2004/10/18

TECHNOLOGIC ARTS INCORPORATED

「実践！組込みUML～UMLの現状とこれから～」

組込みエンジニアのための UMLモデリング入門

久保秋　真（kuboaki@tech-arts.co.jp）

株式会社テクノロジック アート

113-0033 東京都文京区本郷4-1-4コスモス本郷ビル9F

Tel:03-5803-2788 Fax:03-5803-2989

http://www.tech-arts.co.jp/

Agenda

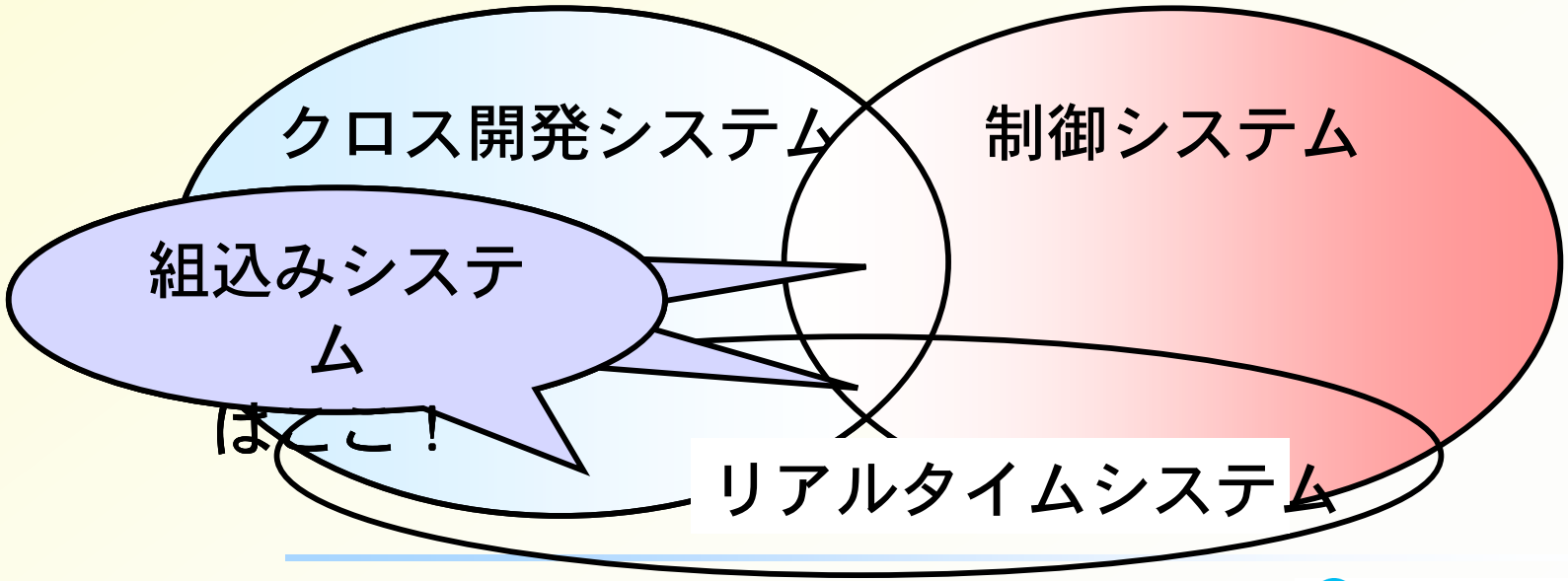
- 組込みシステムとは
- くみこまーの3K
- くみこまー周辺のオブジェクト指向対策
- チャレンジの反省
- 組込みシステムでUMLを使ってみよう
- くみこまーはもっと外に出よう

組込みシステムとは（一般論）

- コンピュータを組込んでいる特定目的のシステム
  - 家庭用電化製品
  - 工業用ロボット・工作機械
- PCよりも使われている数と種類がずっと多い
  - 携帯電話・自動車・カーナビ・炊飯器・信号機・エレベーター・自動販売機・デジタルカメラ・テレビ…
- PCのシステム開発にはない制約
  - 制御したいハードウェアの仕様や性能がソフトの組み方を制約する
  - 決められた時間内に処理を終わらせる（リアルタイム：実時間）

組込みシステムであるなら（持論）

- クロス環境で開発すること
  - PC上の開発環境（クロス）を使ってビルド
  - 対象（ターゲット）機器にダウンロードして実行
- 制御システムであること
  - 外界の副作用と刺激による制御（センサー・アクチュエータ）
- リアルタイム制御があること



組込みシステムの状況

- 大規模化・複雑化
- 慢性的な要員不足
- テストで仕様確認？
  - 障害原因が仕様不備とか

- 組込みエンジニアの業務知識＋ソフト知識
  - 製品知識の重要度が大
  - クロス＋C言語がくみこまーのソフト世界観
    - ソフトウェアの広範な知識は重要視されていない

