アスペクト指向アーキテクチャに基づく組込みソフトウェアの派生開発

2010.6.18

派生開発カンファレンス2010

南山大学情報理工学部 ソフトウェア工学科

野呂 昌満

Masami STM Noro, Nanzan Univ. Dept. Software Eng.

はじめに

- 組込みソフトウェアの開発について20年位かなりまじめに考えてきた.

- PLSEによる開発支援が多くの問題を解決すると考えている.

- PLSEEを試作し開発に応用.

- 一定の成果を確認.

経緯

20年来の企業との共同研究•開発.

- OOによる整理(分析, 設計, 再開発)
- ソフトウェアプロセス
- ソフトウェアアーキテクチャ
- アスペクト指向
- AOアーキテクチャに基づくPLSE

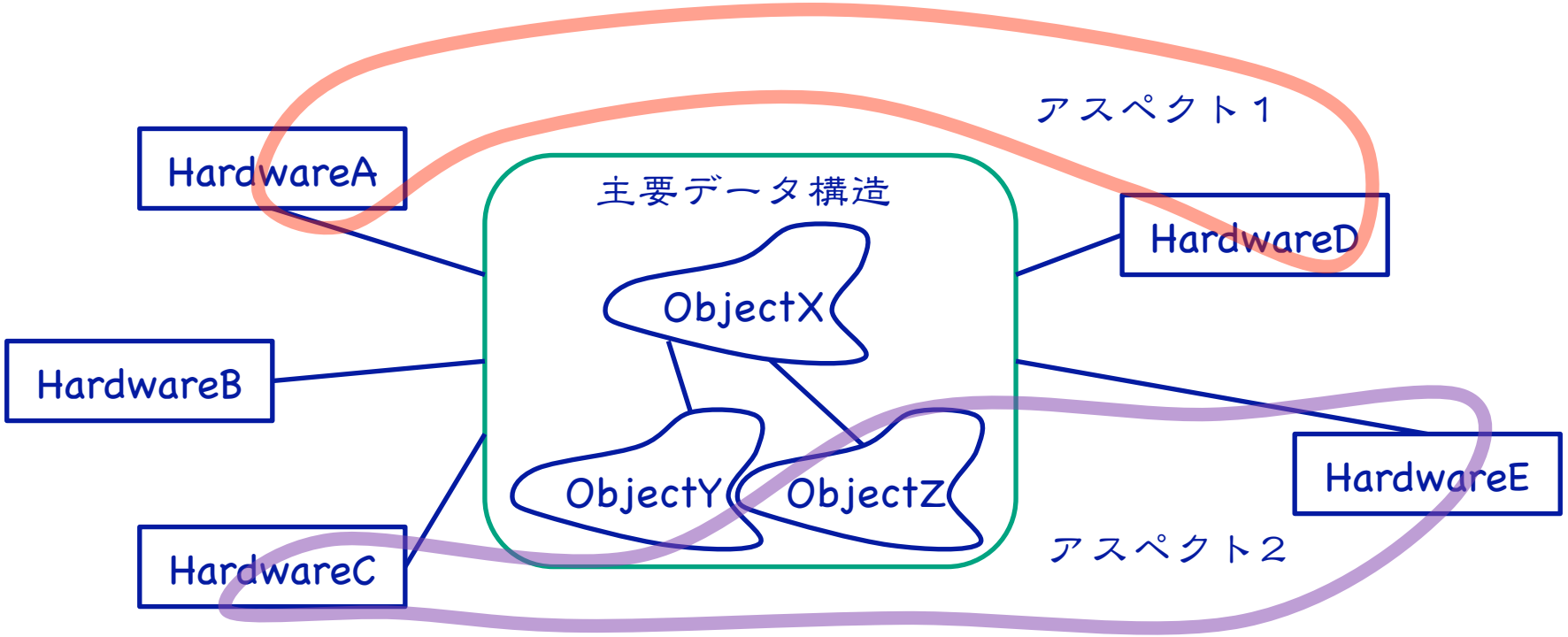
現時点における解

- E-AoSAS++(組込みソフトウェアのためのアスペクト指向ソフトウェアアーキテクチャスタイル)
- PLSE on E-AoSAS++

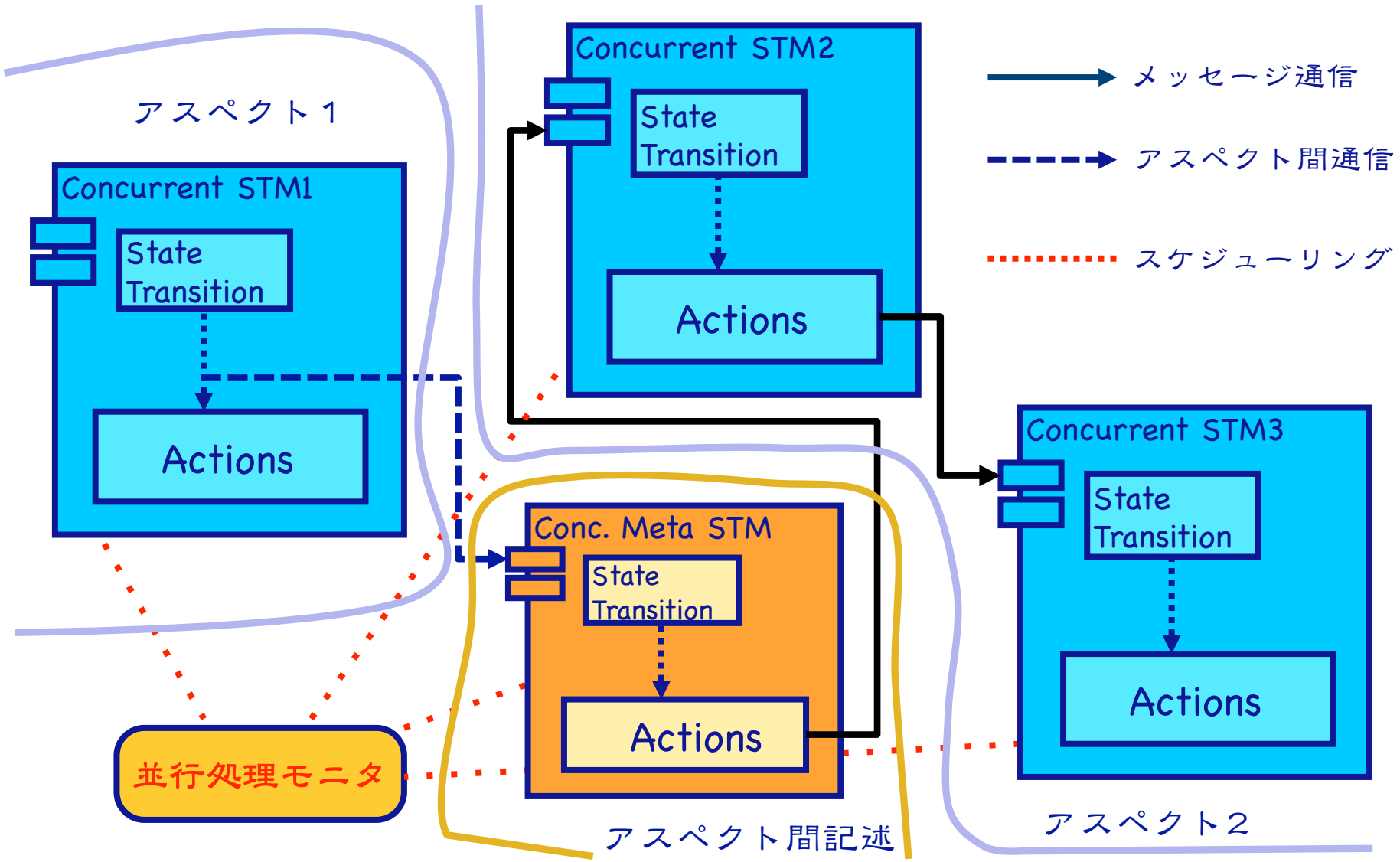
- PLEASE(PLSE Environment based on Aspect-Oriented Software Architecture Style for Embedded Systems)

組込みソフトウェアのアーキテクチャスタイル

- 並行に稼働するハードウェアを状態遷移機械でモデル化
- 主要データ構造をオブジェクト指向でモデル化
- 実時間，耐故障性等のアスペクトとの共存をモデル化



状態遷移機械によるモデル化 - E-AoSAS++ -



何故アスペクトか? - E-AoSAS++ -

いろいろ悩んだ結果...

- 並行性,
- 状態遷移,
- 耐故障性,
- 実時間性,
- エラー処理,

などの横断的関心事を認識.

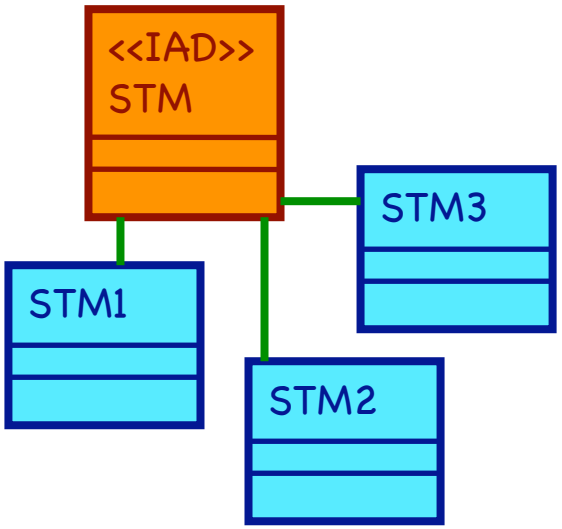
アーキテクチャ構成要素: E-AoSAS++

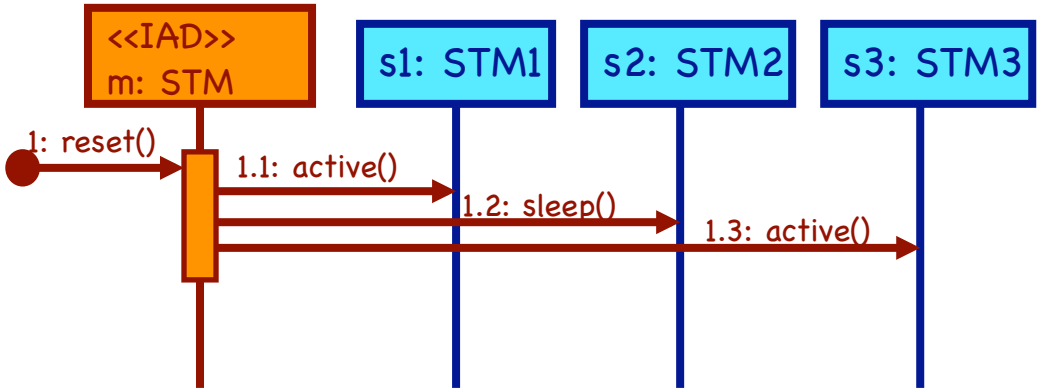
- System
  - 製品全体
- View
  - 横断的関心事
- Aspect
  - 関心事によって分割されたモジュール
  - PrimitiveCSTMの集合
- IAD(アスペクト間記述)
  - CSTM同士の粗結合通信の記述
- CSTM （並行状態遷移機械）
  - 基本構成要素
    - PrimitiveCSTM
  - MetaCSTM
  - CompositeSTM
- Action
  - CSTM同士の密結合通信の記述

UMP(統一モジュール化パターン)- E-AoSAS++ -

ベース状態遷移機械の集合をメタ状態遷移機械が管理する.

