ET2006スペシャルセッション

TOPPERS 組込みコンポーネント仕様 (TECS) の概要

TOPPERSプロジェクト コンポーネント仕様WG主査
オークマ株式会社
大山 博司

©TOPPERSプロジェクト

はじめに

- 3年の歳月をかけ、ようやくコンポーネント仕様が固まりつつある
  - 一方で、まだ固まっていない部分もある
    - 特に動的コンポーネントについて
  - 固まったところ（静的）を中心に説明する

TECS: TOPPERS Embedded Component System という名前もようやくこの春に決まった (この部分追記)

【主な内容】

Part1. 組込みコンポーネントの背景と目標

Part2. TECS仕様について

©TOPPERSプロジェクト

Part1.組込みコンポーネントの背景と目標

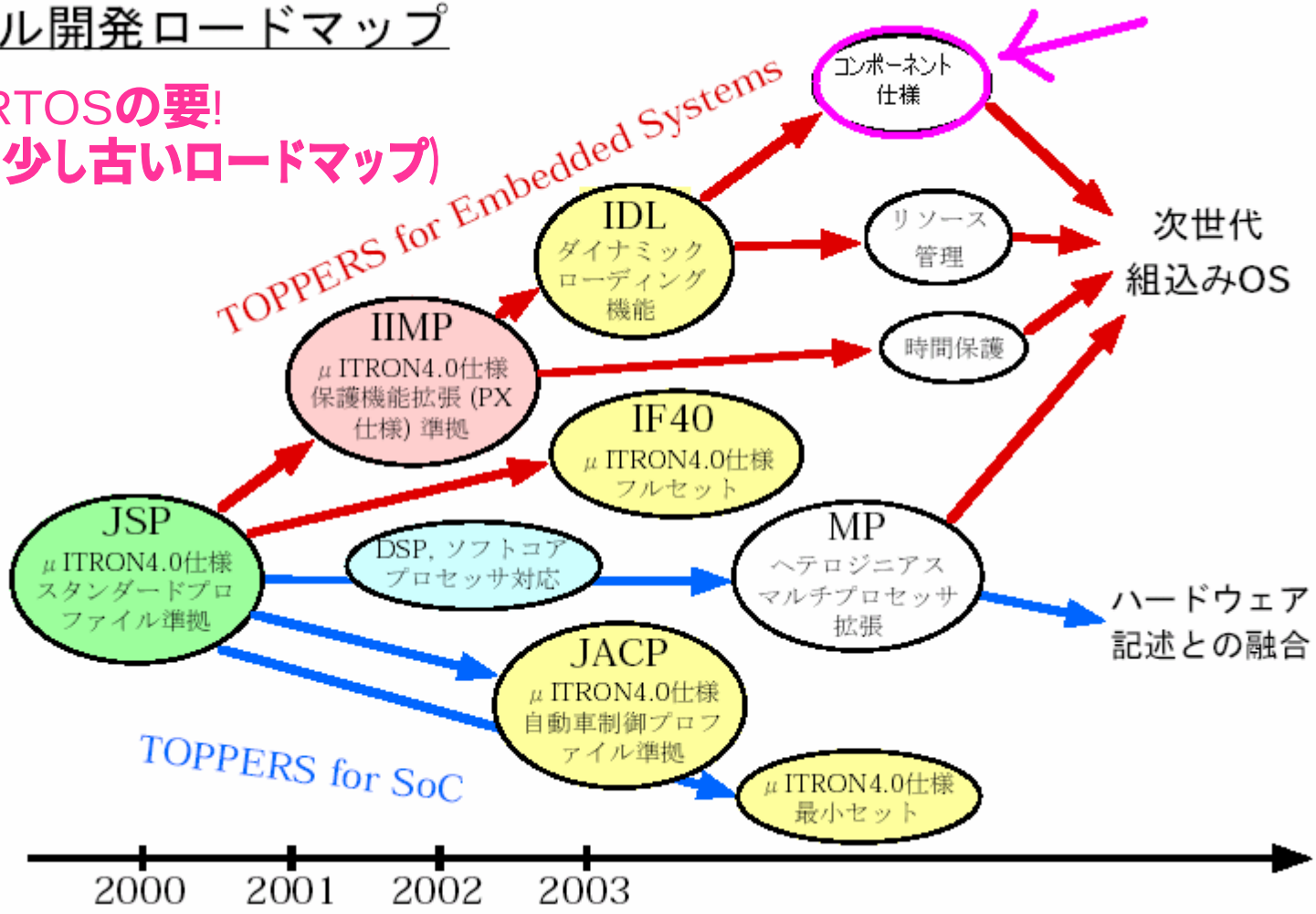
- TOPPERSプロジェクト内での位置づけ
- 組込みコンポーネントへの期待
- 非組込みコンポーネント
- 組込みコンポーネントとは
- なぜ組込みシステムではコンポーネントが、まだ実現していないのか
- なんの役に立つのか(目的)
- 組込みコンポーネントTECSの特徴
- 正確なインタフェース記述
- TECS開発の目標

©TOPPERSプロジェクト

TOPPERSプロジェクト内での位置付け

カーネル開発ロードマップ

- **次世代RTOSの要!**
  (注意！少し古いロードマップ)



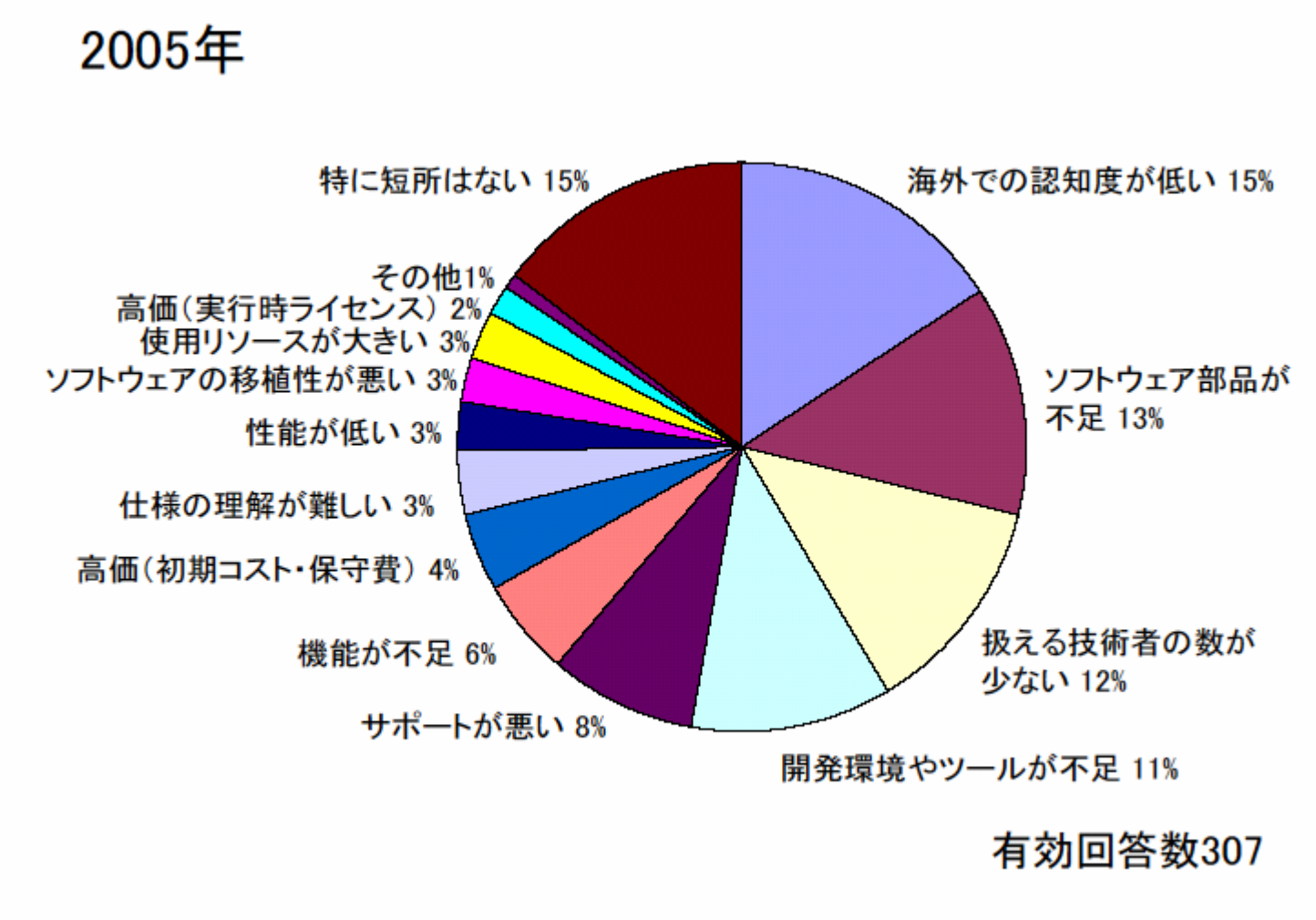
©TOPPERSプロジェクト
TOPPERS

組込みコンポーネントへの期待

- 組込みソフト部品に対する期待は根強い
  - 組込みコンポーネントを組合わせて組込みソフトウェアを組上げれれば生産性は飛躍的に改善されることが期待される
- 長年期待されているが、改善が進んでいない分野
  - 方法論（あるいは戦略）、技術論が欠けていた？
- 組込みコンポーネント仕様の役割
  - 種々の部品について検討するのは時間がかかる
    - TCP/IP API仕様など個別の標準化では間に合わない
  - 部品の標準化を容易にするための仕組みを設ける

