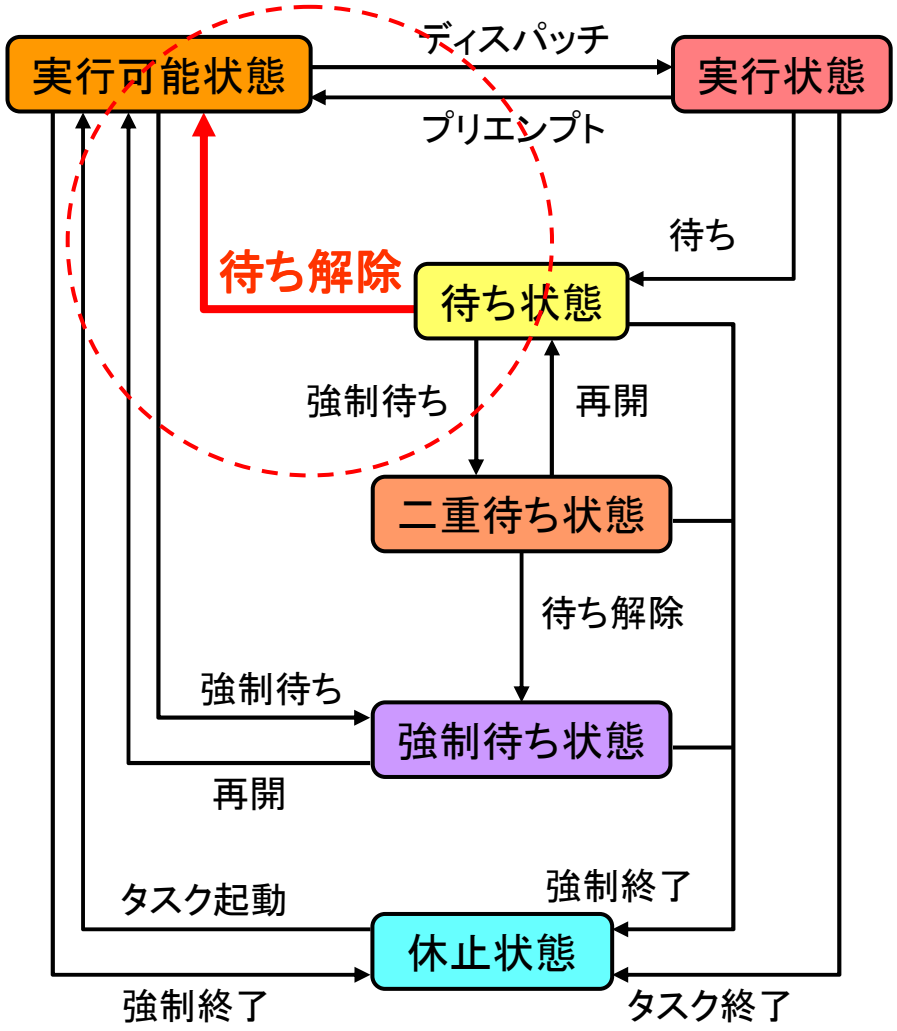
(例)第2章 主要な概念と共通定義
「ディスパッチ禁止状態とディスパッチ許可状態」
プロセッサは，ディスパッチを保留するためのディスパッチ禁止フラグを持つ．ディスパッチ禁止フラグがセットされた状態をディスパッチ禁止状態，クリアされた状態をディスパッチ許可状態と呼ぶ．すなわち，**ディスパッチ禁止状態では，ディスパッチは保留される**．
ディスパッチ禁止状態では，別に規定がない限りは，
自タスクを広義の待ち状態に遷移させる可能性のある
サービスコールを呼び出すことはできない．呼び出した
場合には，E_CTXエラーとなる．
マルチプロセッサ対応カーネルでは，プロセッサ毎に，
ディスパッチ禁止フラグを持つ．すなわち，
プロセッサ毎に，ディスパッチ
許可状態のいずれかの状態を取る．