- 状態遷移機械の集合でアスペクトを構成. メタ状態遷移機械がアスペクト間記述.
- 状態遷移機械の集合でオブジェクト集約を構成. メタ状態遷移機械が集約を管理.





UMLによるアーキテクチャ記述

- Component Diagram
  - モジュール構成，コンポーネント(CSTM)の構成関係記述
- Class Diagram
  - 各関心事（View）におけるコンポーネントの関連
- State Machine Diagram
  - CSTMの仕様
    - 状態遷移
    - アクション
- Sequence Diagram
  - アスペクト間記述
- Object Diagram
  - CSTMの実体の数とその関連

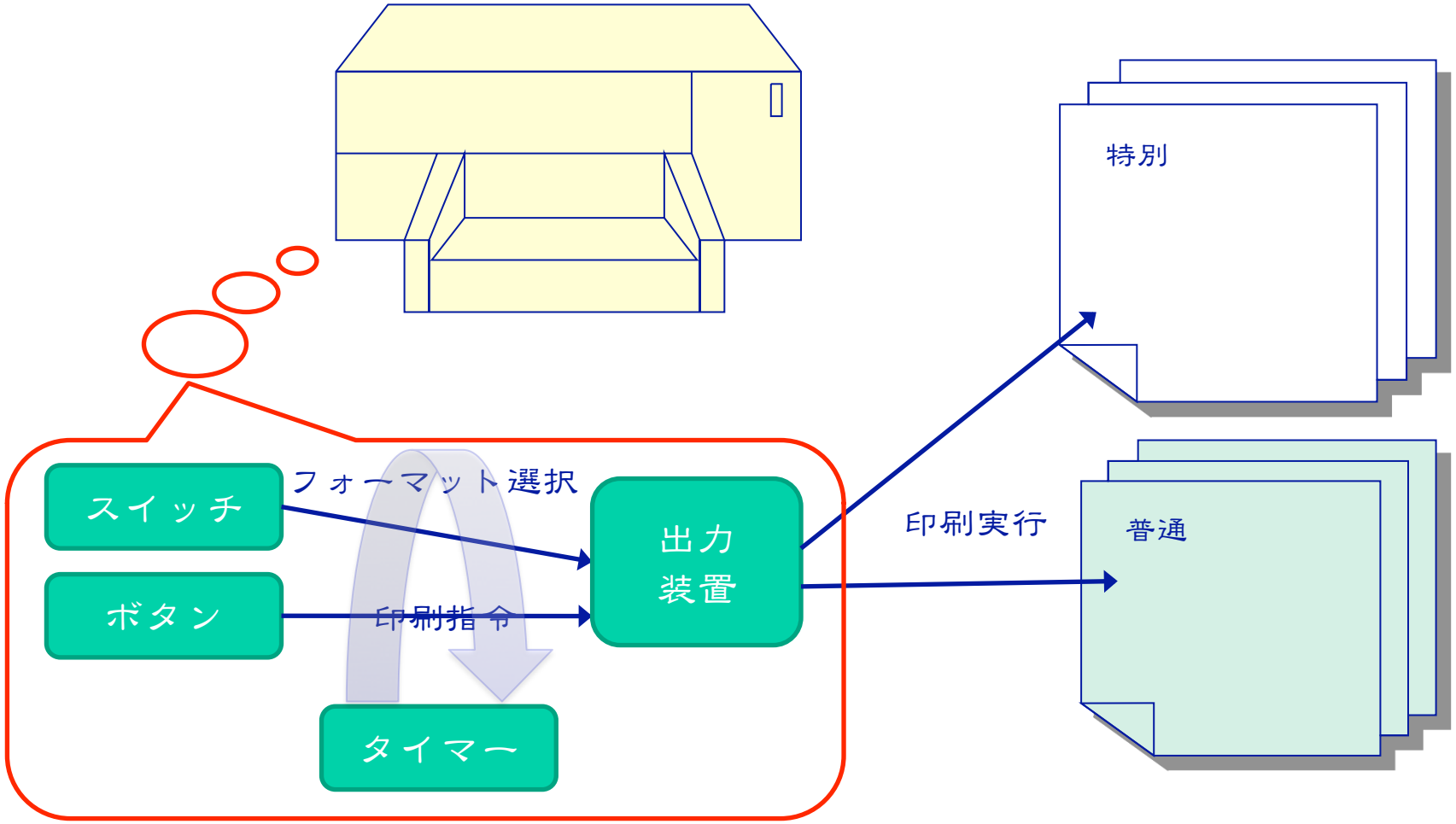
アスペクトとアスペクト間記述

- アスペクトは状態遷移機械の集合.
- アスペクト内の状態戦機械同士は密結合.
- アスペクト間記述はAspectJ方式を拡張し採用. (イベント駆動型システムと親和性のある記述方式)
  - 合流点(Weaving Policy)
  - 合流点のアドバイス(拡張, JP_Advice)
  - ポイントカット: Aspect Coordinator(MetaSTM)
  - アドバイス: AC_Advice(Aspect Coodinator's Action)

説明に使う簡単な例

- プリンタシステム
- スイッチによって印刷の型式を変更
- 型式には普通と特別があり，初期型式は普通
- ボタンを押すとデータを整形し印刷
- スイッチで型式を設定後，5秒以内にボタンを押さなければ型式は普通に復帰
- 5秒以内にボタンが押されたら，印刷後型式を普通に戻す
- スイッチが押されるかタイムアウトによる型式の切り替え中は，ボタンを無視
- 印刷中は，スイッチやタイムアウトによる型式の切り替えは行わない
- 印刷中はボタンを無視する
- 印刷中は，ボタンが押されてからプリントアウトされるまでを指す
- 印刷終了直後の型式は普通（特別型式でも印刷終了直後に普通型式に戻る）