©TOPPERSプロジェクト

ITRON 仕様OSの短所



TRON協会アンケートより

©TOPPERSプロジェクト

非組込みコンポーネント・組込みコンポーネント

- **既に非組込み系では、かなり普及している**
  - **10年ほど前から各種のソフトウェアコンポーネントフレームワークが提案されていて、広く普及している**
  - MS COM
  - JavaBeans
  - CORBA　CCM
  - GNU BONOBO
  - Netscape XPCOM
  - UML**のコンポーネント図**
- **組込みコンポーネント**
  - KOALA(Philips**社**)　…　**プロダクトライン的発想**
  - AUTOSAR(**ヨーロッパの自動車関連メーカ**)　…　**ネット接続**

©TOPPERSプロジェクト

非組込みにおけるコンポーネントの定義

- あるサービスを提供する，独立して交換可能なソフトウェアの構成単位のこと
- 特定のコンポーネント実現システムに従って再利用可能な関連を持ったオブジェクト指向クラス・インタフェース群が、再配布可能かつソースコードが非開示な形でパッケージ化されたもの

（以上、早稲田大学理工学部 深澤研究室ＨＰより）

- any software (sub)system that can be factored out and has a potentially standardizable or reusable exposed interface

(Object Services and Consulting, Inc HP より)

- システム内の物理的かつ交換可能な部分で、一連のインターフェースに適合し、それを実装するもの

(UMLの用語集)

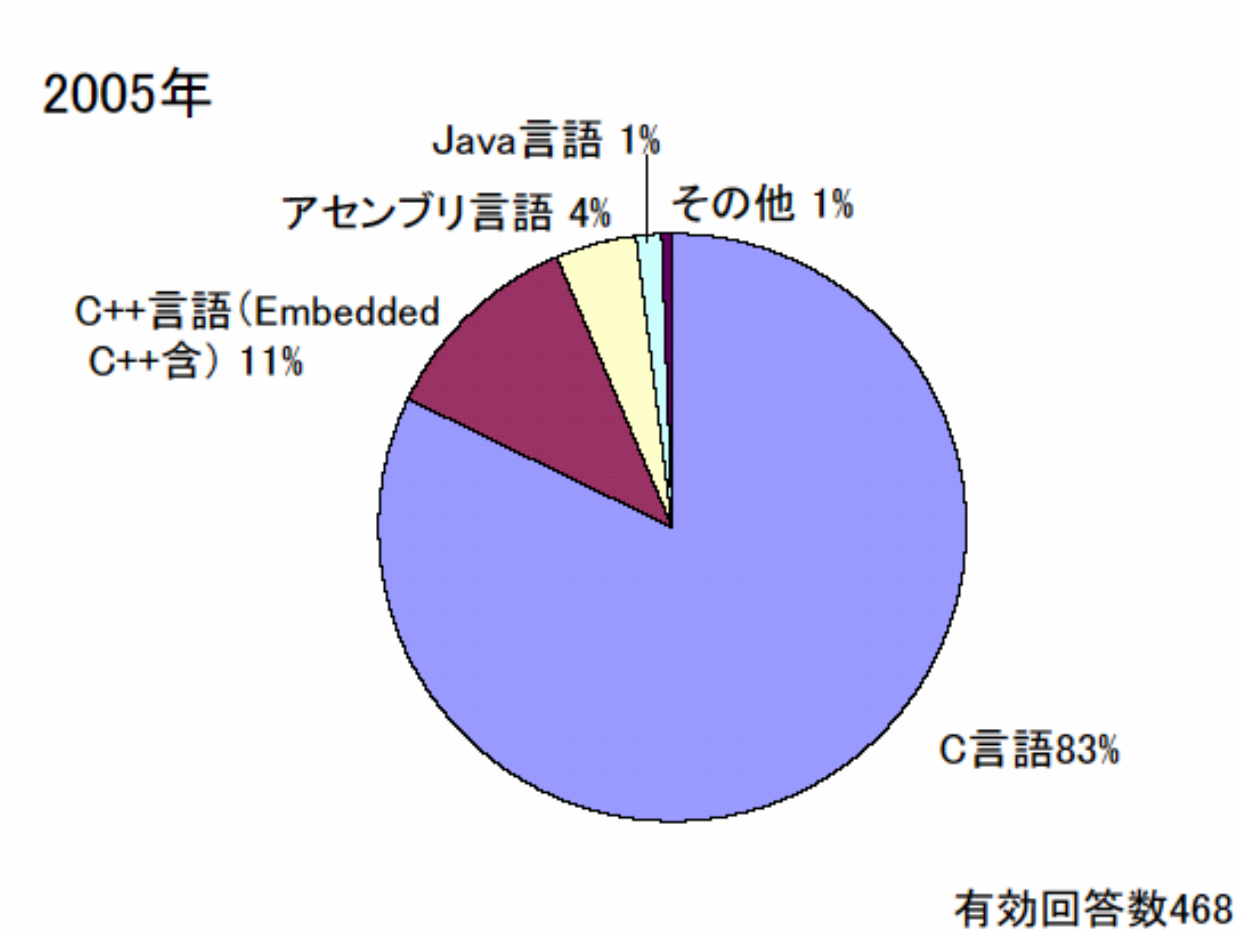
©TOPPERSプロジェクト

組込みコンポーネントとは

- 組込みシステムの特性に適した機能、制約を考慮した、組込みソフトウェア部品化の仕組み
  - オブジェクト指向技術を直接的には取り入れない
    - 組込みシステム開発ではC言語の使用が中心
  - 静的なコンポーネント定義・生成・結合を基本とする
    - 無用なオーバーヘッドを避ける
    - 動作させなくても検証できる要素が多いほうが望ましい
  - 過剰な機能は必要ない
    - 組込みドメインに限定すれば、機能限定して、低コスト（開発費用だけでなくメモリ、CPUを含めて）が望ましい
  - 大規模化する組込みソフトを効率よく開発することに焦点をあてる
    - 部品不足、標準化の遅れがより重要な課題
    - リアルタイム制約、メモリ制約の配慮など、今後の課題