統合仕様書:13,564行

**ディスパッチ禁止状態で
wup_tskを発行する
テストケースが必要！**

ばらつき①が発生する要因(2) APIの対称性

タスクの状態遷移



待ち状態から実行可能状態へ遷移させるAPIは複数存在する

(例)

wup_tsk：
　起床待ちを起床する

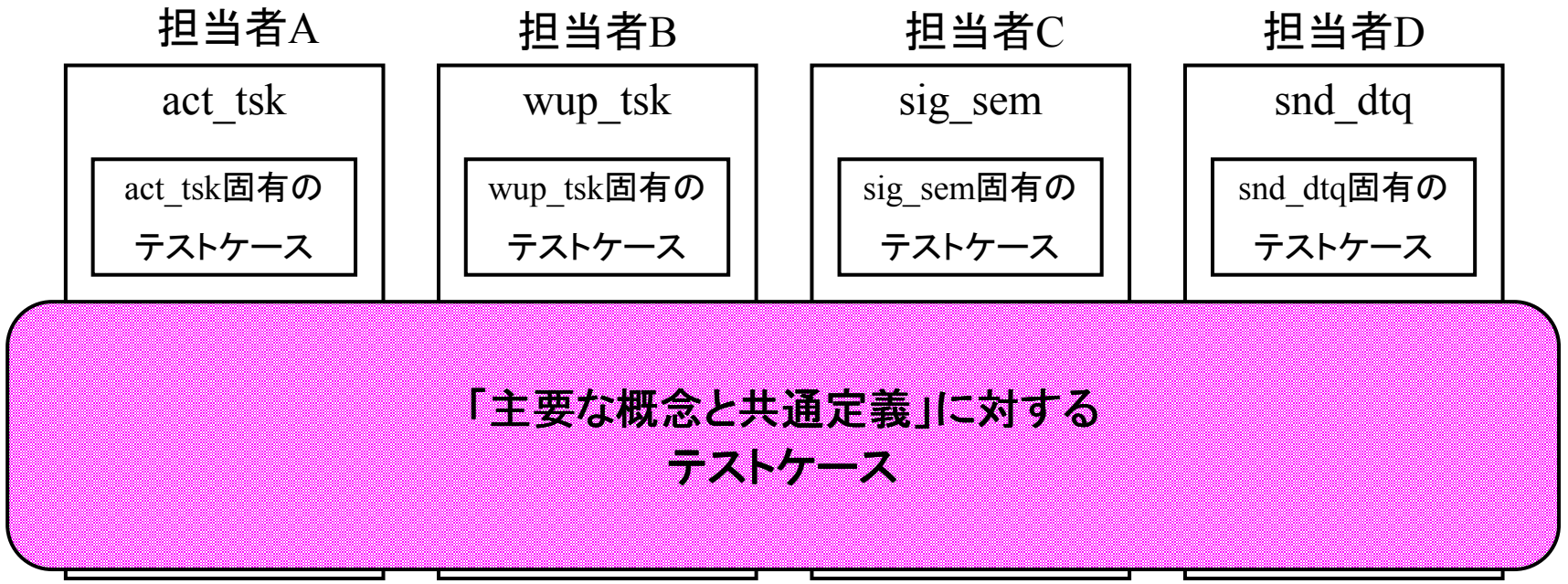
sig_sem：
　セマフォ待ちが待ち解除される

rel_wai：
　待ち状態が強制解除される

待ち状態が解除された場合にテストするカーネルの振舞いを全APIのテストケースで統一する必要がある

ばらつき①発生のイメージ

API毎に担当者を割り振っているため，
APIに関係ないカーネル全体の仕様部分の
テストケースがばらついてしまう



発生したばらつき② - 実施範囲 –

担当者A作成

1.2. 正常条件のテストケース

(g) 起床要求キューイング数が0である自タスク以外のタスクを指定して呼び出す.

(g-1) 起床待ち状態のタスクを指定する.

(g-2) 二重待ち(起床待ち)状態のタスクを指定する.

(g-3) 二重待ち(時間経過待ち)状態のタスクを指定する.

(g-4) 実行可能状態のタスクを指定する.

(g-5) 強制待ち状態のタスクを指定する.

(g-6) 時間経過待ち状態のタスクを指定する.

(g-7) セマフォの資源獲得待ち(タイムアウト無)状態のタスクを指定する.

(g-8) セマフォの資源獲得待ち(タイムアウト有)状態のタスクを指定する.

:

担当者B作成

1.2. 正常条件のテストケース

(g) 起床要求キューイング数が0である自タスク以外のタスクを指定して呼び出す.

(g-1) 起床待ち状態のタスクを指定する.

(g-2) 二重待ち状態のタスクを指定する.