説明に使う簡単な例：事例イメージ図

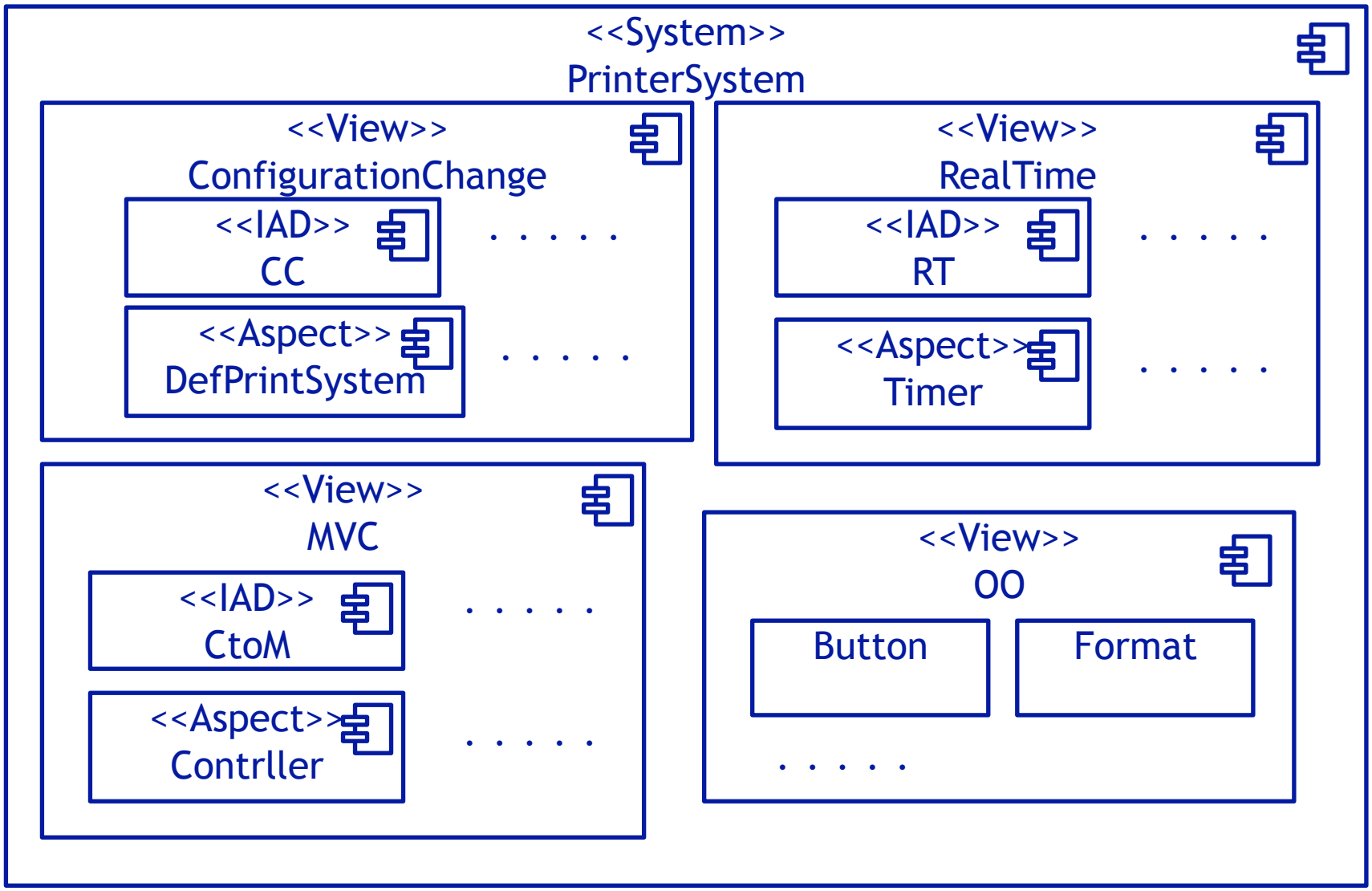


組込みソフトウェアのアーキテクチャ

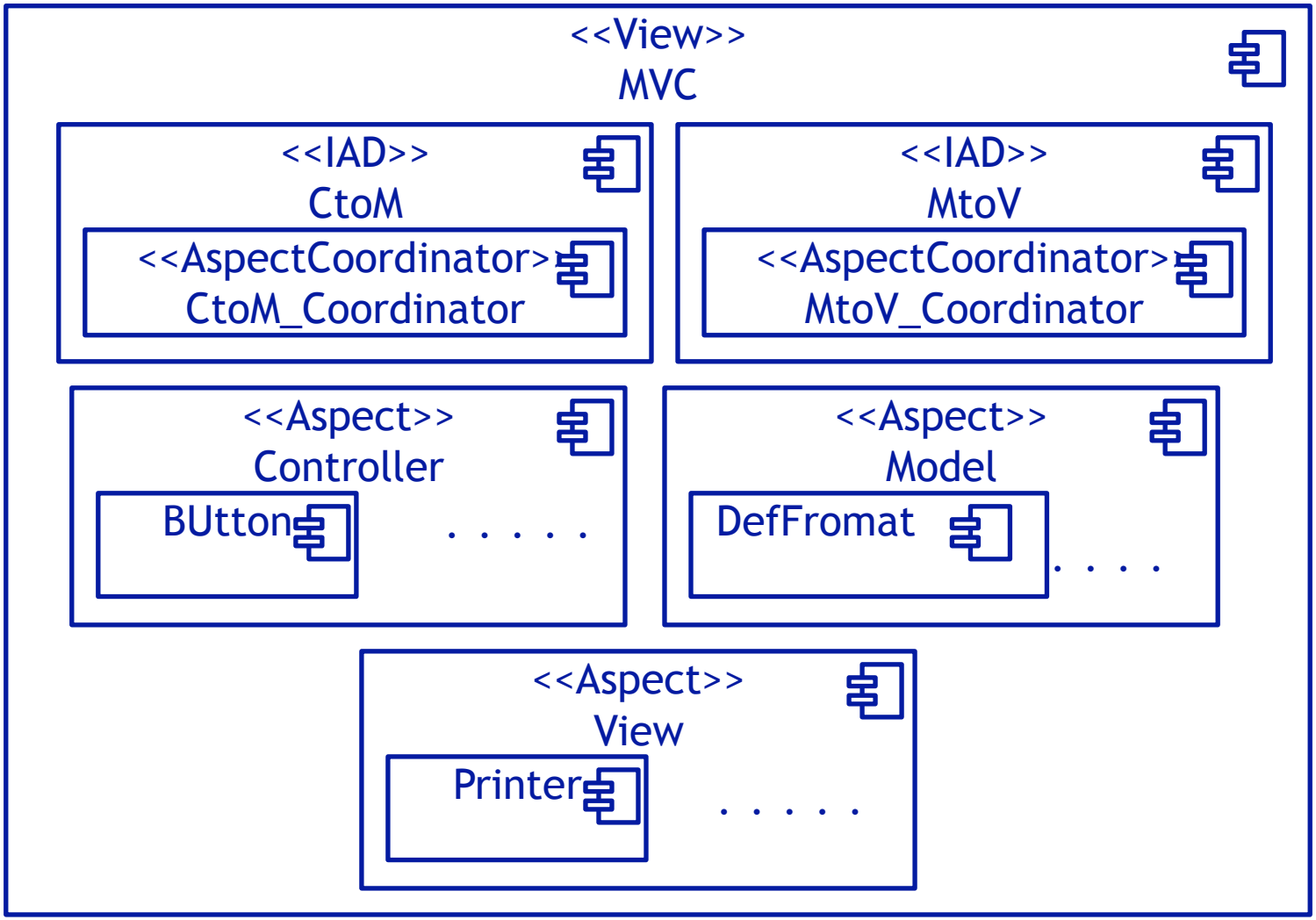
- 並行に稼働するハードウェアを状態遷移機械でモデル化
  - スイッチ，ボタン，プリンタ
- 主要データ構造をオブジェクト指向でモデル化
  - フォーマット
- 実時間，耐故障性等のアスペクトとの共存をモデル化
  - 実時間アスペクト

- 以上をMVCでモデル化

System

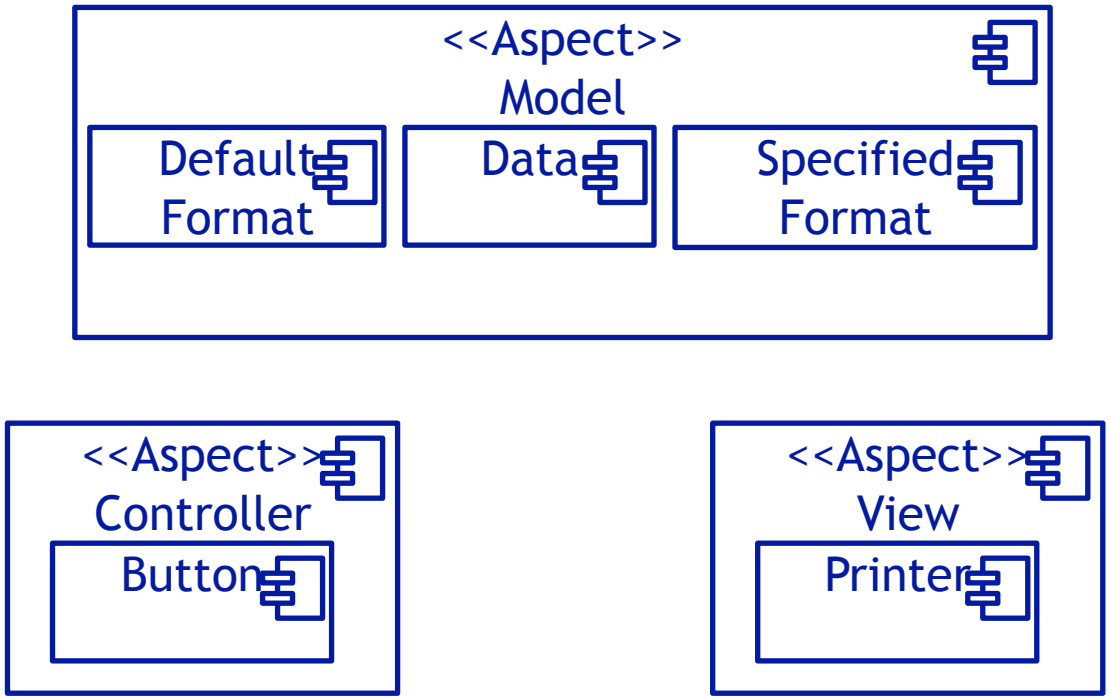

<<System>>
PrinterSystem
<<View>>
ConfigurationChange
<<IAD>>
CC
<<Aspect>>
DefPrintSystem
<<View>>
RealTime
<<IAD>>
RT
<<Aspect>>
Timer
<<View>>
MVC
<<IAD>>
CtoM
<<Aspect>>
Contrller
<<View>>
OO
Button
Format

View

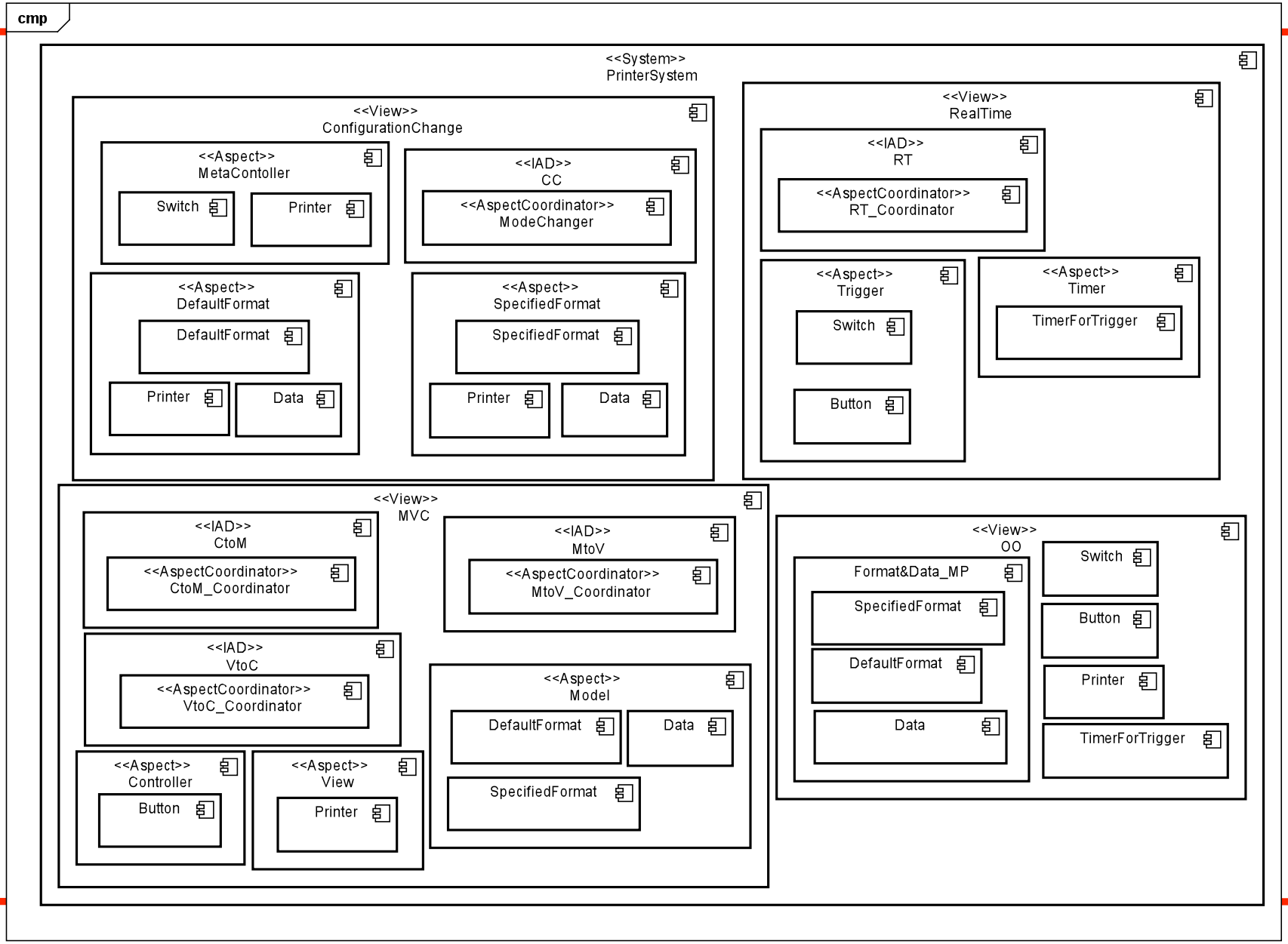

<<View>>
MVC
<<IAD>>
CtoM
<<AspectCoordinator>>
CtoM_Coordinator
<<IAD>>
MtoV
<<AspectCoordinator>>
MtoV_Coordinator
<<Aspect>>
Controller
BUtton
. . . . .
<<Aspect>>
Model
DefFromat
. . . .
<<Aspect>>
View
Printer
. . . . .