©TOPPERSプロジェクト

組込みシステム開発のプログラミング言語



TRON協会アンケートより

©TOPPERSプロジェクト

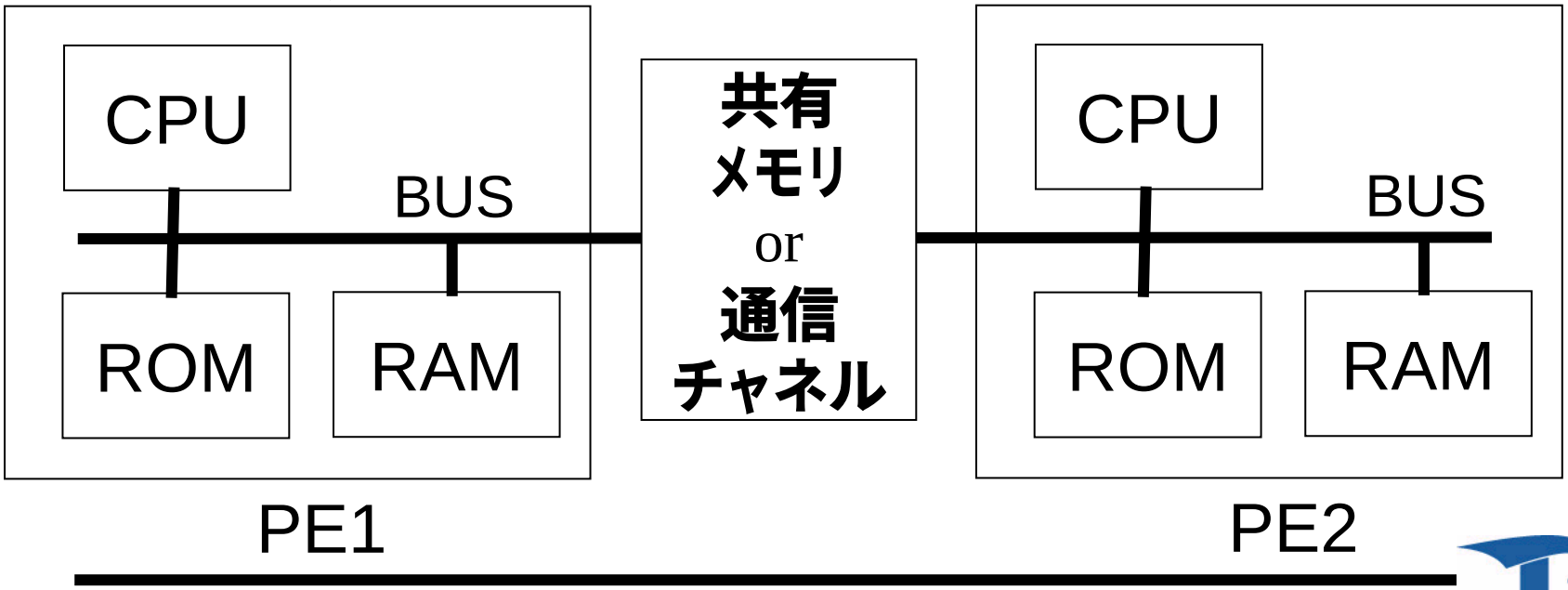
なぜ組込みシステムではコンポーネントがこれまで実現していないのか？

- オブジェクト指向プログラミングの普及が進んでいない
  - (C++よりはCがよく利用されている)
  - ⇒ 組込みコンポーネントはオブジェクト指向を前提にしない
- ソフトウェア部品を組込む手段として、静的ライブラリ、リロケータブルオブジェクトの供給が普通である
  - ポーティングに手間がかかるほうがベンダーには色々と都合がよい？
    - ベンダー間の競争が促進されるような仕組みが必要
- 分散フレームワークが普及していない
  - 分散フレームワークは、独立性の高いモジュールを作成する規範を与える (インタフェース定義言語)
  - 一般的な分散フレームワークは過剰であり、大げさすぎる
    - オーバーヘッドが十分に小さなものであること
  - 組込みドメインの通信は多様であり、汎用的なものが考えにくかった

©TOPPERSプロジェクト

分散フレームワークとは (追加ページ)

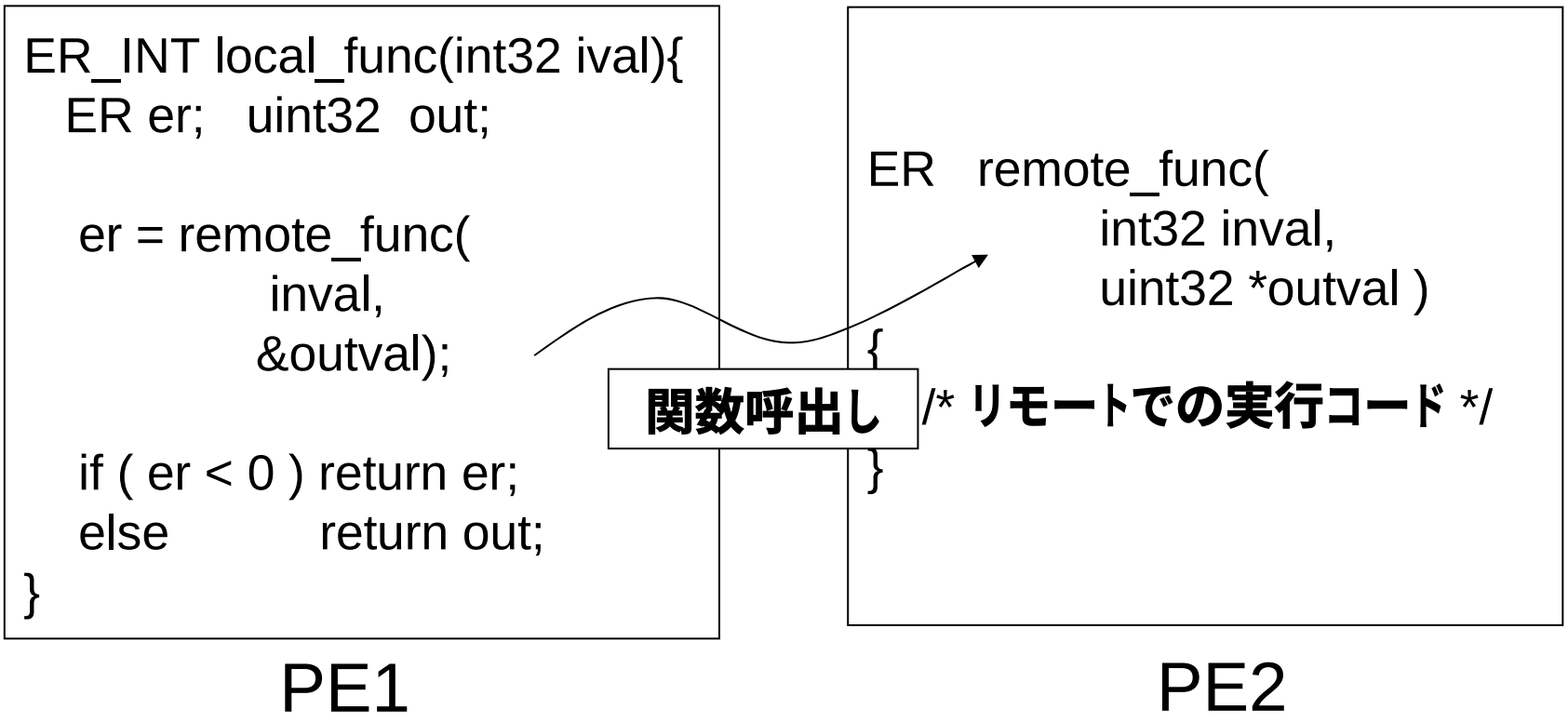
- マルチプロセッサ間での関数呼出し
  - 必ずしもこれに限定されるものではないが、今日多く用いられるものは、この方式である
  - 例えば以下のような共有メモリで結合されたマルチプロセッサシステムにおいて



©TOPPERSプロジェクト

分散フレームワークとは2 (追加ページ)

PE1で呼出された関数がPE2で実行される



©TOPPERSプロジェクト

なんの役に立つのか

- モノリシック（一枚岩）構造の組み込みソフトウェアからコンポーネント構造の組込みソフトウェアへと、設計パラダイムをシフトする
  - コンポーネントを意識したアプリ開発
    - 適度な粒度のオブジェクトに着目した設計（facade）
    - 独立性の高いモジュール構造の実現
  - ソフトウェア部品の流通
    - デバイスドライバなどのインターフェースの基準
    - インターフェースの統一による互換性のある部品の流通
    - 異なるメーカーのスタックを組合わせて利用
- システム内、デバイス内のマルチプロセッサ環境にも対応する分散フレームワークを実現する
  - 組込み分散フレームワークの実現
  - マルチCPUシステムのPE間の通信設計の自動化、容易化

©TOPPERSプロジェクト

組込みコンポーネントTECSの特徴

- システム内のすべてのソフトウェアをコンポーネントとして扱える
  - アロケータ、RPCチャネルなどもコンポーネントである
- 静的なコンポーネント結合を基本とするコンポーネントシステム
  - オーバーヘッドが小さく、小さな組込みシステムにも容易に適用できる
- 柔軟な分散機構や動的生成/結合機構の実現
  - 静的なコンポーネントであることが、動的なコンポーネント以上に柔軟性を与えている
- その他の特徴
  - コンポーネントを合成した複合コンポーネントの機能を持つ
  - コンポーネントの提供する機能に加えて、コンポーネントが使う機能を記述することができ、コンポーネント間の依存関係が明瞭である（アーキテクチャの表現手段）

©TOPPERSプロジェクト

正確なインタフェース記述

ーコンポーネントシステムの肝ー

- **C言語のインタフェース記述（プロトタイプ宣言）は曖昧**
  - ER do_function(char *buf, int size, int *result);
- TECS**によるインタフェース定義**　（TECS**用語ではシグニチャ**）
  - ER do_function([in,size_is(size)] const char *buf,
    [in] int size,
    [inout] int *result);