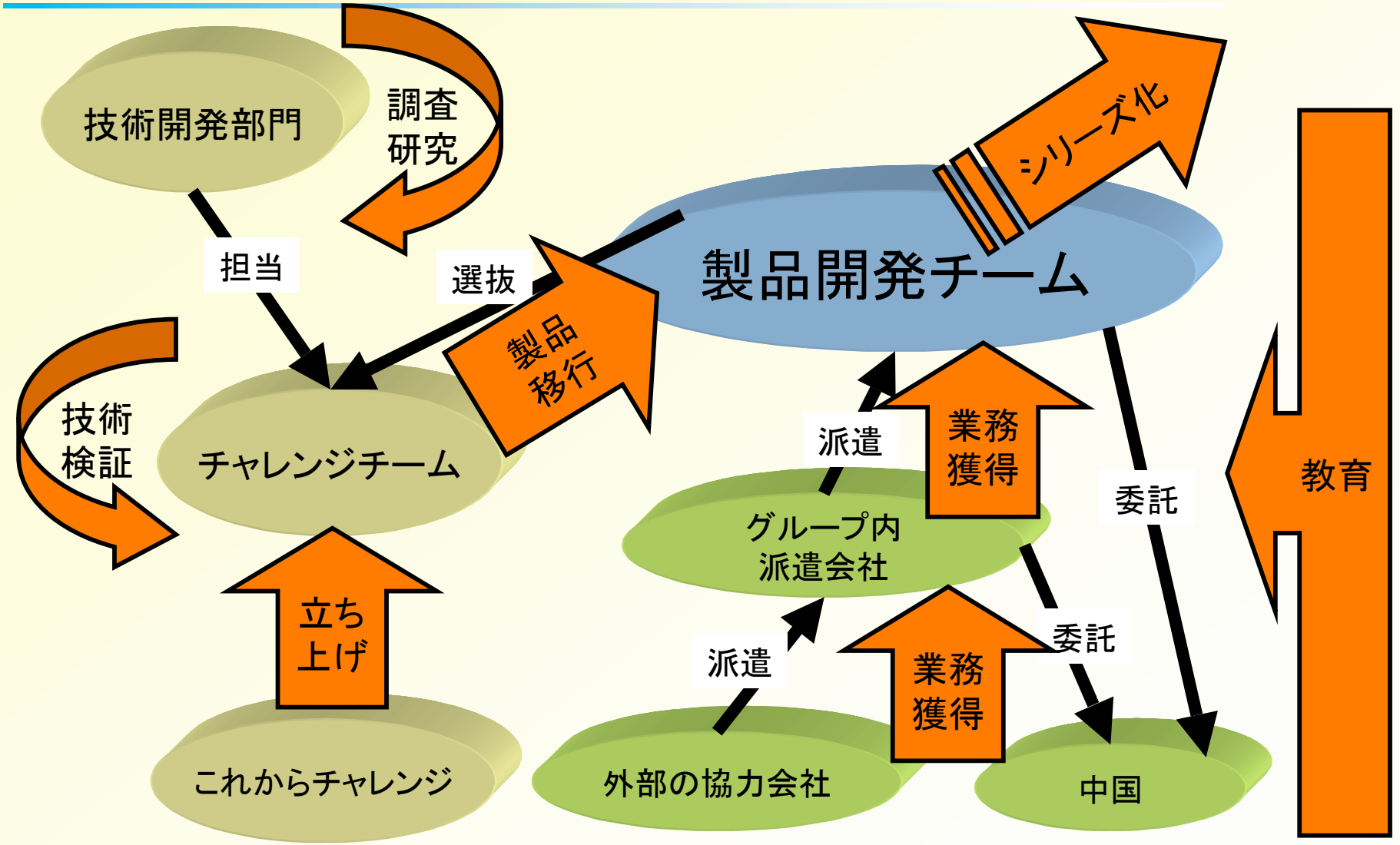
- 組込み技術者のスキル標準化
- 組み込み開発のプロセス標準化
- オブジェクト指向はチャレンジ段階で立ち往生

くみこまーの3K

- 厳しい
  - 納期
  - 対象の動作環境
  - 就労環境や機密保持
- 細かい
  - 精密な制御
  - 小さな対象機器
  - ささいな欠点も許容しない
- 帰れない
  - 残業が日常化
  - 障害が多い
  - みんなの書いた全てのコードが動くまで帰れない…