(g-3) 実行可能状態のタスクを指定する.

(g-4) 強制待ち状態のタスクを指定する.

(g-5) 時間経過待ち状態のタスクを指定する.

(g-6) セマフォの資源獲得待ち状態のタスクを指定する.

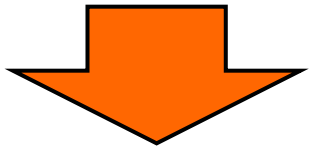
:

実施対象とするタスクの状態が複数存在する場合担当者毎にテストケースが異なる

テストする範囲に関して明確な規定(ポリシー)がなかったために，テストケースがばらついてしまう(同値分割の粒度が異なってしまう)

ばらつき①②への対策

対称性のあるRTOSのAPIのテストケースに対称性を持たせるには，抽出ポリシー(マニュアル)が必要である



ばらつき①
統合仕様書全体を確認し，
テストケース作成時に考慮すべき
カーネルの振舞いについてまとめた

ばらつき②
どのレベルまでテストを実施する
必要があるかコンソーシアムへ参加
している各社の経験を持ち寄りまとめた

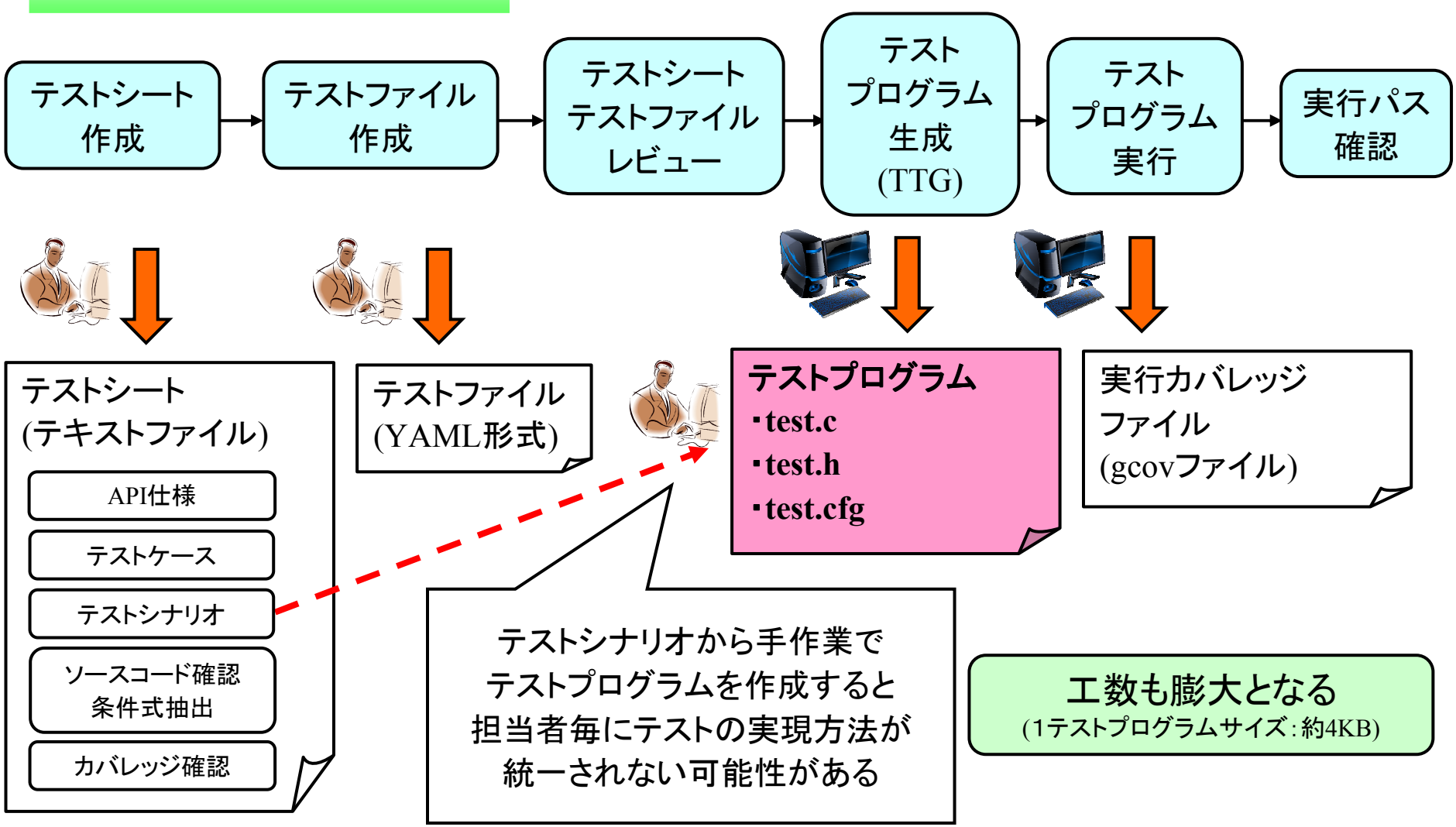
(例)
- ディスパッチ禁止状態ではディスパッチが発生しないパターンを考慮する
- APIを発行する対象タスクの取り得る状態は代表する10種類とする
- 取り得る優先度は抽象化した高，中，低の3種類から組み合わせる
- 引数に与えるデータ型毎の境界値分析の適用方法
- 非タスクコンテキストのテストはアラームハンドラを使用する