（*result **が**inout**なのは、前回の結果が今回の実行に影響するため**）

- **引数が入力か出力かが明確（特にポインタ型）**
- **ポインタが配列を指しているかスカラ型を指しているかが明確**
- **配列であるならば、配列の大きさが明確**

©TOPPERSプロジェクト

正確なインタフェース記述2 (追加ページ)

- アロケータにより確保されたメモリの引渡し
  - 受け取った側は、メモリ解放しなくてはならない
  - send, receive により in, out とは区別する *(TECSの特徴)*
  - と同時に使用するアロケータを指定する *(同上)*
    - 異なるタイプのアロケータを利用できる

```
ER  send_mem(
        [send(sigAlloc),count_is(len)]char *buf,
        [in]int len );
ER  recv_mem(
        [recv(sigAlloc),count_is(len)]char **buf,
        [out]int *len );
```


©TOPPERSプロジェクト

正確なインタフェース記述の意義

- 簡潔に曖昧なく記述できる
- 部品化するにあたってインタフェース記述が統一される
- 結果としてインタフェースの誤解が少なくなる
- データ構造も曖昧なく記述される
  - 配列か非配列かを明示 (この行追記)
  - 構造体メンバのポインタについても曖昧さがない (許さない)

- コードを自動生成できる
  - 分散処理用のマーシャラ (シリアライザ) コード
  - 排他制御コード
  - セキュリティコード

©TOPPERSプロジェクト

TECS開発の第一レベルの目標

以下を実現すること

- 組込みコンポーネントモデル
- 組込みコンポーネント図
- 組込みコンポーネント記述言語
- プログラミング規則（C言語による実装のルール）

以下のサンプル実装を行うこと

- ジェネレータ
- ランタイム

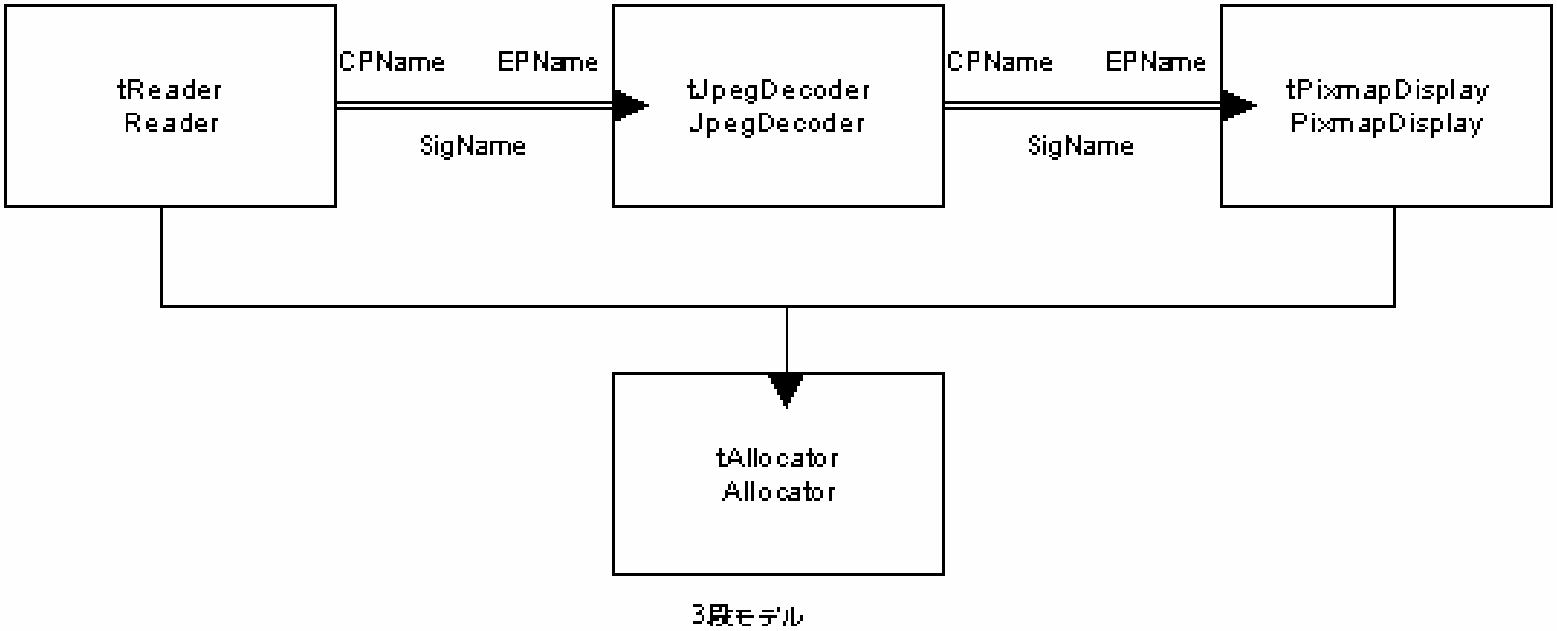
©TOPPERSプロジェクト

TECS開発の第二レベルの目標

- セル間は、関数結合を基本とする
  - 変数による結合はない（結合度の低減）
- すべてのものがコンポーネントとして扱われる
  - メモリアロケータ、RPCチャネルもコンポーネント
    - 他のコンポーネントシステムでは、そらの下に隠されている
- 静的結合を基本とする
  - オーバヘッドを減少すると共に予測性を高く保つ
- コンポーネント図により組上げられる
  - システムのアーキテクチャを可視化する
- 分散に対応したコンポーネントを標準とする
  - 再利用性の高いコンポーネント
- 多段リレーモデル を実現できる
  - 効率の高いシステムの実現

©TOPPERSプロジェクト

多段リレーモデル



3段モデル

- 一段目で確保したメモリ領域が最終段まで引き継がれる
  - 各段はonewayのシグニチャで連結された分散セル(別タスク実行)
  - メモリ獲得、解放のオーバヘッドを低減
  - ハードウェアコンポーネントとのすり替えを考慮

©TOPPERSプロジェクト

TECS開発の第二レベルの目標 (続き)

- C 言語の曖昧さをカバーする
- 関数記述には C 言語と互換性を持たせる
  - '[', ']' で囲まれた部分を取り除くと C 言語記述と一致
- ファクトリ、RPC ジェネレータは外部ツールで
  - 種々の状況に、容易に対応できる
    - TOPPERS(ITRON)以外にも対応できる
    - 種々のRPCチャネルに対応できる
    - セキュリティコンポーネントなどを呼び口と受け口の間に置くこともできる
- 多様な接続形式に対応する分散フレームワーク(RPC)
  - トランスペアレント接続 (メモリ空間共通、ポインタ共有可)
    - インタスク接続 (直接呼出し)
    - アウトタスク接続 (別タスク呼出し)
  - オペイク接続 (メモリ空間非共通、ポインタ共用不可)
    - クローズドネットワーク接続 (専用線、共有メモリ結合マルチCPU)
    - パブリックネットワーク接続 (LAN、WAN接続マルチCPU)

©TOPPERSプロジェクト

TECS開発の第二レベルの目標 (続き)

- 排他制御コードの自動生成
- ID値のチェックコードの自動生成
- ローダブルモジュール
- テスト、デバッグの支援機能
- 例外

©TOPPERSプロジェクト

Part2. TECS(TOPPERS Embedded Component System)の仕様

- TECSの用語
- 用語の説明
- 記述ファイル
- 開発作業の流れ
- TOPPERS/ASPのsyslogコンポーネント化
- 組込みコンポーネントモデル
- ロガー
- 複合コンポーネント
- 分散コンポーネント

©TOPPERSプロジェクト

TECSの用語

斜字体は CDL **のキーワード,後ろは名前の慣例**(接頭辞の付け方)
下段:他でよく用いられているコンポーネントシステムの用語
(CDL: Component Description Language)

- **セル** *cell* cellName (**接頭辞なし、小文字で始める**)
  - **コンポーネント(オブジェクト、インスタンス)**
- **セルタイプ** *celltype* tCellType
  - **コンポーネントクラス**
- **シグニチャ** *signature* sSignature
  - **インタフェース**
- **呼び口** *call* cPort
  - (**クライアントスタブ**)
- **受け口** *entry* ePort
  - (**サーバースタブ**)
- **複合セル** *composite* tComposite