- 3K職場（さんけいしょくば）
  - きつい・汚い・危険な職場
  - ゴミ処理・産廃処理の現場
  - 就労者に嫌われている

くみこま一周辺のオブジェクト指向対策



くみこまー、オブジェクト指向に飛びつく

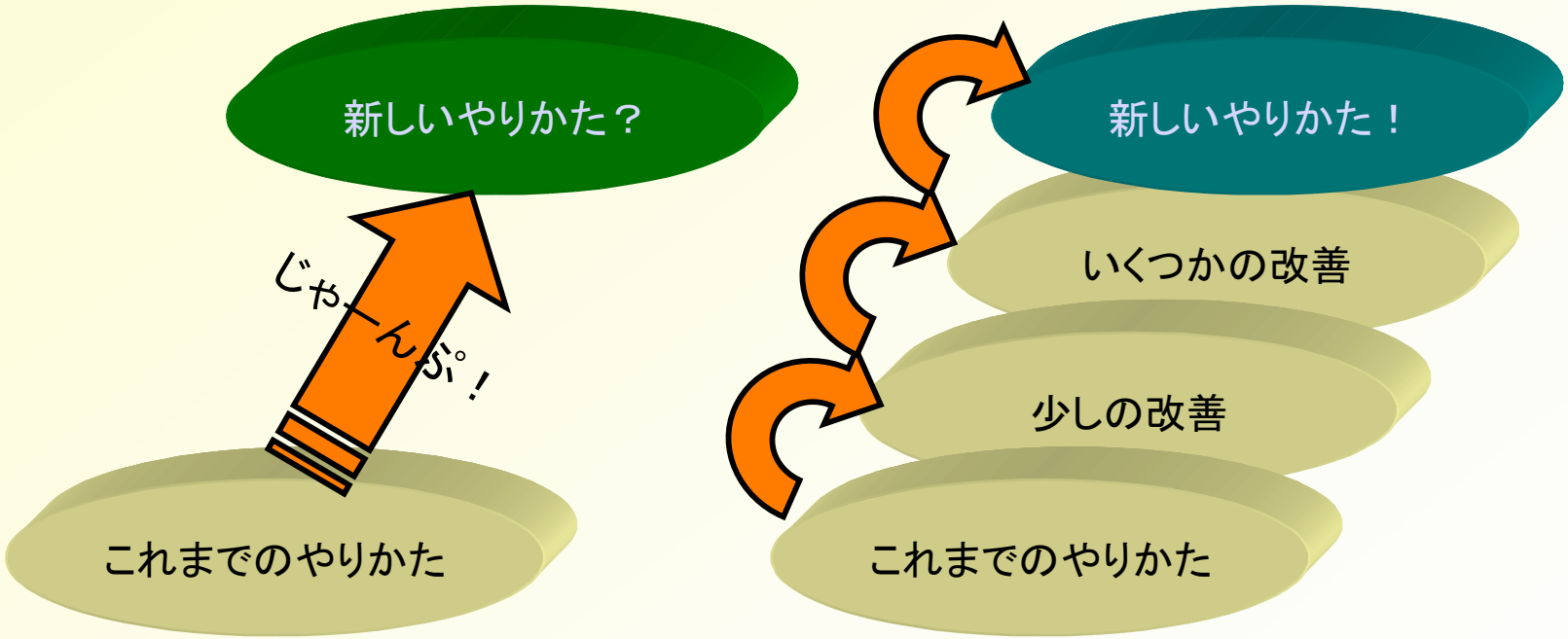
- 組込みシステムは大変な状況だ！
  - それならオブジェクト指向だ！
  - モデリングだ！
- チャレンジしてみたが、立ち往生…
  - 製品適用にはとてもいたらず
- 目的を見失ってしまった？
  - 問題は何か
    - 本来解決したかった問題はなんだったのか（何が大変？）
  - それは誰の問題か
    - 解決すべき問題は誰の問題なのか（誰が大変？）

反省1：それは大きなジャンプだった…

- くみこま一存亡の危機
  - 大規模化・複雑化・短納期化
  - 要員不足なのに単価低迷
  - 安い上に「くみこま一の3K」では救われない
- 「オブジェクト指向を活用して打開しよう！」
  - UMLを学びましょう
  - 開発プロセスを学びましょう
  - 改めて要求を定義しなおしましょう
  - 「分析は…、設計は…、実装は…、
    いままでとは進め方が大きく変わりますよー。」
- 「そんなんじゃ、できねーよ！」ってことに…