Aspect

View MVC内の各アスペクト

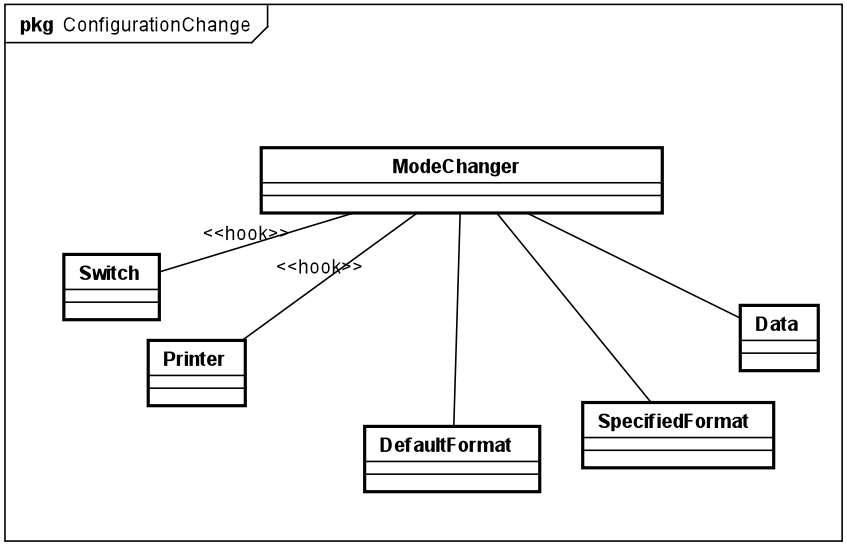


コンポーネント図（全体）

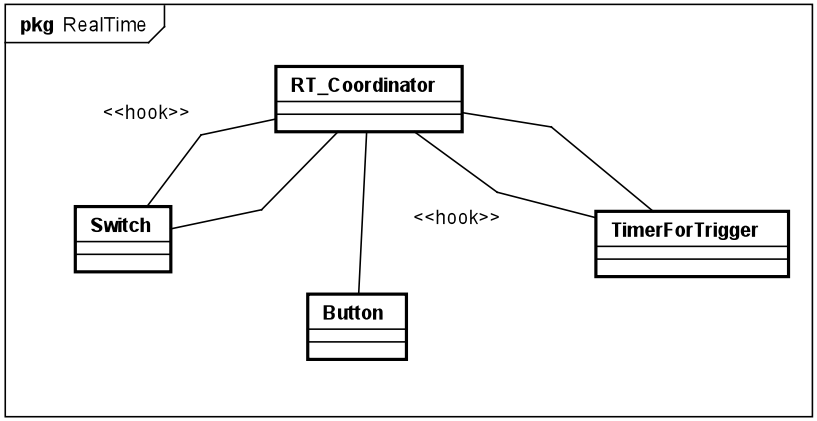

cmp
<<System>>
PrinterSystem
<<View>>
ConfigurationChange
<<Aspect>>
MetaContoller
Switch
Printer
<<IAD>>
CC
<<AspectCoordinator>>
ModeChanger
<<Aspect>>
DefaultFormat
DefaultFormat
Printer
Data
<<Aspect>>
SpecifiedFormat
SpecifiedFormat
Printer
Data
<<View>>
RealTime
<<IAD>>
RT
<<AspectCoordinator>>
RT_Coordinator
<<Aspect>>
Trigger
Switch
Button
<<Aspect>>
Timer
TimerForTrigger
<<View>>
MVC
<<IAD>>
CtoM
<<AspectCoordinator>>
CtoM_Coordinator
<<IAD>>
MtoV
<<AspectCoordinator>>
MtoV_Coordinator
<<IAD>>
VtoC
<<AspectCoordinator>>
VtoC_Coordinator
<<Aspect>>
Model
DefaultFormat
Data
SpecifiedFormat
<<Aspect>>
Controller
Button
<<Aspect>>
View
Printer
<<View>>
OO
Format&Data_MP
SpecifiedFormat
DefaultFormat
Data
Switch
Button
Printer
TimerForTrigger