アジェンダ

1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

テストプログラム作成

APIテスト作業フロー



テストプログラム作成ツール

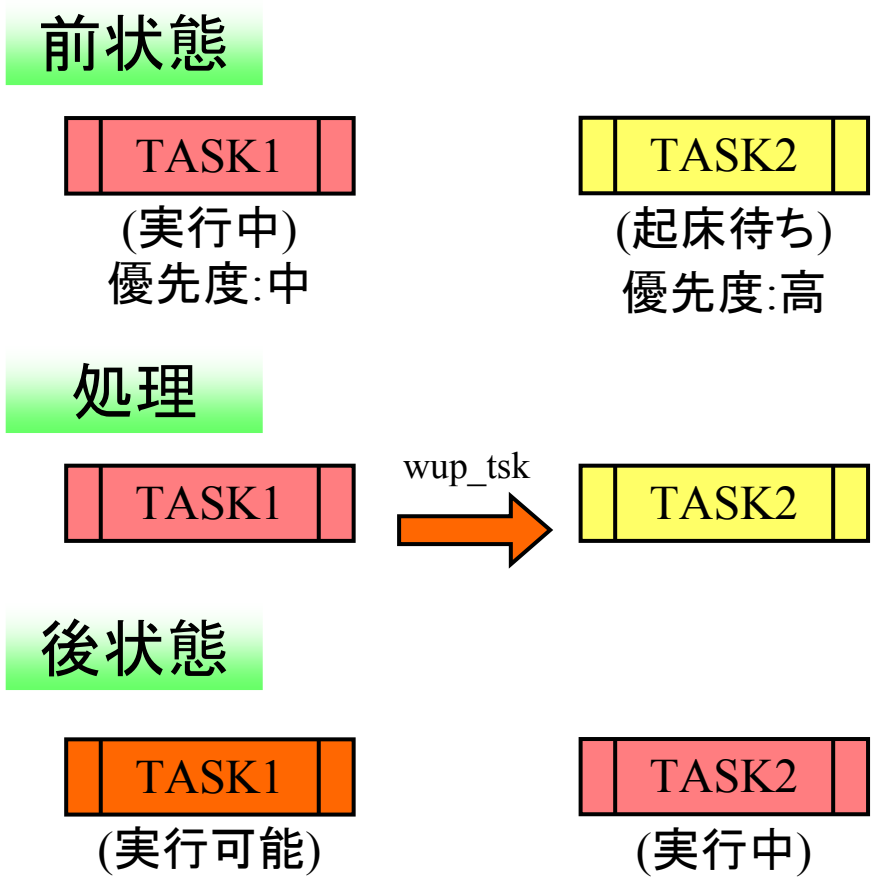
ツールの必要性

以下のテストケースを実施するための
テストプログラムを作成する場合を考える

前状態
　優先度中のTASK1が実行状態
　優先度高のTASK2が起床待ち状態

処理
　TASK1がwup_tsk(TASK2)を発行する

後状態
　TASK1が実行可能状態となる
　TASK2が実行状態となる



テストプログラム作成時の問題点

＜実現する前状態＞
[TASK1]
　優先度:中
　実行状態
[TASK2]
　優先度:高
　起床待ち状態

act_tsk:タスク起動
ref_tsk:タスク状態参照
slp_tsk:タスク起床待ち

担当者A作成

優先度：**TEST_TASK** ＜ TASK1 ＜ TASK2

```
TEST_TASK()
{
    act_tsk(TASK2);
    ref_tsk(TASK2);
    act_tsk(TASK1);
}

TASK2()
{
    slp_tsk();
}

TASK1()
{
    ref_tsk(TASK1); 前状態実現/確認完了
    wup_tsk(TASK2);
}
```