大きなジャンプは難しい

- くみこまーは
  - ソリューションを買うのではなく
  - 自ら創り出す必要がある

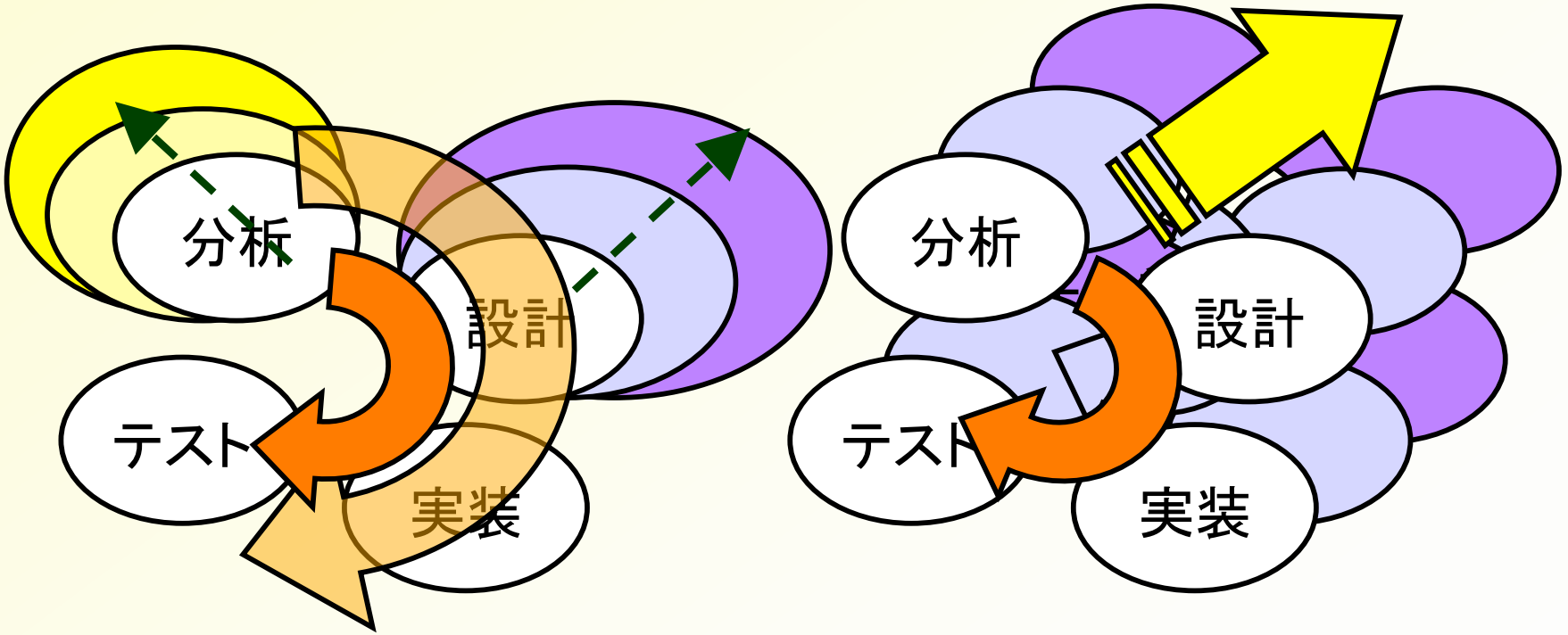


反省2:「動かす」ことから遠ざかっていた…

- 組込み開発では動かして確かめることが大事
  - 大規模化・多機能化と確かめることのバランスをどうとるか？
- 組込みシステムでモデリングがうまく使えない….
  - すぐに動かして確かめられない，という苛立ちや不安.
- 静的なモデルだけでは動かない.
  - 結局，動かすには実装までやる羽目になる.
  - 動作確認までに長い作業ループが必要になる. ←これが辛いわけだ.
- 短期的な成果を求められる組込みエンジニアは苦しむ.
  - これでは悪構造問題（唯一の解が定まらない仕事）には向かない.
  - だから「くみこまー」はすぐにコードを書きたがる.
    - 確かめたいのだ. これは正常な反応だ.

さっさと動かすこと vs. じっくり設計すること

- 動かして確認する作業ループが長いとつらい
- 大規模化・多機能化対策でよい設計が必要
- いまの作業ループが長く延びては困る



チャレンジを見直してみる

- オブジェクト指向は特効薬・万能薬ではない
  - 問題は何か、それはどの工程の問題か
  - 間違った処方（手法やツール）は効き目がない
- 要員不足を正しく見極めよう
  - 本当に力不足なのはどの工程・担当部分か？
    - バリューストリーム・マップを作成してみる
  - むやみな増員は生産性・品質の悪化をまねく
  - 囲い込みによる技術者の成長阻害は起きていないか？
- 大規模化・複雑化に力で対抗しない
  - 1行、1ページの表現力を上げる
    - これまでと同じ粒度のままでは効率は悪化する
  - 管理強化の前にムリ・ムダを排除しよう
    - 管理手法のブルトーザーで押し切らない

むやみなジャンプはやめよう

- 対策を少し見直そう
  - プロセスの入れ替えは大きなジャンプだった
  - チームが手をつないでジャンプしたら川に落ちる
  - 誰もができることをみんなでやった方がいい
  - みんなができたら、その次へ進めばいい
  - そのうちにスキル標準・プロセス標準もできてくる

- UMLを活用して底上げを図ってはどうか

- 「できそうなオブジェクト指向」から始めよう
  - ジャンプしないで、できそうなことから始めてみよう
  - あせらなくて大丈夫、UML採用はまだ15%以下
  - UMLモデリングして、コードを導いてみよう
  - うまくできたら、コミュニケーションのツールにしよう