©TOPPERSプロジェクト

用語の説明

- セル
  - コンポーネント（インスタンス）のことで受け口、呼び口、属性を持つ
- 受け口
  - 外部に対して機能を提供する窓口
  - 機能の提供は、関数頭部の集合により定義される。これを「シグニチャ」と呼ぶ
- 呼び口
  - セルが他のセルの提供する機能を利用するための窓口であり、呼び口を受け口に結合して使用する
  - 呼び口も利用する機能に応じてシグニチャに対応付けられており、シグニチャの一致する呼び口と受け口のみを結合ずることができる
- セルタイプ
  - 同じ呼び口、受け口、属性を持ち同じ振舞いをするセルの総称。セルの振舞いを記述するコードはセルタイプに対して作成

©TOPPERSプロジェクト

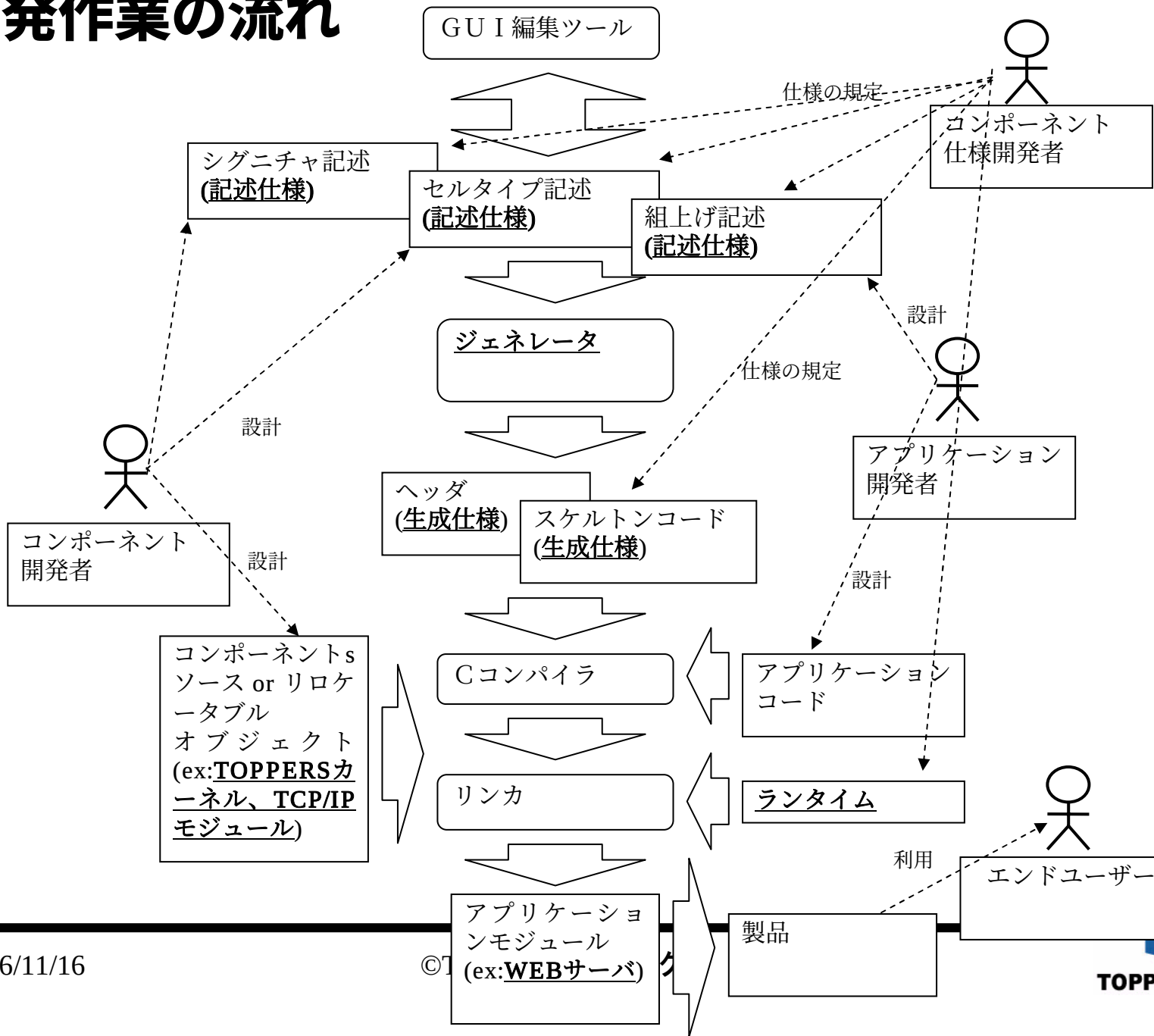
記述ファイル

コンポーネント記述ファイル CDL は4つの部分がある

- 型・定数記述
  - typedef, struct, enum, const を記述
- シグニチャ記述
  - シグニチャの定義を記述
- セルタイプ記述
  - セルタイプの定義を記述
  - 複合コンポーネントの定義を記述
- 組上げ記述
  - セルの定義を記述
    - 呼び口と受け口の結合の記述

©TOPPERSプロジェクト

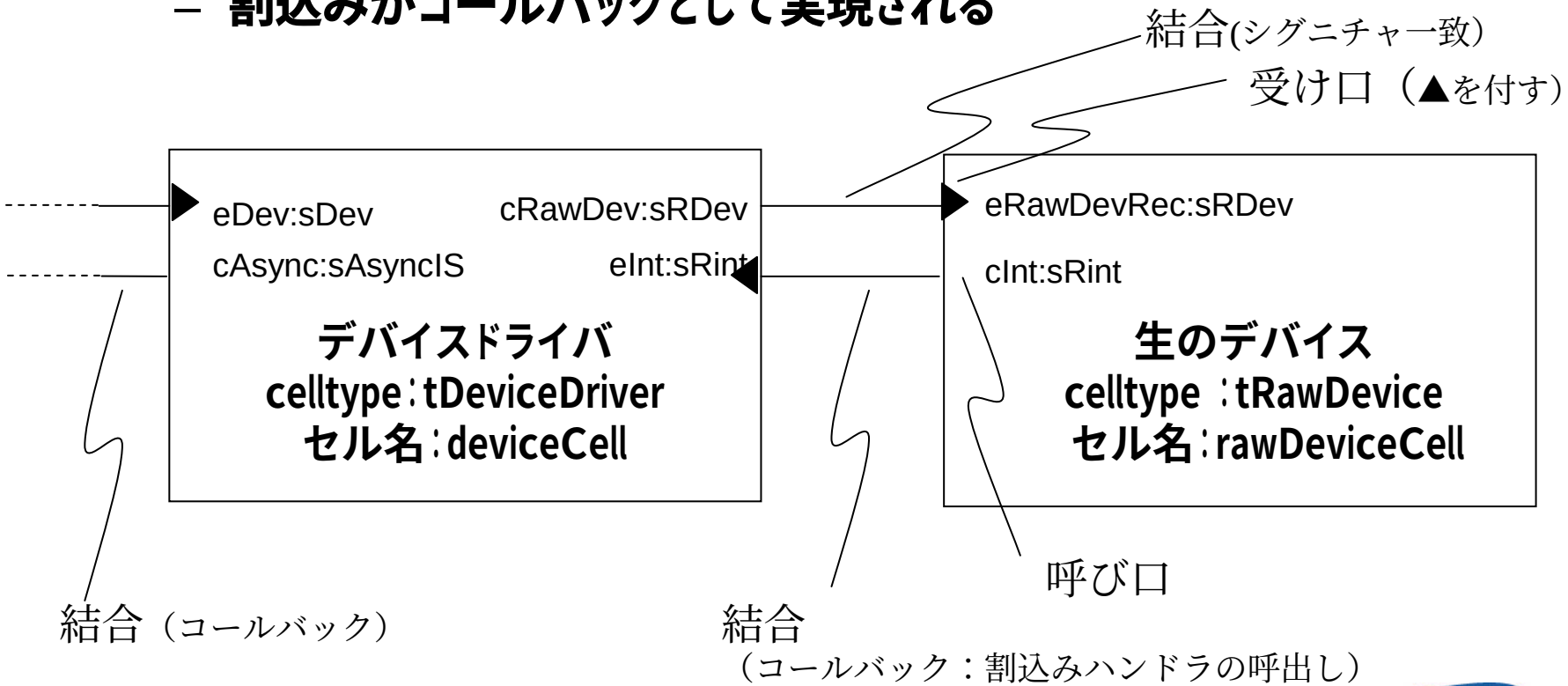
開発作業の流れ


GUI編集ツール
仕様の規定
コンポーネント
仕様開発者
シグニチャ記述
(記述仕様)
セルタイプ記述
(記述仕様)
組上げ記述
(記述仕様)
設計
ジェネレータ
仕様の規定
設計
アプリケーション
開発者
ヘッダ
(生成仕様)
スケルトンコード
(生成仕様)
コンポーネント
開発者
設計
設計
コンポーネントs
ソース or リロケ
ータブル
オブジェクト
(ex:TOPPERSカ
ーネル、TCP/IP
モジュール)
Cコンパイラ
アプリケーション
コード
リンカ
ランタイム
利用
エンドユーザー
アプリケーショ
ンモジュール
(ex:WEBサーバ)
製品