クラス図

View:ConfigurationChange

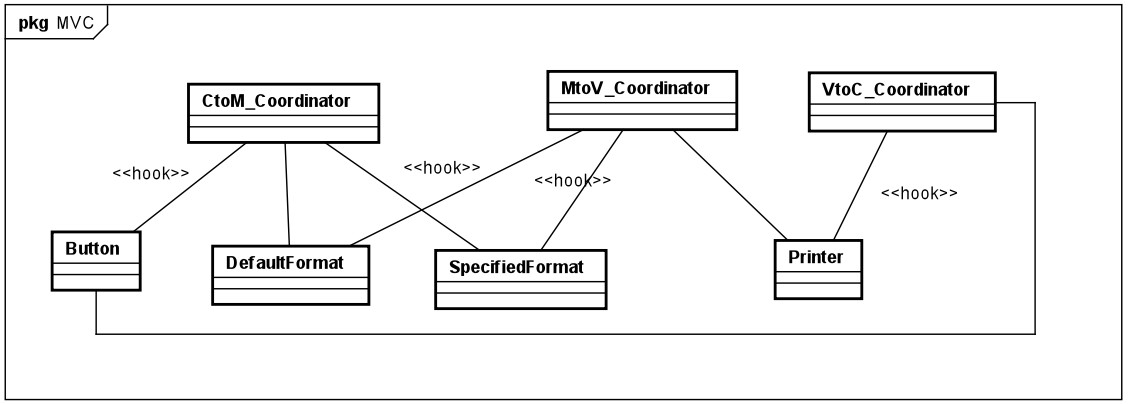


View:RealTime

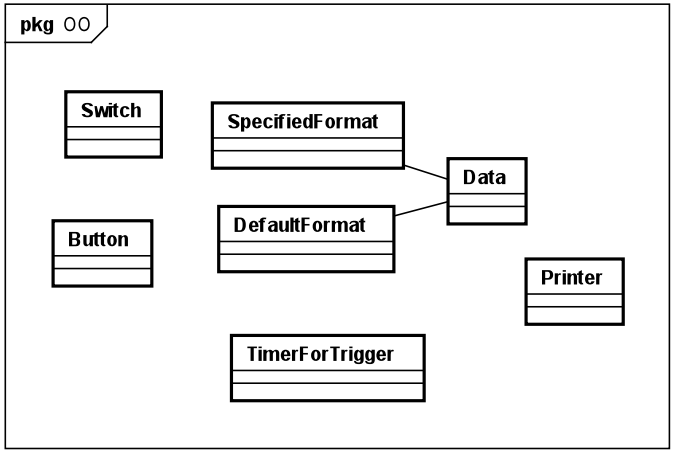


Component Relation in each View

View:MVC

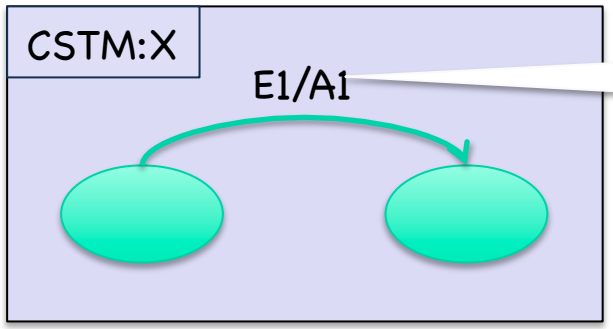


View:OO

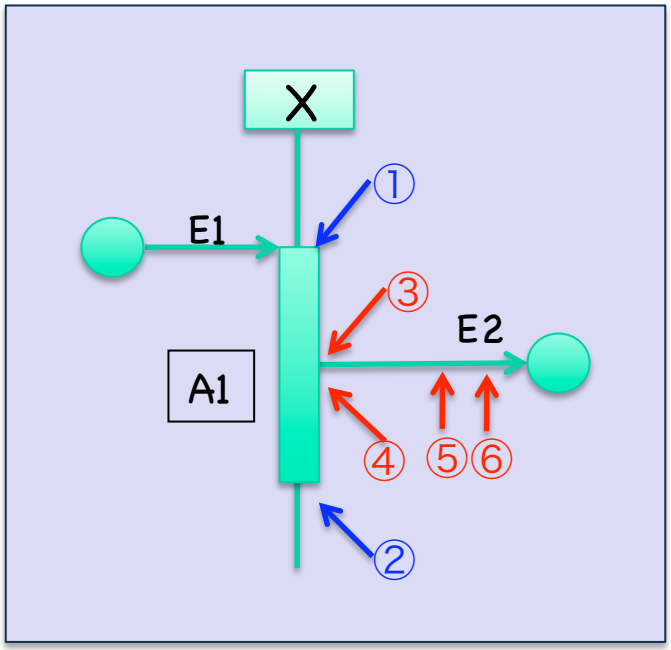


IAD （記述法）

XのE/A図



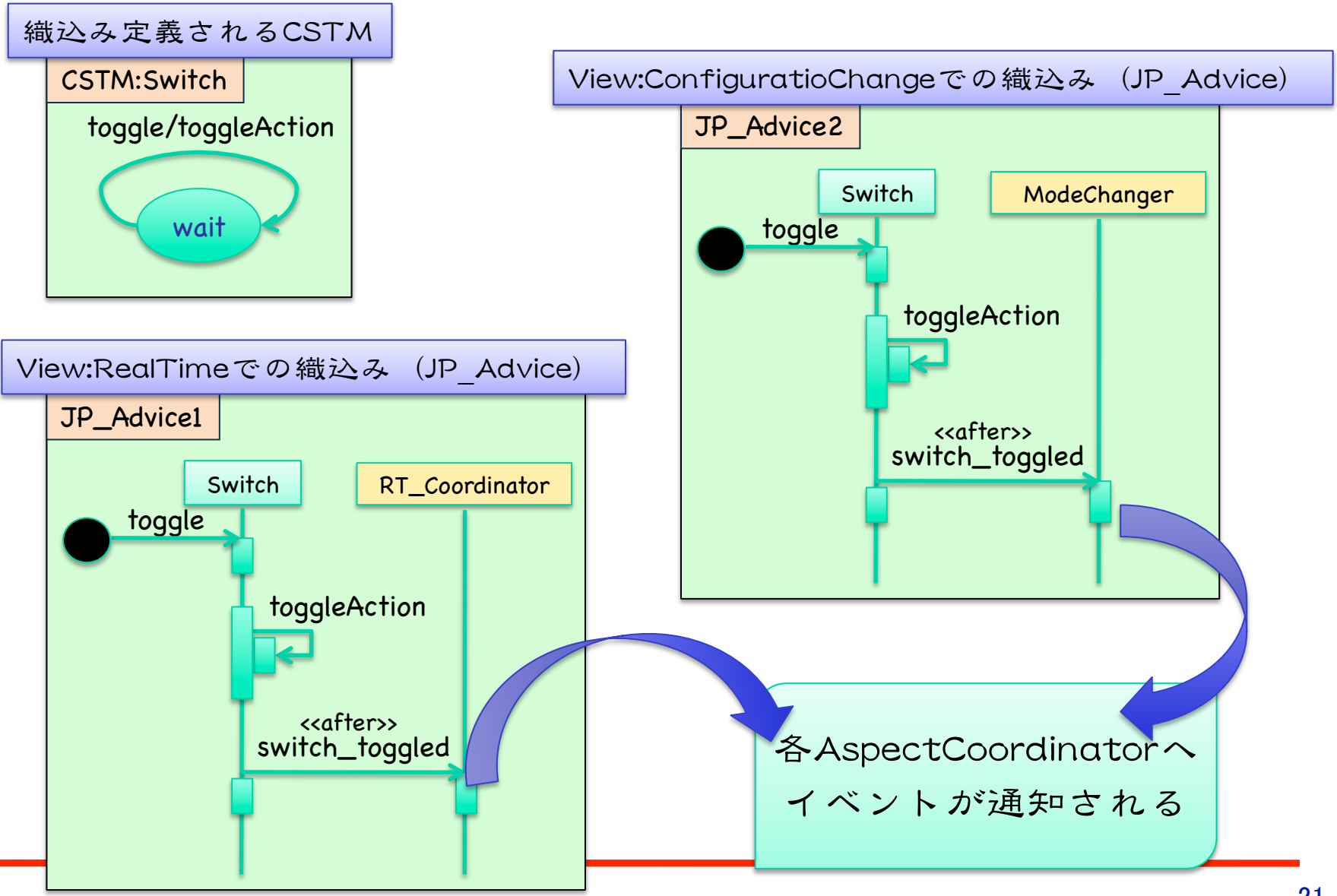
A1アクションでE2を送信するとする



イベント受理によって実行されるアクション
① <<Before Action>>
② <<After Action>>

イベント生起
③ <<Before MP>>
④ <<After MP>>
⑤ <<MP Flow>>
⑥ <<Around MP>>

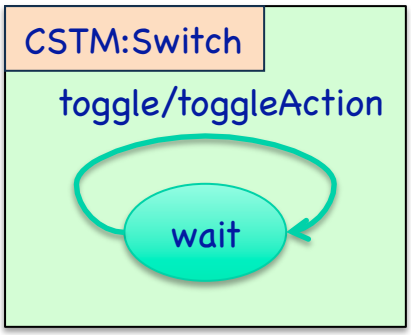
IADの記述例（アクションに対する織込み）



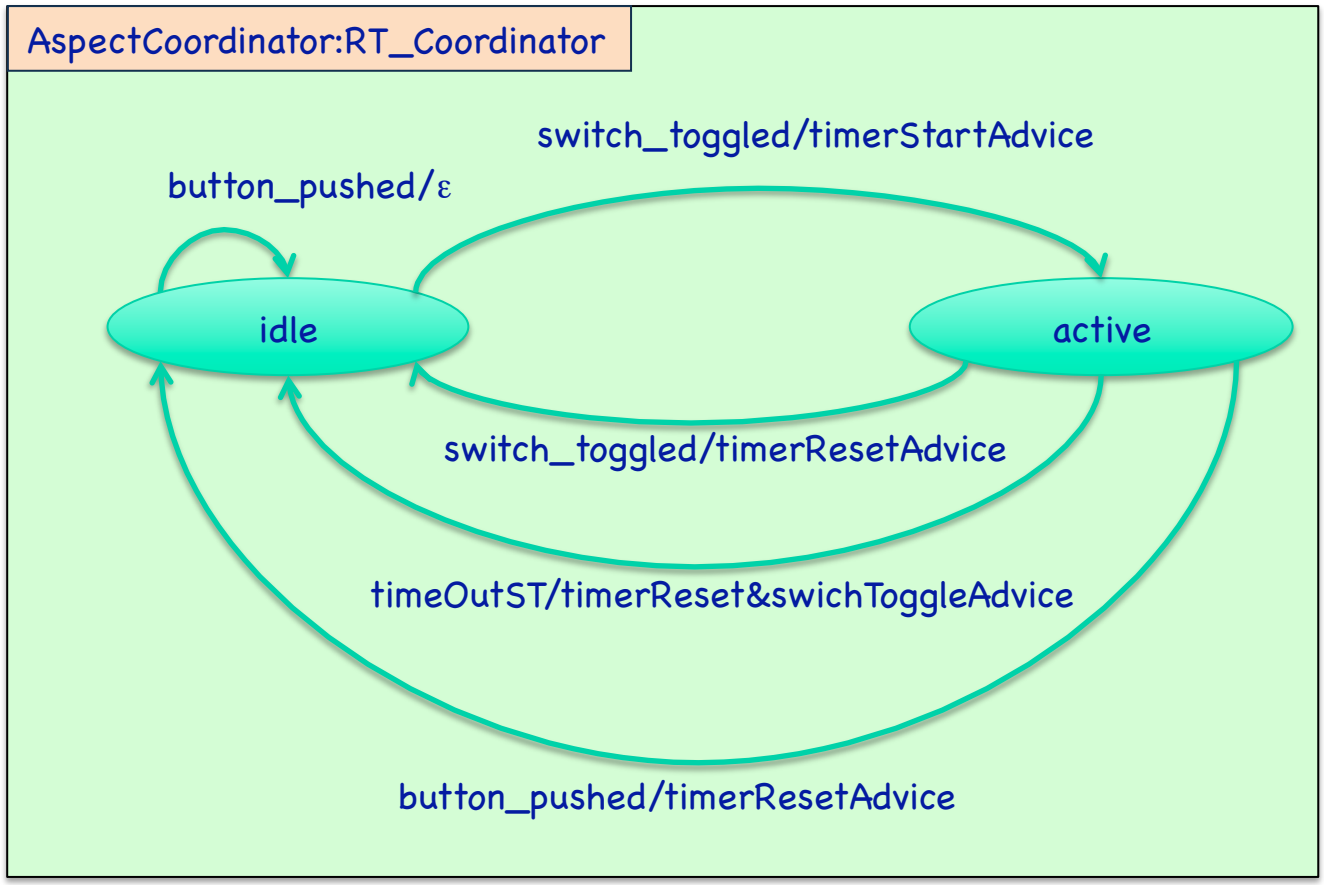
CSTM

PrimitiveCSTMの仕様とAspectCoordinatorの仕様はステートマシン図によって記述される

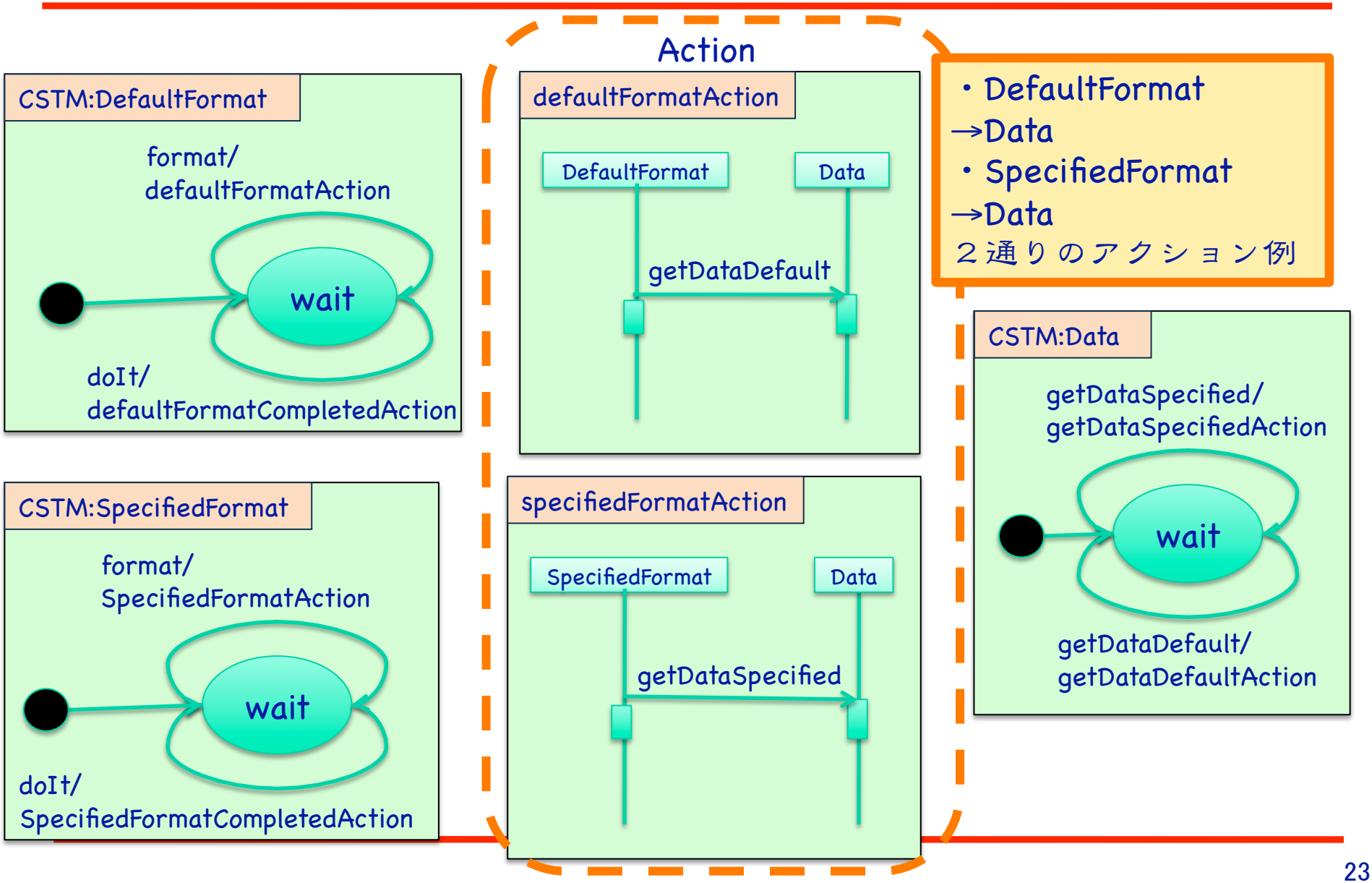
PrimitiveCSTMの例（Switch）



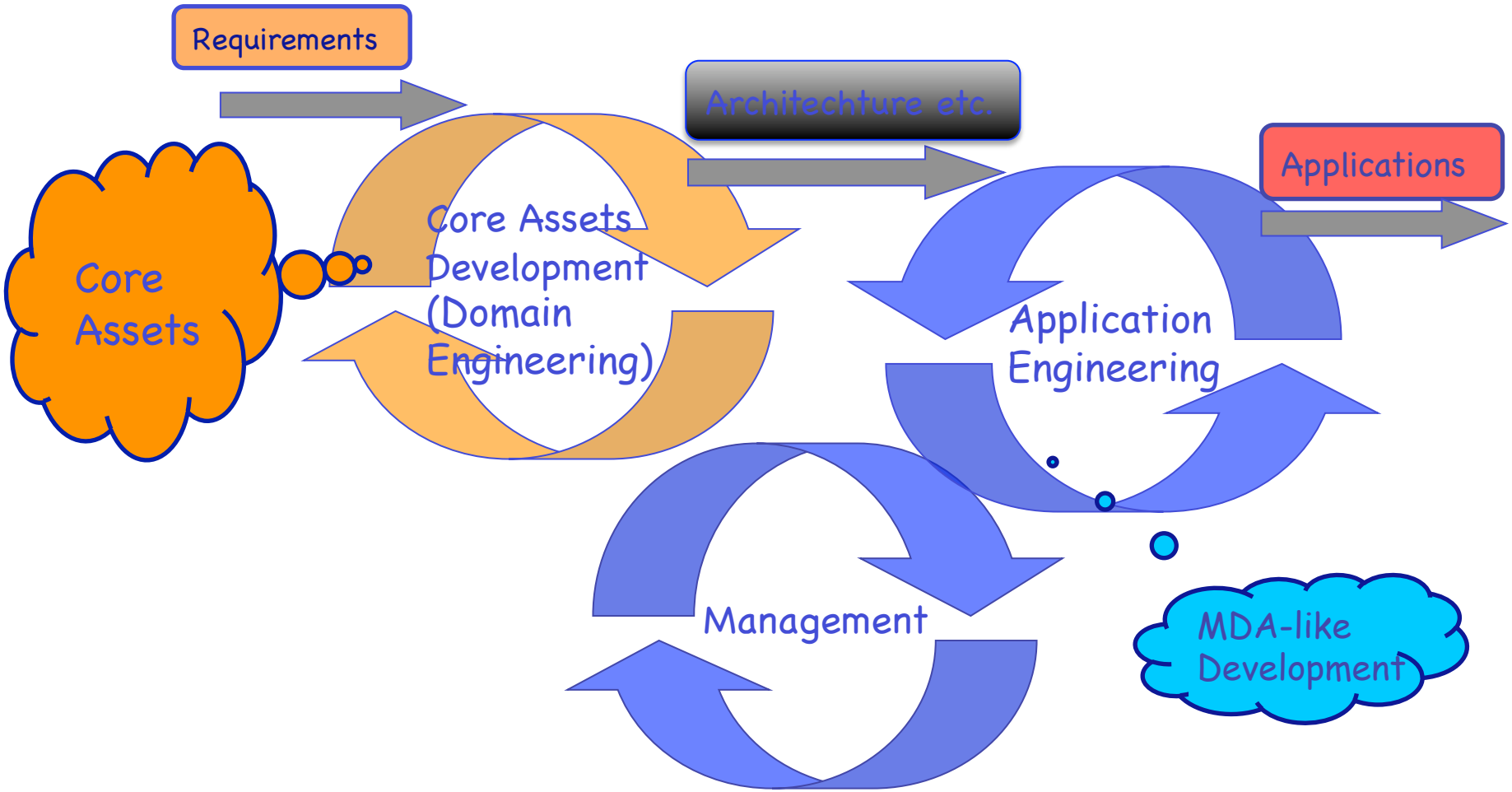
AspectCoordinatorの例 （RT_Coordinator）



Action


Action
CSTM:DefaultFormat
format/
defaultFormatAction
wait
doIt/
defaultFormatCompletedAction
defaultFormatAction
DefaultFormat
Data
getDataDefault
・DefaultFormat
→Data
・SpecifiedFormat
→Data
２通りのアクション例