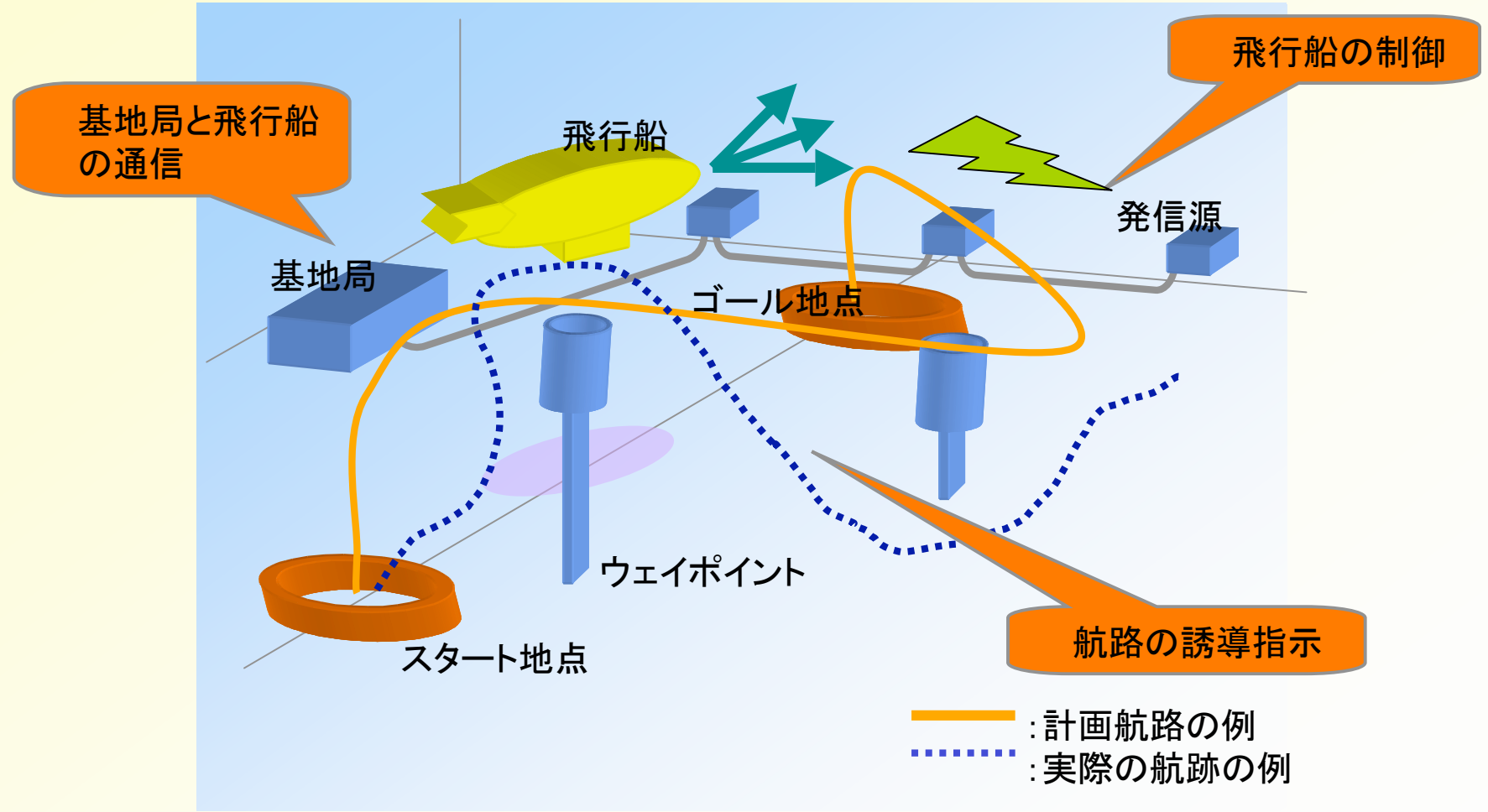
組込みシステムでUMLを使ってみよう

- MDDロボットチャレンジからの紹介
  - 情報処理学会：組込みソフトウェアシンポジウム 2004より

- システムの概要
  - 出発点から離陸し、ウェイポイント2点を通過して目的地に着地させる
  - 基地局と飛行船は赤外線と超音波で通信する
  - 飛行船には左右2つ上下用1つのファンがついている
  - 基地局は飛行船の位置を算出し、飛行制御信号を送ってファンを操作し、飛行船を目的地へ誘導する