©T

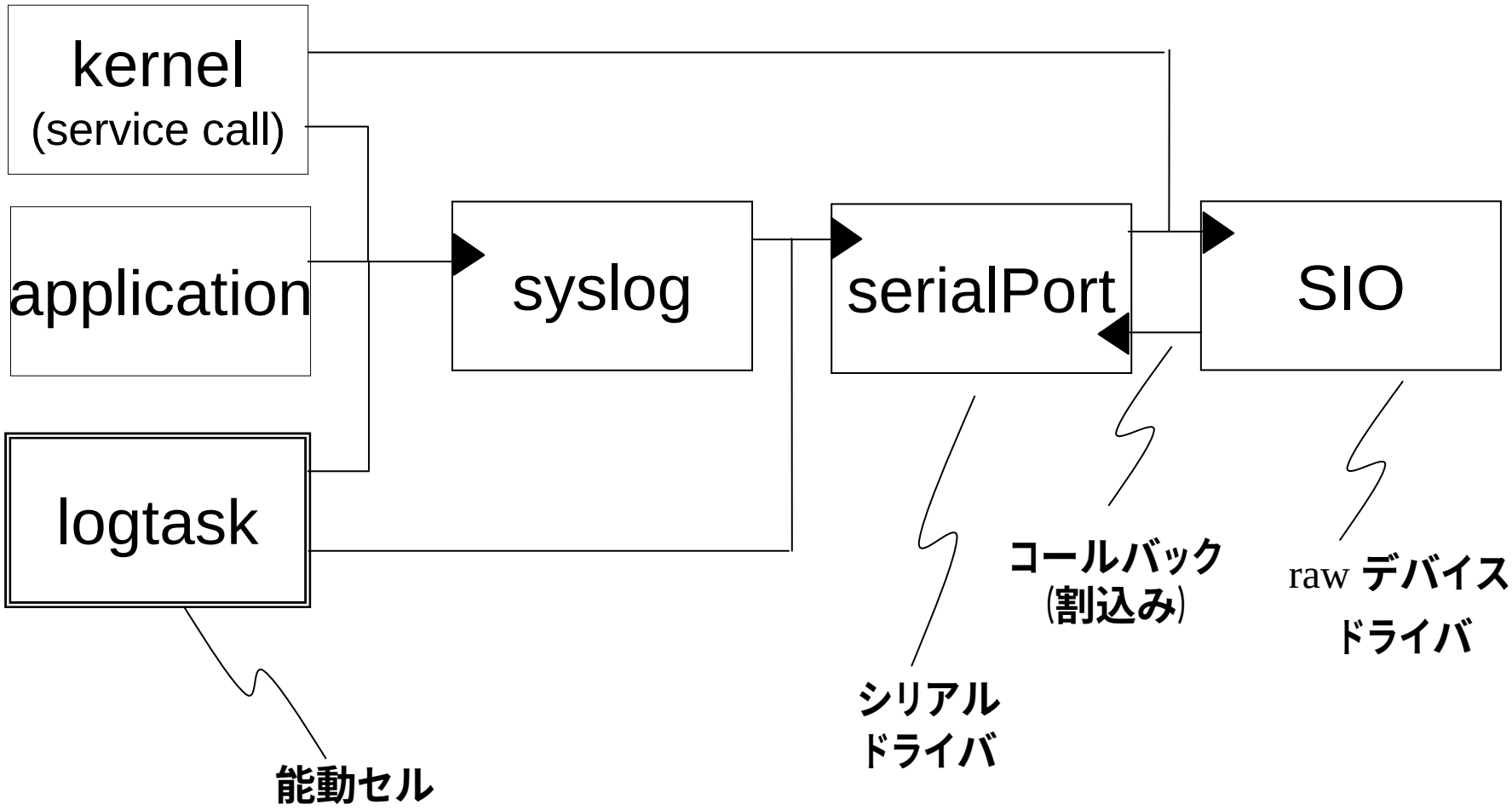
組込みコンポーネントモデル

- デバイスドライバの結合
  - 矢線は呼出しを表す
    - 呼出し可能関数がシグニチャとして定義される
  - 割込みがコールバックとして実現される



©TOPPERSプロジェクト

TOPPERS/ASPのsyslogのコンポーネント化

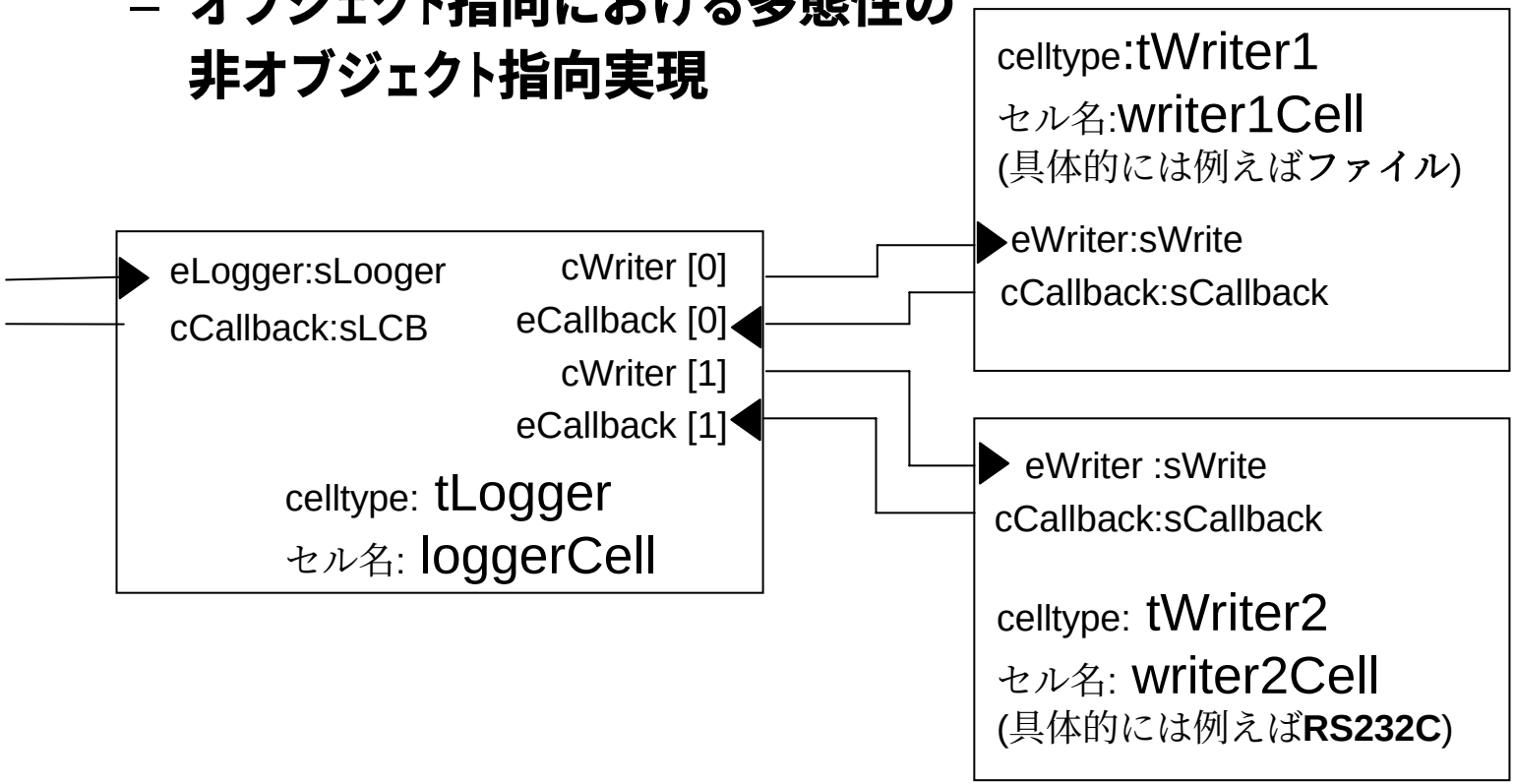


©TOPPERSプロジェクト

ロガーのコンポーネントモデル

(前のスライドのASPのsyslogとは関係ありません)