担当者B作成

優先度：TASK1 ＜ TASK2 ＜ **TEST_TASK**

```
TEST_TASK()
{
    act_tsk(TASK2);
    act_tsk(TASK1);
    slp_tsk();
}

TASK2()
{
    slp_tsk();
}

TASK1()
{
    ref_tsk(TASK1);
    ref_tsk(TASK2); 前状態実現/確認完了
    wup_tsk(TASK2);
}
```

前状態の実現/確認方法が異なる

どちらも正しいテストプログラムだが，
正しいことを確認するレビュー時の可読性や，
修正時の保守性が著しく低下する

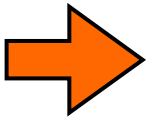
テストファイル

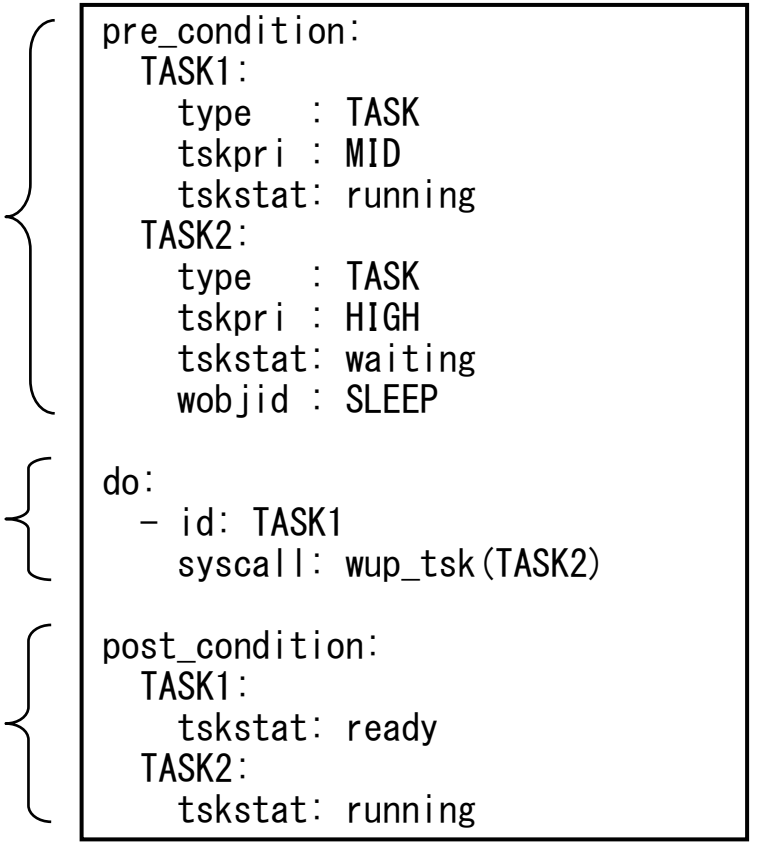
テストプログラムを作成するツールへの入力データ
(テストファイル)を階層型データ形式言語であるYAML形式として
全カーネルオブジェクトの属性/状態の記述方法を定めた

前状態
　優先度中のTASK1が実行状態
　優先度高のTASK2が起床待ち状態

処理
　TASK1がwup_tsk(TASK2)を発行する

後状態
　TASK1が実行可能状態となる
　TASK2が実行状態となる





```
pre_condition:
  TASK1:
    type    : TASK
    tskpri  : MID
    tskstat: running
  TASK2:
    type    : TASK
    tskpri  : HIGH
    tskstat: waiting
    wobjid  : SLEEP

do:
  - id: TASK1
    syscall: wup_tsk(TASK2)

post_condition:
  TASK1:
    tskstat: ready
  TASK2:
    tskstat: running
```

TTG(TOPPERS Test Generator)