対象システムの概観

- 飛行船を基地局からリモート操作してゴールを目指す



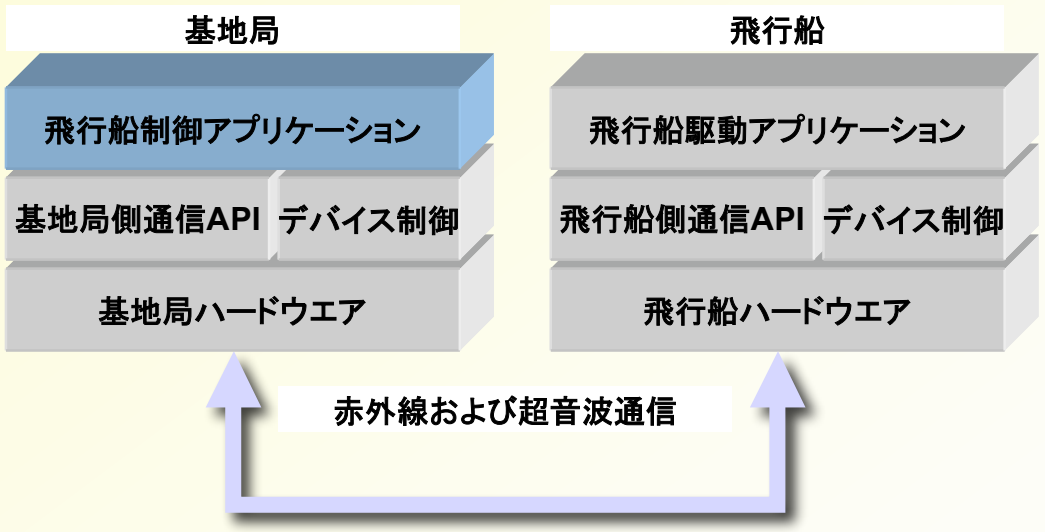
開発作業の流れ

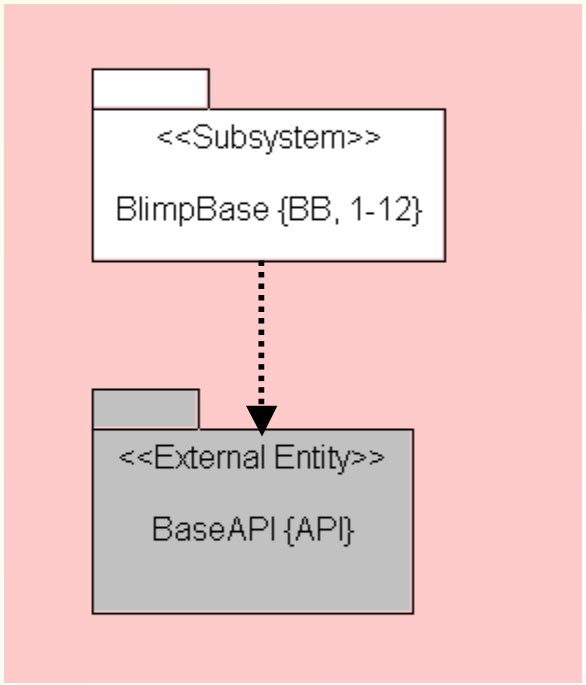
- 順番に進めるには…
  - 開発対象範囲を決定
  - ユーケース記述とロバストネス図の作成
  - クラス図と状態遷移図を作成
  - 状態図に合わせて振る舞いをプログラムに書く
  - クロス環境でモジュール作成
  - 実機にダウンロードしてデバッグ

- 確かめながらすすめるには…
  - まず動作する開発系を確立する
    - モデル、状態図の記述からモジュール作成までの処理を通す
    - 単純なことを試す（コマンドを送ってファンをまわすとか）
  - その後、順次機能をモデルに追加していく