- ログを複数の出力先に出力するコンポーネント
- 呼び口配列　・受け口配列を使用
  - 複数の出力先から選択可能
  - オブジェクト指向における多態性の非オブジェクト指向実現



©TOPPERSプロジェクト

ロガーのシグニチャ

以下前のスライドのロガーを例にコンポーネント記述を説明

- シグニチャ
  - 提供する機能を関数頭部の集合として定義
  - 分散対応するためにはERを返すべき (分散フレームワークからのエラーが ER として返される)

```
typedef  int32 ER;              /* 型定義:int32 はTECS組込み型 */
typedef  int16 wchar_t;
signature sLogger {
  ER initialize(void);
  ER writeString([in,string] const wchar_t *str);
  ER quit (void);
};
```


©TOPPERSプロジェクト

ロガーのセルタイプ

- セルタイプ　ロガーセルの持つ外部インタフェースの定義
  - 以下の要素からなる

    呼び口、受け口、属性

```
celltype  tLogger {
  call sWriter cWriter [];
  entry sCallback eCallback [];
  entry sLogger eLogger;
  call sLCB eCallback;
  attribute{
    int32  initialOutput = 0;
  }
};
```


©TOPPERSプロジェクト

ロガーの組上げ

- 組上げ
  - セルの定義
    - セルの呼び口の接続先の受け口の指定
    - 属性値の指定

```
cell tLogger loggerCell {
  cWriter [0] = writerCell1.eWriter;
  cWriter [1] = writerCell2.eWriter;
  cCallback = mainCell.eCallback;    /* 図示せず */
  initialOutput = 0
};
```


©TOPPERSプロジェクト

ジェネレータ出力

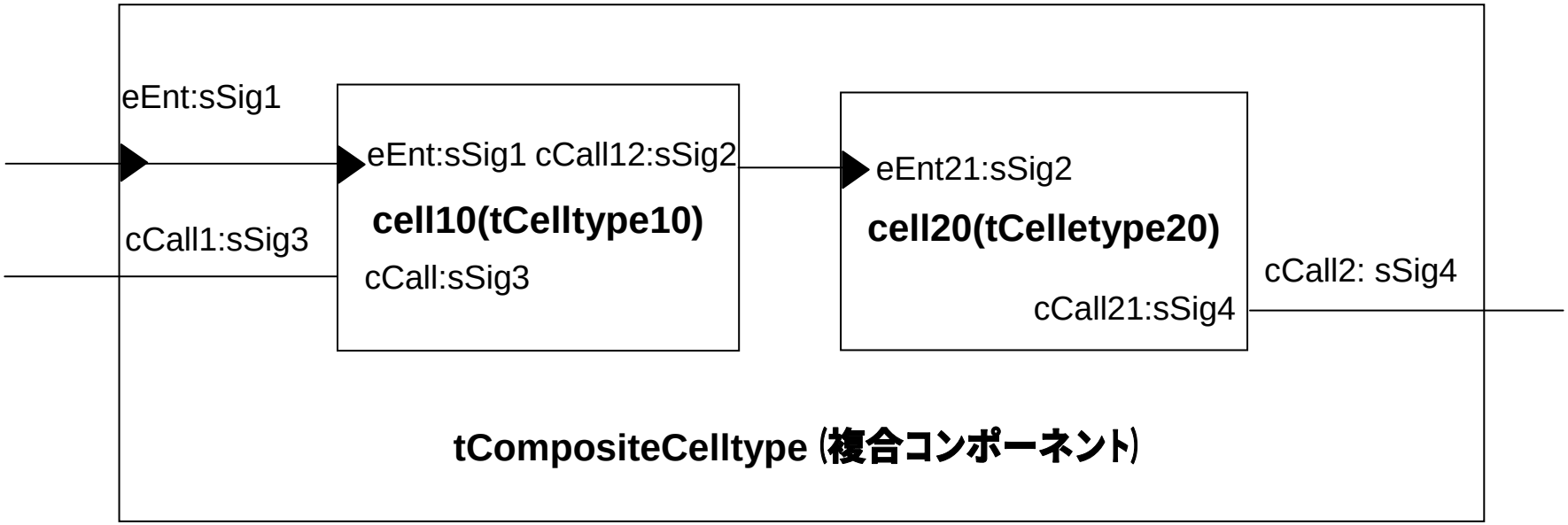
記述ファイルをジェネレータに通すと以下のファイル（Cのプログラム）が生成される。これと種のコードをコンパイル、リンクしてアプリ作成

- グローバルヘッダ global_tecsgen.h
  - 型、構造体、定数定義
- シグニチャヘッダ *sSigname*_tecsgen.h
  - シグニチャ（関数テーブル）の型定義
- セルタイプヘッダ *tCelltype*_tecsge.h
  - セルのCB(Control Block)の型定義
  - 受け口ディスクリプタ型定義
  - 呼び口関数のマクロ定義 etc
- セルタイプコード *tCelltype*_tecsgen.c
  - セルのCBの定義
  - 受け口スケルトン関数
- セルタイプテンプレートコード *tCelltype*_template.c
  - これをベースにセルタイプ実装コードを記述する

複合コンポーネント

以下、その他のコンポーネントについて説明

- 複合コンポーネント：複数のセルを組合わせて一つの新しいセルタイプを作ることができる



©TOPPERSプロジェクト

複合コンポーネントの記述

- **複合コンポーネントのコンポーネント記述**

```
Composite tCompositeCelltype {
    cell tCelltype20 cell20 {
    };
    cell tCelltype10 cell10 {
       cCall12 = cell20.eEnt21;
    };
    eEnt = cell10.eEnt;
    cCall1 = cell10.cCall;
    cCall2 = cell20.cCall21;
};
Cell tCompositeCelltype {
   cCall1 = …;    /* sSig3 の受け口 */
   cCall2 = …;    /* sSig4 の受け口 */
};
```


©TOPPERSプロジェクト

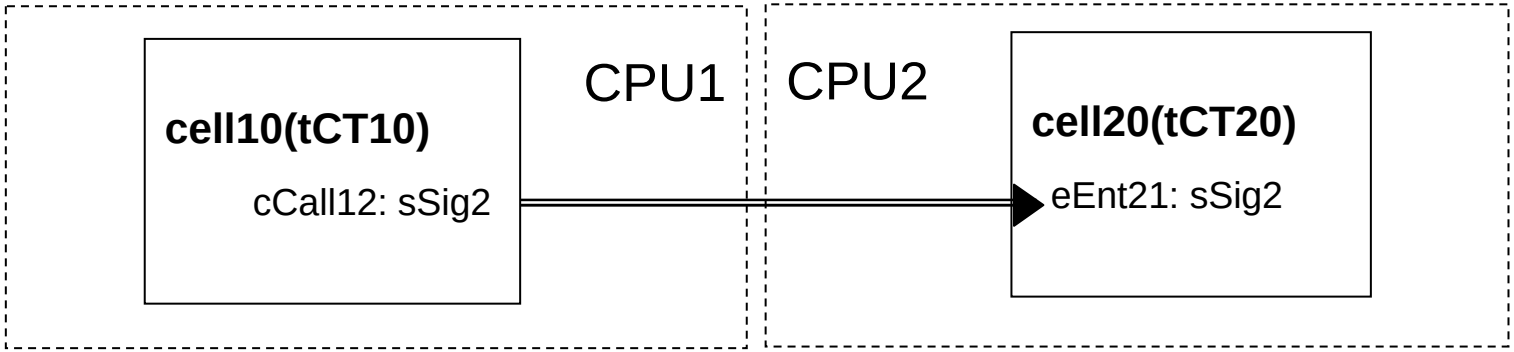
分散コンポーネント

- 別タスク、別プロセッサに存在するコンポーネントを関数呼出しにより扱う手段を提供
  - 分散コンポーネント開発フレームワークは、分散アプリケーションの構築、改変を容易にする
    - 設計の自動化による生産性の向上
- 通信チャネル自体がコンポーネントである
  - 種々の通信チャネルに対して適応性がよい
    - メッセージボックス、FIFO、共有メモリ、TCP/IPネットワーク
- 種々の接続形態に対応(再掲)
  - トランスペアレント接続 (メモリ空間共通、ポインタ共有可)
    - インタスク接続 (直接呼出し)
    - アウトタスク接続 (別タスク呼出し)
  - オペイク接続 (メモリ空間非共通、ポインタ共用不可)
    - クローズドネットワーク接続 (専用線、共有メモリ結合マルチCPU)
    - パブリックネットワーク接続 (LAN、WAN接続マルチCPU)