CSTM:Data
getDataSpecified/
getDataSpecifiedAction
wait
getDataDefault/
getDataDefaultAction
CSTM:SpecifiedFormat
format/
SpecifiedFormatAction
wait
doIt/
SpecifiedFormatCompletedAction
specifiedFormatAction
SpecifiedFormat
Data
getDataSpecified

PLSE プロセス


Requirements
Architechture etc.
Applications
Core Assets
Core Assets Development (Domain Engineering)
Application Engineering
Management
MDA-like Development

PLSEプロセス

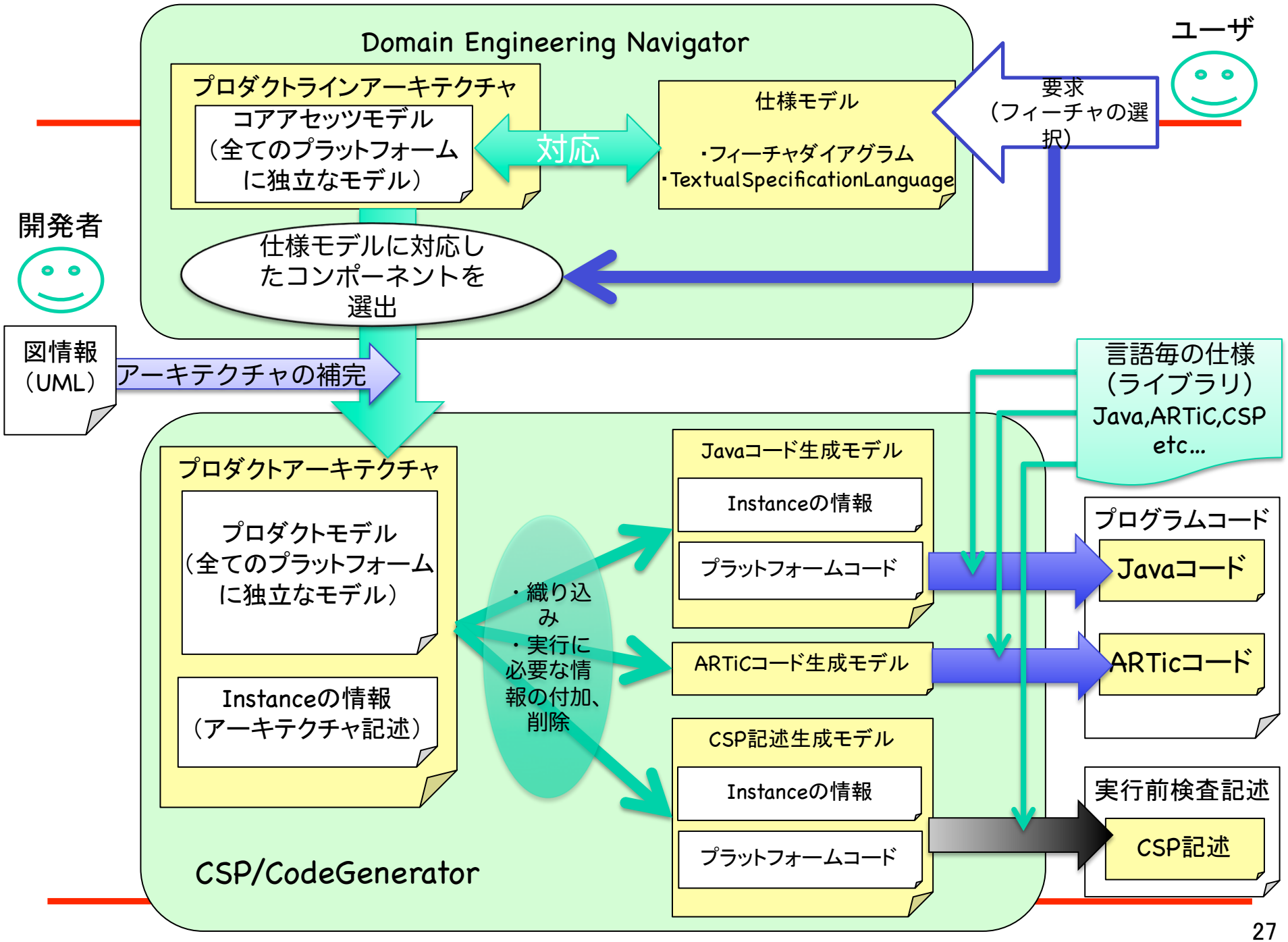
- Domain engineering side
  - Select Conceptual Architecture
  - Get Product Line Architecture
  - Build Product Architecture Frame
  - Fill PA Frame & Make Product Architecture
  - Pre-Execution Check
  - Code Generation
- Application engineering side
  - Fill Application Logic
  - Coding, Reuse & Test
  - Completion Application

Core Assets

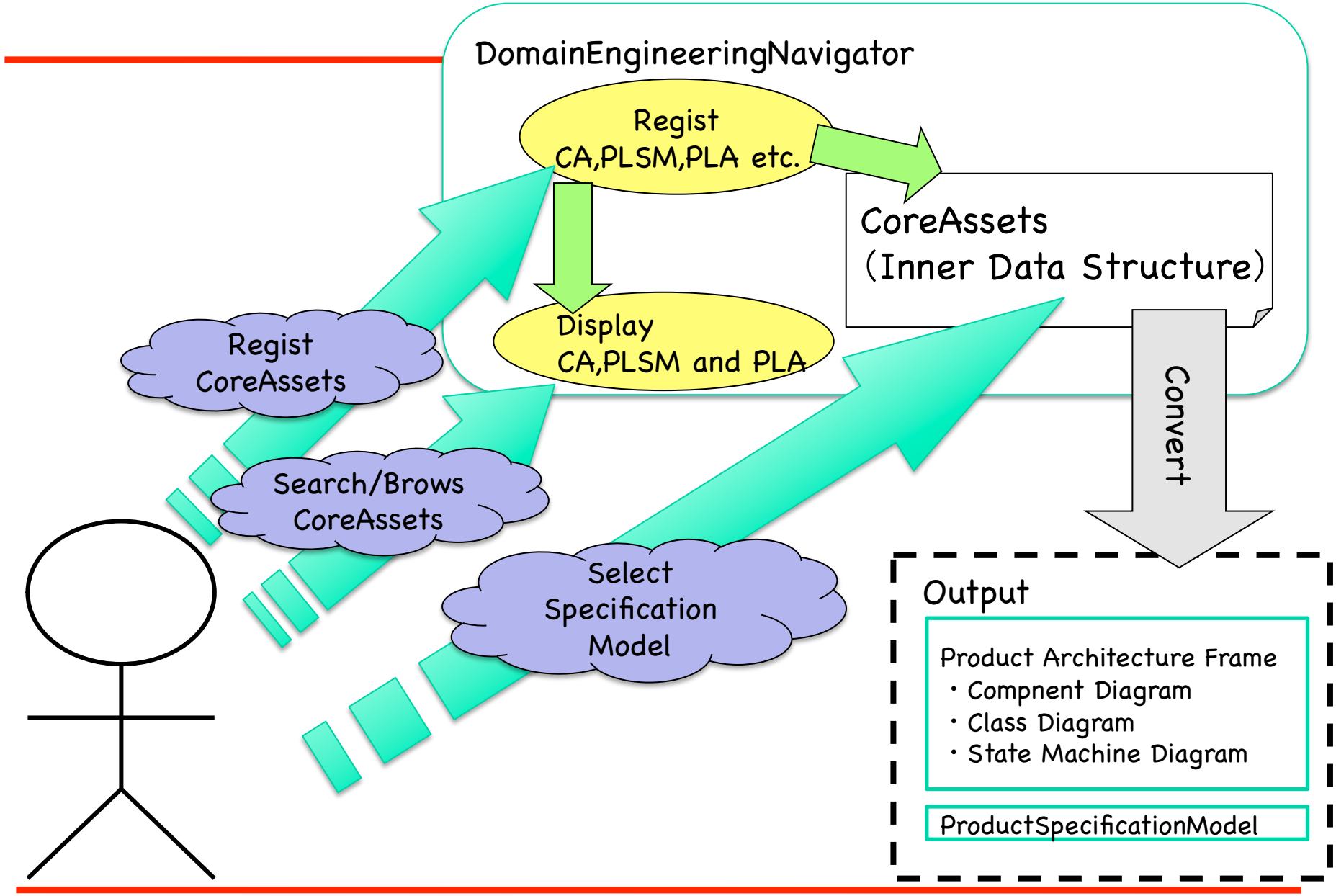
- State Transition Machine
- Inter Aspect Description
- Specification Model
- Conceptual Architecture
- Product Line Architecture
- Product Architecture
- Platform Code