対象システムの分割

- ここでは基地局アプリケーションを対象
  - 基地局サブシステムと基地局用APIで構成する





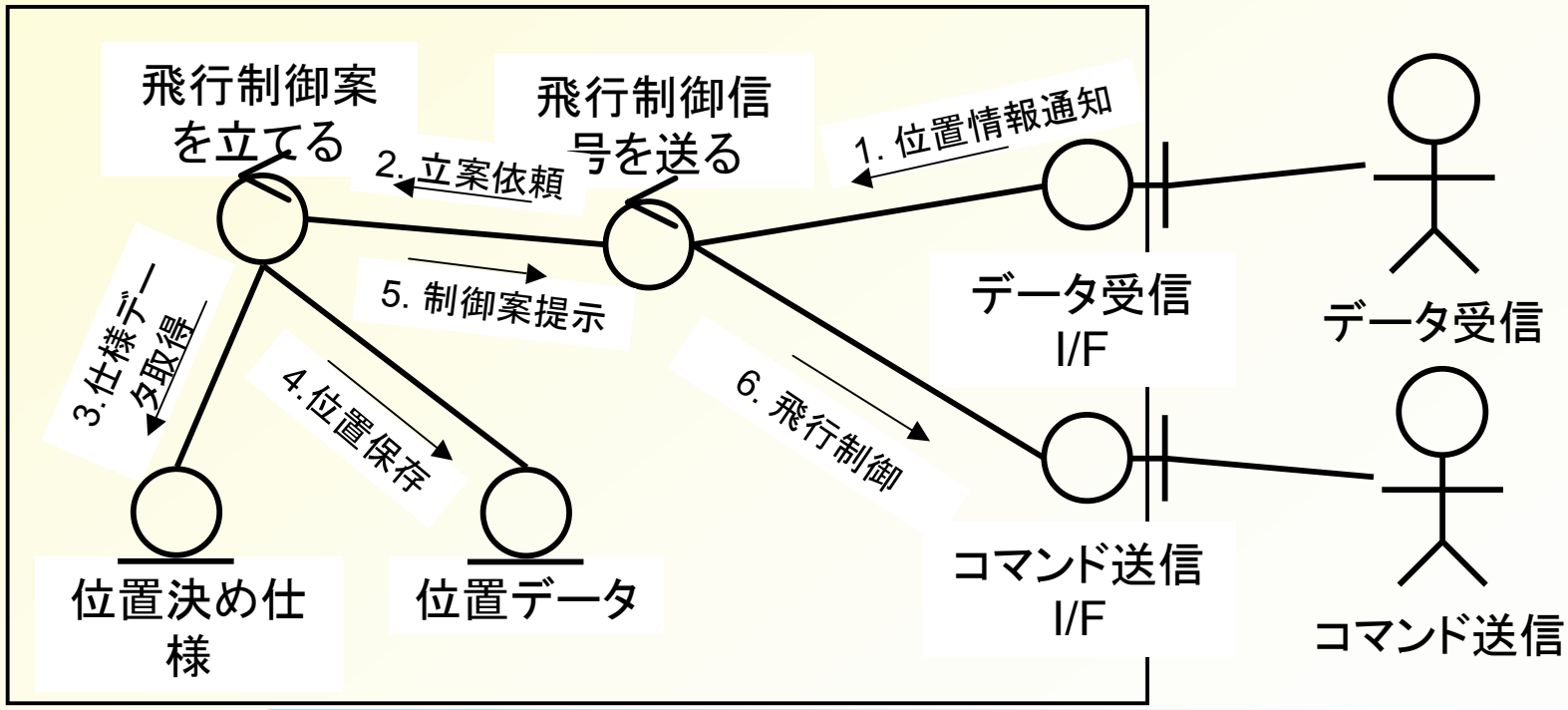
ユースケース記述を作成

- ユースケース：飛行船を制御する
  - 通信プロトコルの仕様から，飛行船の制御に関しては次の3つの制御が必要であることがわかった．
    - 飛行船からの時刻合わせ通知を契機に位置情報検出用信号を送信する．
    - 飛行船からの位置情報通知用信号を契機に飛行船制御信号を送信する．
    - 飛行船からのメッセージ通知用信号を受け取る．

- 「位置検出信号を送信する」の記述
  - 飛行船が時刻合わせ信号を送信すると，システムはシリアルポートを通じてこれを受け取る．
  - システムは，発信源のIDを伴った位置情報検出用信号を作成し，100ms 間隔で3つの発信源から飛行船に送信する．