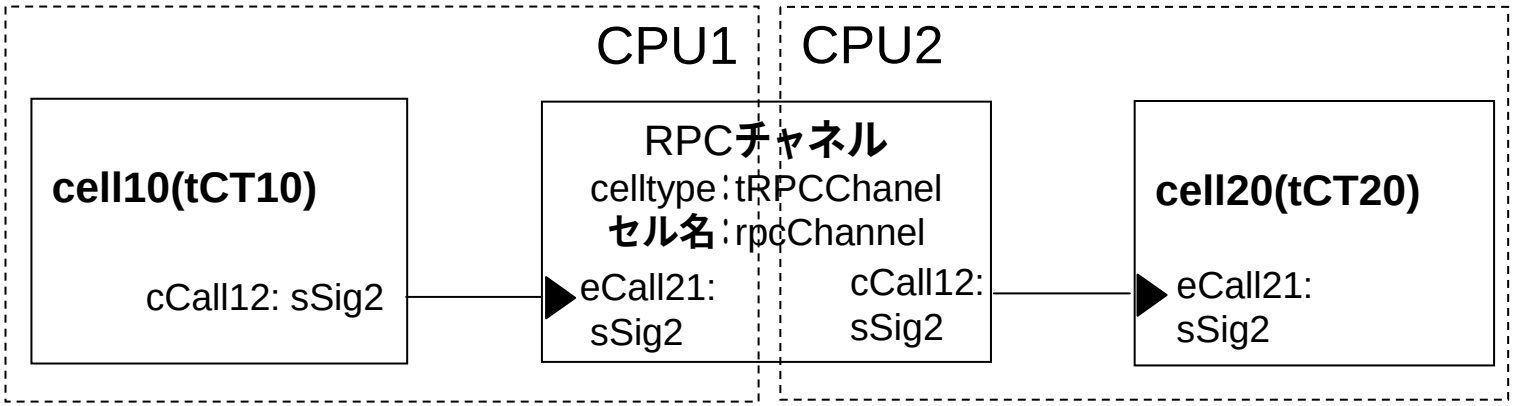
©TOPPERSプロジェクト

分散コンポーネント図

- **分散コンポーネントの接続** (RPCを介してつなぐ場合「接続」と呼ぶ)



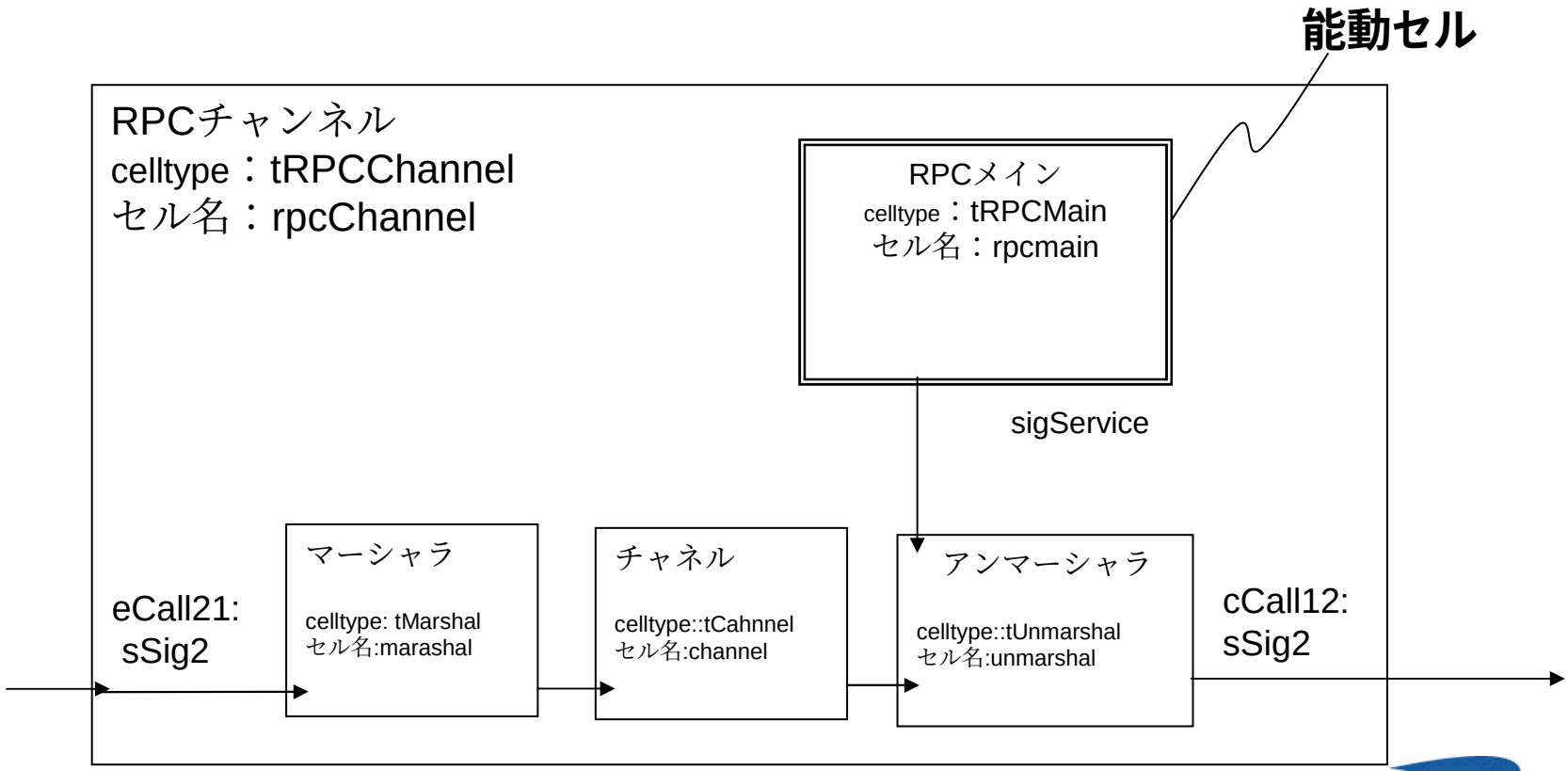
- **実際の結合**



©TOPPERSプロジェクト

RPCチャンネルコンポーネント

- RPCチャンネルコンポーネントは次のような複合コンポーネントとして実現される



©TOPPERSプロジェクト

応用

– 次のようなものに応用されることを考えている

- 各種デバイスドライバおよびプロトコルスタック
  - USB, IEEE1394, CAN, LIN, Bluetooth
  - TCP/IP + Ethernet/PPP/無線LAN
- 共有メモリを介して接続されるマルチプロセッサシステム内の分散フレームワーク
  - FIFOであったり通信であってもよい
- ファイルシステム
- GUIコンポーネント

©TOPPERSプロジェクト

コンポーネント仕様WGの活動の紹介

- 毎月一回名古屋でミーティングを開催
  - 2004年1月より毎月開催
  - 組込みコンポーネントWGメーリングリストにて案内中
- TOPPERS組込みコンポーネントシステムTECS仕様の策定作業およびサンプル実装作業進む
  - TOPPERS/ASP向けを優先して実装中
- wikiにより仕様を構築中
  - まだまだ議論白熱中
  - 収束する方向には向かっている
- まだ議論にあがらない課題も豊富
  - 次ページのような

©TOPPERSプロジェクト

これからの課題

未検討で、面白そうな課題がまだまだ多数ある(その前にやることも多数)

- ハードウェアとソフトウェアの協調設計
  - 短納期化のためにハードウェアができる前にソフトウェアを開発（完了）する必要がある
  - 複雑化するシステムの全体を見通すことが難しくなり、最適な負荷分散、最小の製造コスト、最小の開発コスト、最良の性能、最短納期を同時実現することが難しくなっている

    ↓ ↓ ↓ ↓
  - ハードウェアとソフトウェアのコンポーネントの交換容易性
- 高級言語化による開発効率の改善
  - 組込みソフトウェア開発の言語はアセンブラからCへ移った
  - それ以上に高級な言語への移行はまだ進んでいない

    ↓ ↓ ↓ ↓
    - C++のような高機能言語への移行
    - Java, JavaScriptのように平易な言語への移行
- もちろん従来からの問題もある
  - リソース制約、時間制約を満たさなくてはならない

©TOPPERSプロジェクト