PLSE支援ツール一覧

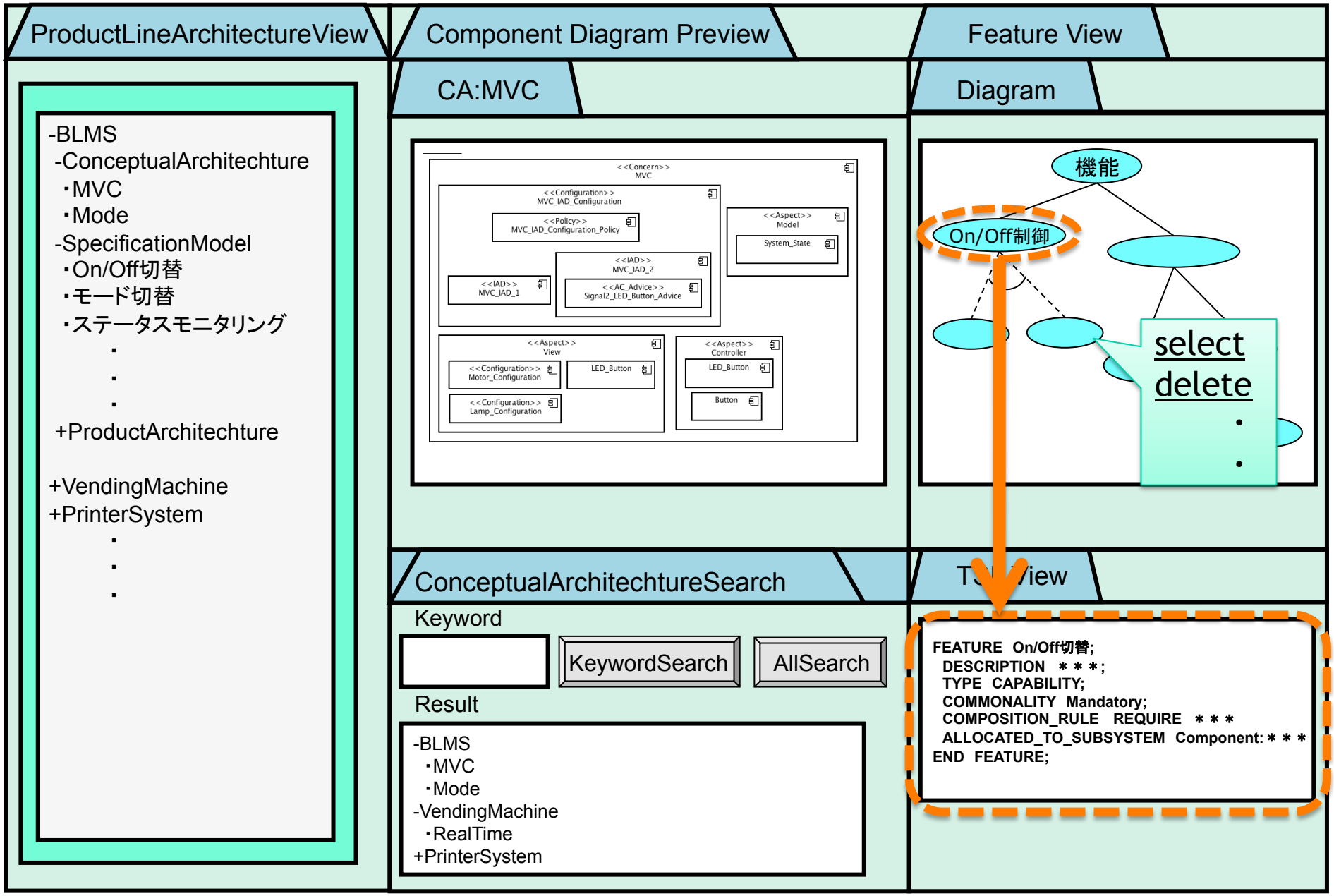
- Domain Engineering Navigator
- プラットフォーム
- 実行前検査ツール
- 実行前検査記述/コードジェネレータ


Domain Engineering Navigator
プロダクトラインアーキテクチャ
コアアセッツモデル
（全てのプラットフォームに独立なモデル）
対応
仕様モデル
・フィーチャダイアグラム
・TextualSpecificationLanguage
ユーザ
要求
（フィーチャの選択）
開発者
仕様モデルに対応したコンポーネントを選出
図情報
（UML）
アーキテクチャの補完
言語毎の仕様
（ライブラリ）
Java,ARTiC,CSP
etc…
プロダクトアーキテクチャ
プロダクトモデル
（全てのプラットフォームに独立なモデル）
Instanceの情報
（アーキテクチャ記述）
・織り込み
・実行に必要な情報の付加、削除
Javaコード生成モデル
Instanceの情報
プラットフォームコード
ARTiCコード生成モデル
CSP記述生成モデル
Instanceの情報
プラットフォームコード
プログラムコード
Javaコード
ARTicコード
実行前検査記述
CSP記述
CSP/CodeGenerator