テストファイル(YAML)から
テストプログラムを生成するツールを開発した

<u>テストプログラム自動生成のメリット</u>

- 開発工数削減
- テストケースの保守性，可読性
- カーネルのバリエーション展開
- 複数のテストケースをまとめられるため，
  テスト実施工数が削減可能
- ターゲットのメモリ制約に応じてテストプログラムを分割可能
- テストポリシー拡張の可能性

但し，一部対応できない複雑なテストケースもあった
→工数との兼ね合いでスクラッチでテストプログラムを作成した

アジェンダ

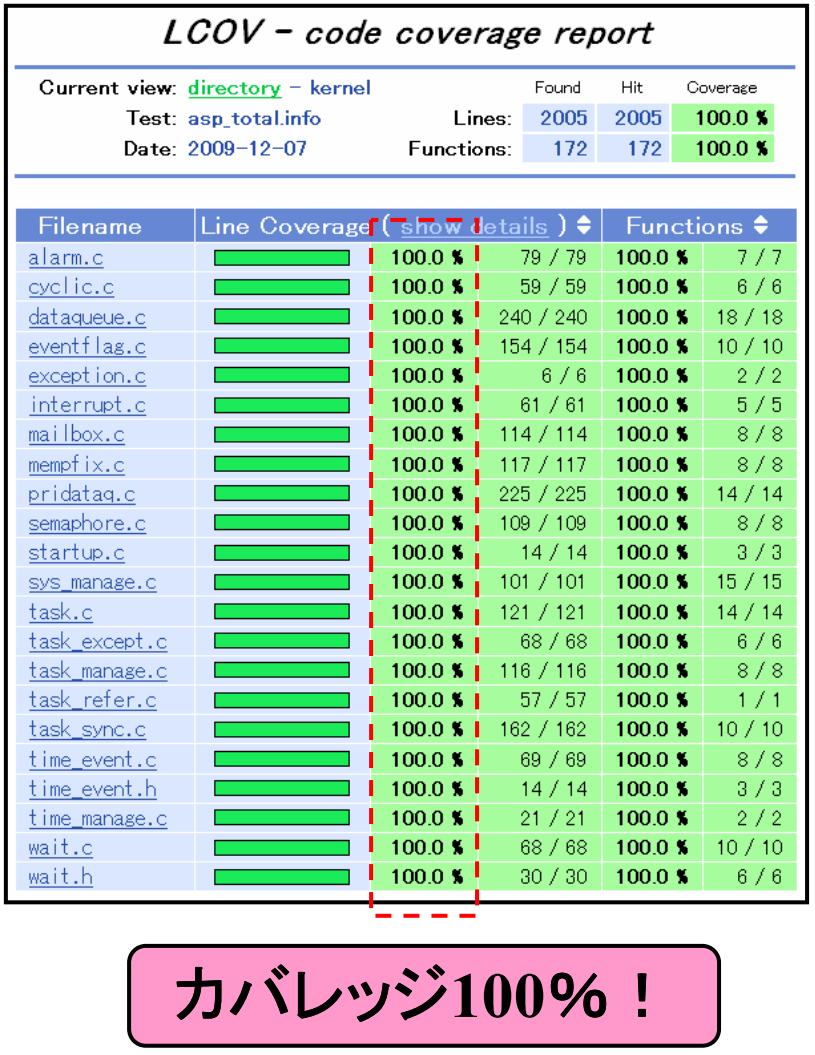
1. オープンソースの組込み向けRTOSとそのテスト
2. マルチプロセッサ対応RTOSに関するコンソーシアム型研究組織
3. ASPカーネル仕様
4. APIテストの概要と開発フロー
5. 発生したばらつきと対策
6. テストプログラム開発工数削減への取り組み
7. テスト実施結果と今後の展望

テスト実施結果

機能毎のテストプログラム数一覧

機能名	テスト ファイル数	スクラッチ数	テスト プログラム数
タスク管理機能(task_manage)	144	1	145
タスク付属同期機能(task_sync)	200	4	204
タスク例外処理機能(task_except)	63	0	63
データキュー機能(dataqueue)	263	1	264
イベントフラグ機能(eventflag)	144	1	145
メールボックス機能(mailbox)	85	1	86
固定長メモリプール機能(mempfix)	84	7	91
優先度データキュー機能(pridataq)	302	1	303
セマフォ機能(semaphore)	101	1	102
周期ハンドラ機能(cyclic)	17	5	22
アラームハンドラ機能(alarm)	31	1	32
システム状態管理機能(sys_manage)	62	3	65
割込み管理機能(interrupt)	19	3	22
CPU例外管理機能(exception)	0	8	8
タスクの状態参照機能(task_refer)	4	1	5
システム時刻管理機能(time_manage)	7	0	7
静的API	0	108	108
合計	**1,526**	**146**	**1,672**

全テストプログラム実行後のカバレッジ



LCOV - code coverage report

Current view: directory - kernel
Test: asp_total.info
Date: 2009-12-07

	Found	Hit	Coverage
Lines:	2005	2005	100.0 %
Functions:	172	172	100.0 %

Filename	Line Coverage (show details)		Functions	
alarm.c	100.0 %	79 / 79	100.0 %	7 / 7
cyclic.c	100.0 %	59 / 59	100.0 %	6 / 6
dataqueue.c	100.0 %	240 / 240	100.0 %	18 / 18
eventflag.c	100.0 %	154 / 154	100.0 %	10 / 10
exception.c	100.0 %	6 / 6	100.0 %	2 / 2
interrupt.c	100.0 %	61 / 61	100.0 %	5 / 5
mailbox.c	100.0 %	114 / 114	100.0 %	8 / 8
mempfix.c	100.0 %	117 / 117	100.0 %	8 / 8
pridataq.c	100.0 %	225 / 225	100.0 %	14 / 14
semaphore.c	100.0 %	109 / 109	100.0 %	8 / 8
startup.c	100.0 %	14 / 14	100.0 %	3 / 3
sys_manage.c	100.0 %	101 / 101	100.0 %	15 / 15
task.c	100.0 %	121 / 121	100.0 %	14 / 14
task_except.c	100.0 %	68 / 68	100.0 %	6 / 6
task_manage.c	100.0 %	116 / 116	100.0 %	8 / 8
task_refer.c	100.0 %	57 / 57	100.0 %	1 / 1
task_sync.c	100.0 %	162 / 162	100.0 %	10 / 10
time_event.c	100.0 %	69 / 69	100.0 %	8 / 8
time_event.h	100.0 %	14 / 14	100.0 %	3 / 3
time_manage.c	100.0 %	21 / 21	100.0 %	2 / 2
wait.c	100.0 %	68 / 68	100.0 %	10 / 10
wait.h	100.0 %	30 / 30	100.0 %	6 / 6

カバレッジ100%！

検出されたバグ

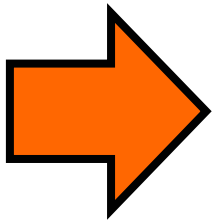
統合仕様書とASPカーネルの
バグを計12件検出した

仕様上のバグ(5件)の例

- 曖昧な用語定義/エラー条件の整備
- ソース上考慮されている振舞いが，仕様に記載されていない

ソース上のバグ(7件)の例

- 仕様変更に追従していなかった処理が存在した
- 不要な条件分岐が存在する
- 使用されていない内部関数が存在する



**仕様，ソースの両面から包括的に
APIの振舞いをテストできた**

今後の展望

統合仕様書とテストスイートのマッピング

・テストスイートと仕様書の関係を管理できる

　・仕様書が改訂された場合に対応するテストケースを抽出できる

→実現案として仕様書の全ての記述に対して番号を付与する

※仕様書をナンバリングする記法やツールについては未検討　何かありませんか？？

テストポリシーのカスタマイズ

・今回開発したテストスイートのテストポリシーは固定されている

　→実際はRTOSを使用する企業によってポリシーは異なる

・テストポリシー自体を形式的に定義し，テストを自動生成する手法が望まれる

　→求める品質に合わせたテストを効率的に実施可能となる

FMPカーネルへの拡張

・マルチプロセッサに対応したテストポリシーを策定する

まとめ

- 名古屋大学を中心にマルチプロセッサ対応RTOSに関するコンソーシアム型研究組織を立ち上げた
- 2009年度はAPIに着目したテストスイートを開発し，実施した
  - テストポリシー策定の必要性
  - テストプログラム作成ツール開発
- 結果，約1,600件の包括的なテストケースを開発できた

NCESより

今後，マルチプロセッサ向けRTOSの共同研究を継続するにあたり，

参加して頂ける企業/団体を募集しています

http://www.nces.is.nagoya-u.ac.jp/