Domain Engineering Navigator



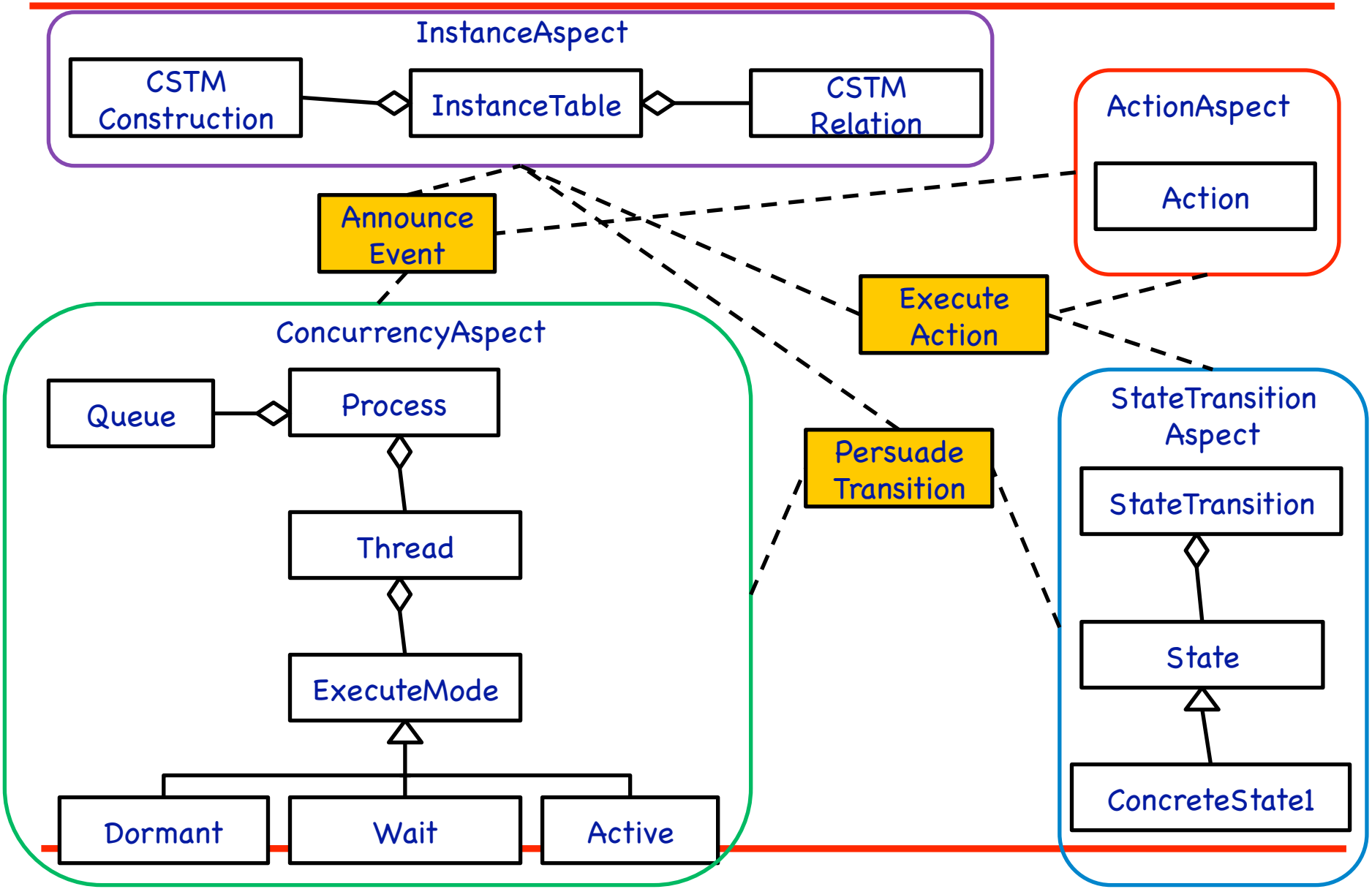
Domain Engineering Navigator -UI Image-



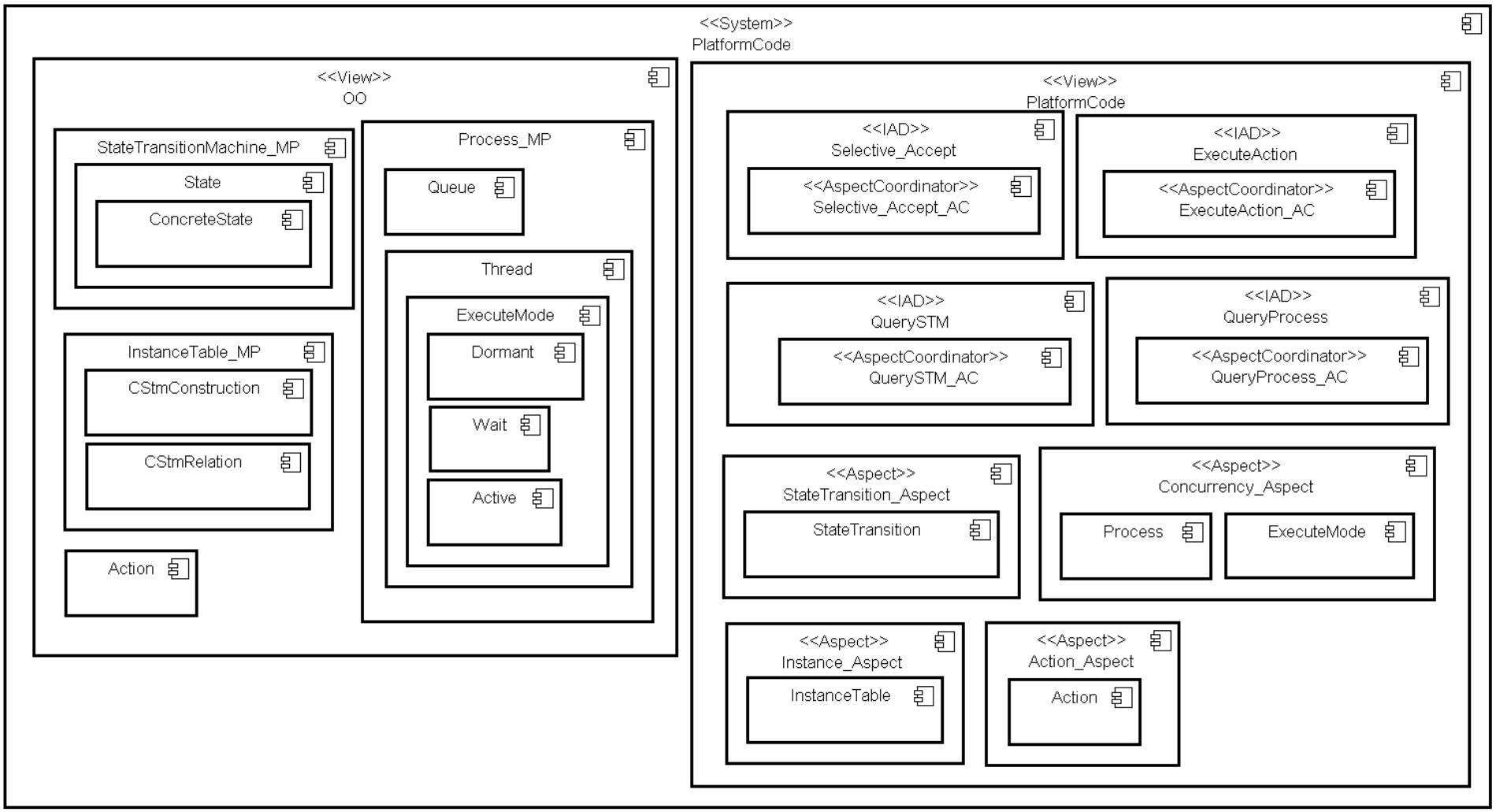
プラットフォーム仕様

- 並行に動作するCSTMの集合
- イベントキューを介した非同期通信
- コンポーネントの実行モード
- アプリケーションロジック
- インスタンス処理

プラットフォームのアスペクト指向設計



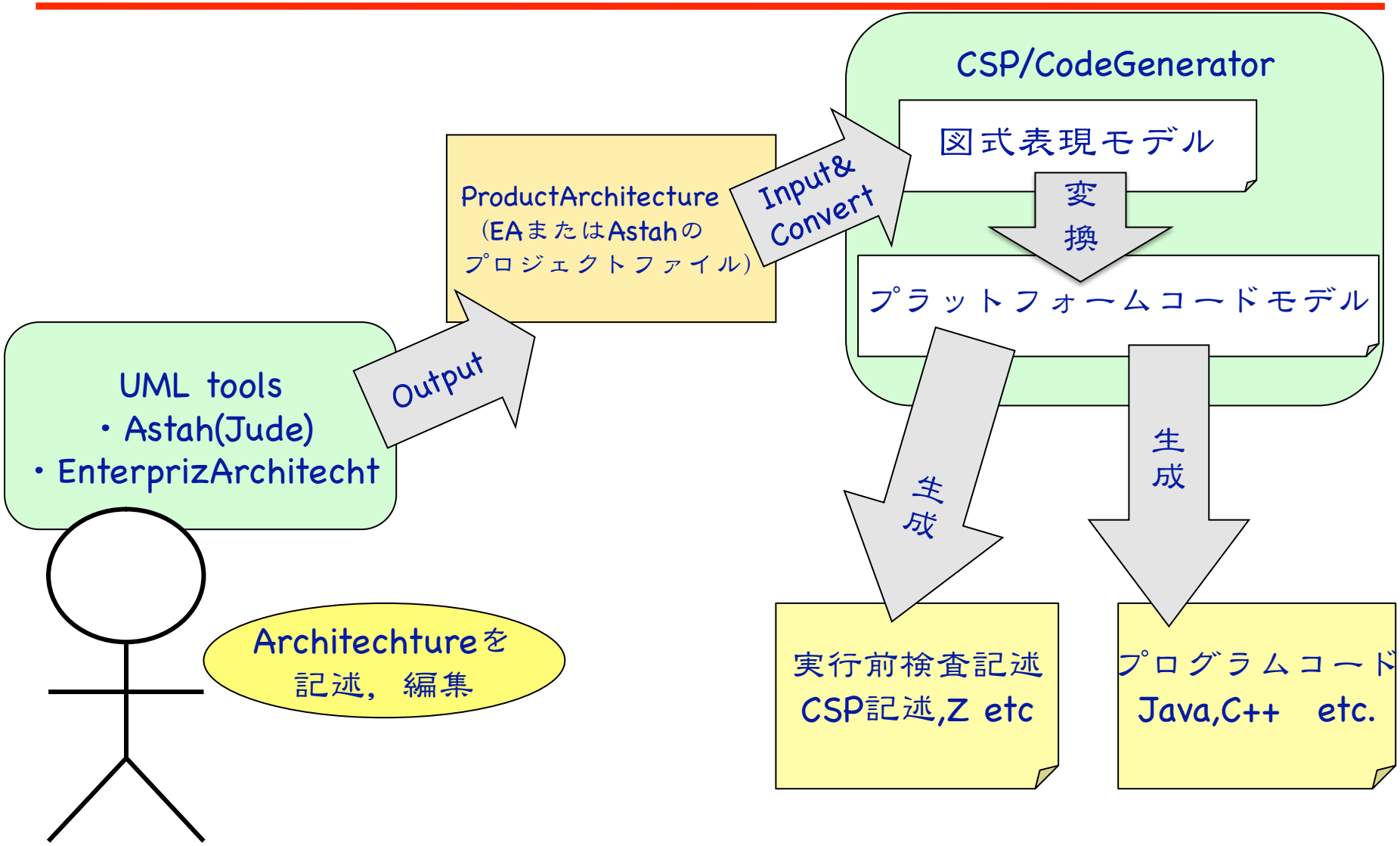
プラットフォームのアスペクト指向設計



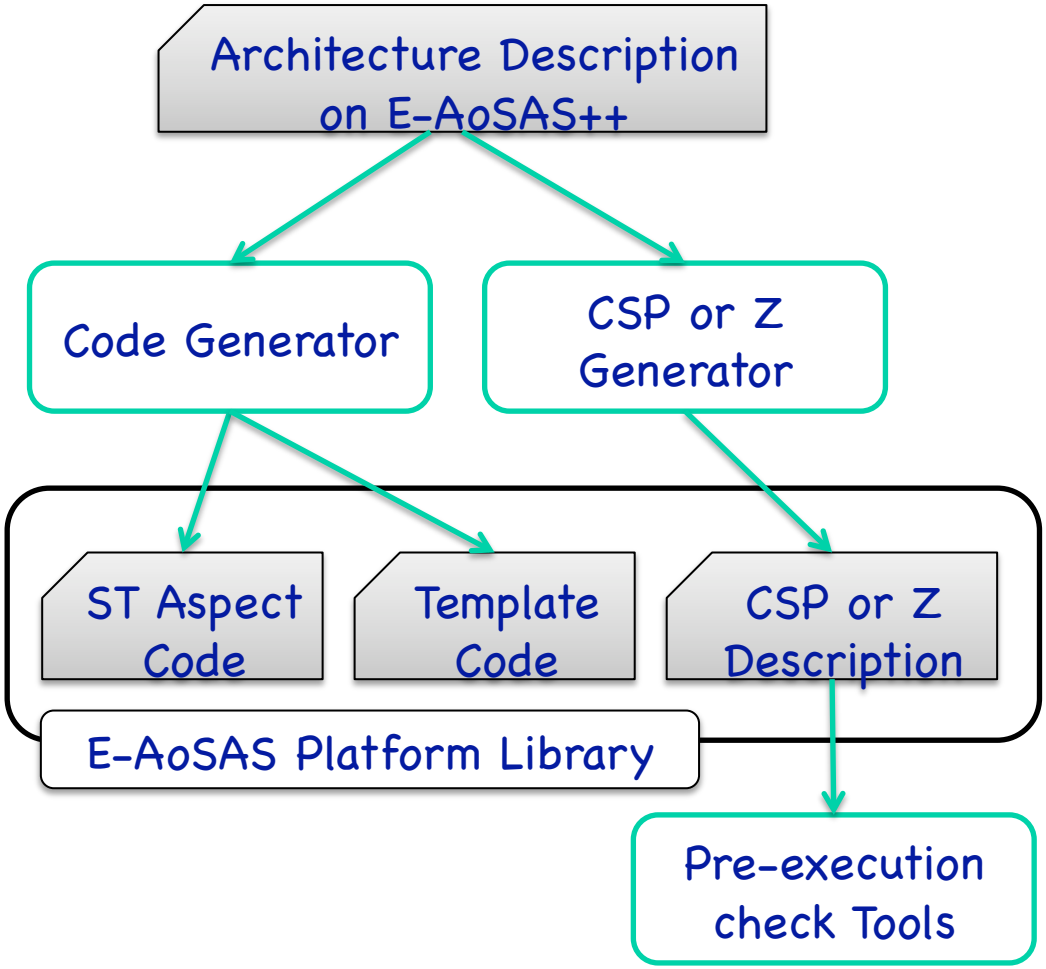
実行前検査ツール

- FDR: CSP理論に基づく並行システム検証ツール
  - 実装も仕様もCSP記述によってモデル化される
- SPIN: Promela記述による検証ツール
  - 実装にオートマトンを用い，仕様は時相論理式によってモデル化する

実行前検査記述/コードジェネレータ （概要）



実行前検査記述/コードジェネレータ（データフロー）



PLEASEの実用化に向けて

- UML:ソフトウェア開発言語
  - UMLを拡張したアーキテクチャ記述を提案
  - 記述エディタとしてAstah,Enterprise Architectを利用
- Eclipse:統合開発環境
  - Java(AspectJ),C/C++などのプラグインを利用
  - CodeGenerator等のツールの開発に利用
- Java and other platform
  - CodeGeneratorのプラットフォームとしてJavaやC/C++, CSPやPromelaを想定

使用した既存のツール

- ソフトウェアツール
  - 既存のUMLエディタ
    - Astah
    - EnterpriseArchitect
  - CodeGenerator
    - Java,C/C++ Generator
    - CSP,Promela Generator
  - 既存の実行前検査ツール
    - FDR
    - SPIN

まとめ

- 組込みソフトウェアを状態遷移機械の集合としてモデル化
- アスペクト指向ソフトウェアアーキテクチャスタイルE-AOSASA++
- PLSE環境PLEASE
- 実際の開発に適用し開発期間の短縮を確認