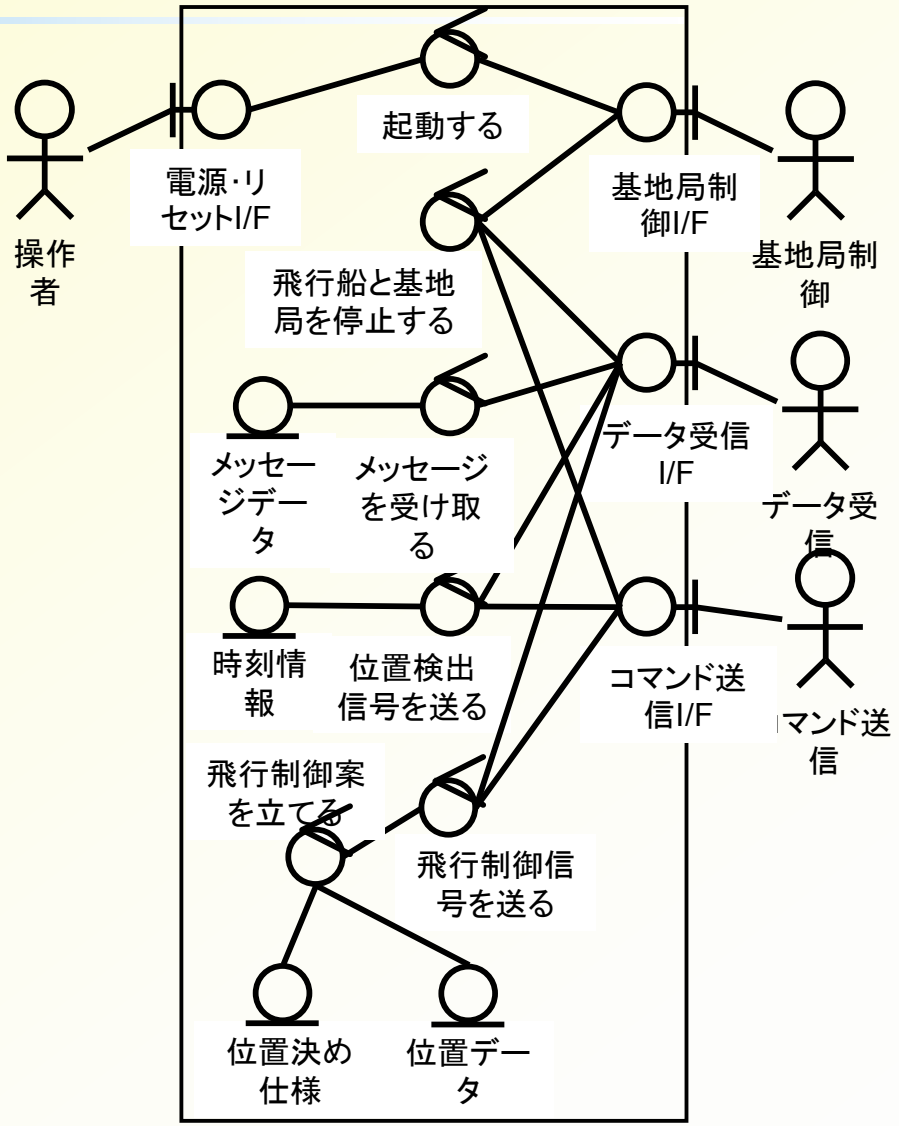
ロバストネス図を作成

- ユースケース記述を書くとき、登場人物を整理する
- バウンダリ・エンティティ・コントロールに切り出す
  - システムの境界、データを保持するところ、シナリオを制御するところに分類する

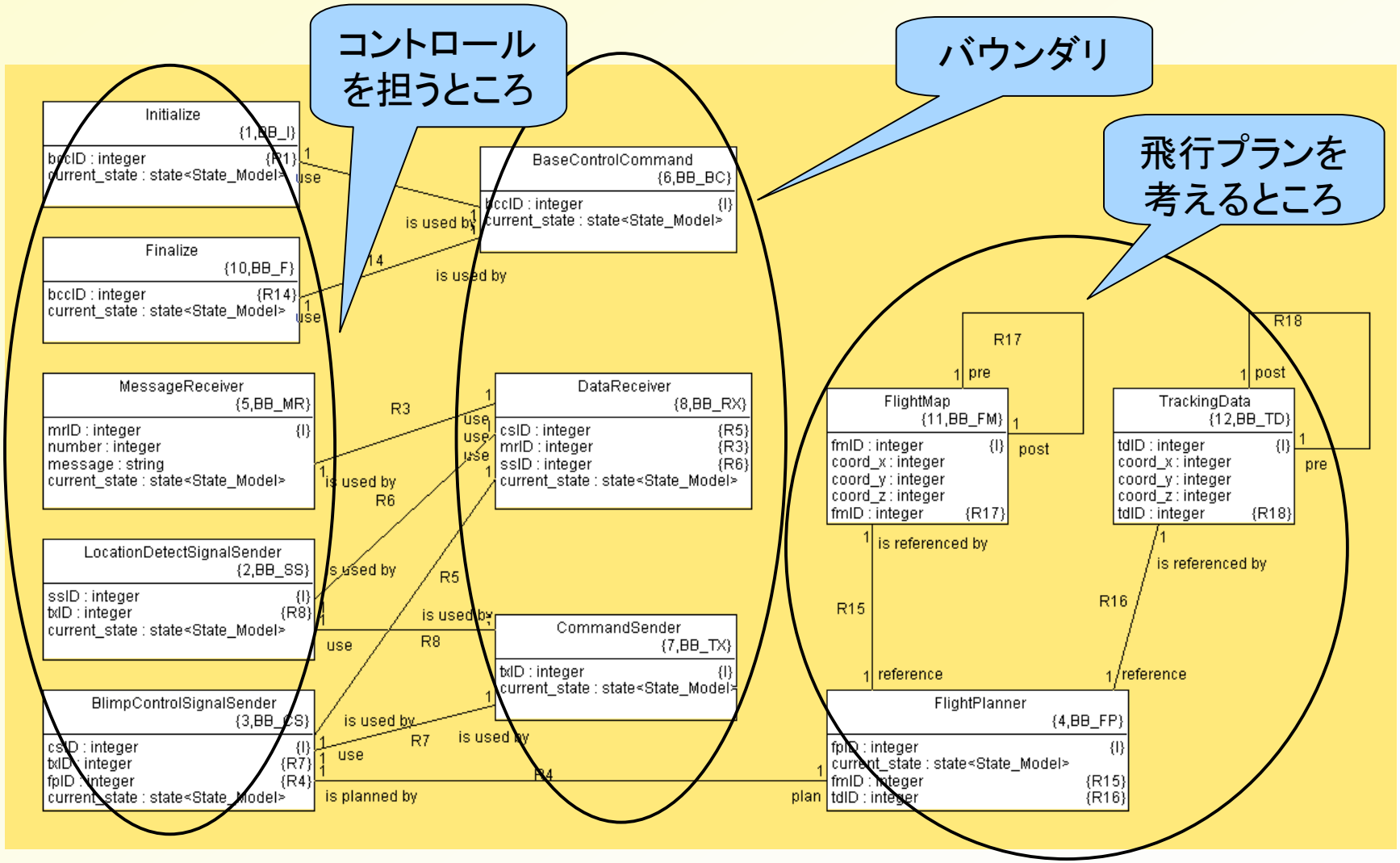


ロバストネス図をマージ

- シナリオごとのロバストネス図をマージしてシステム全体のロバストネス図を作成する
- この図に対応するクラス図を描けばよい

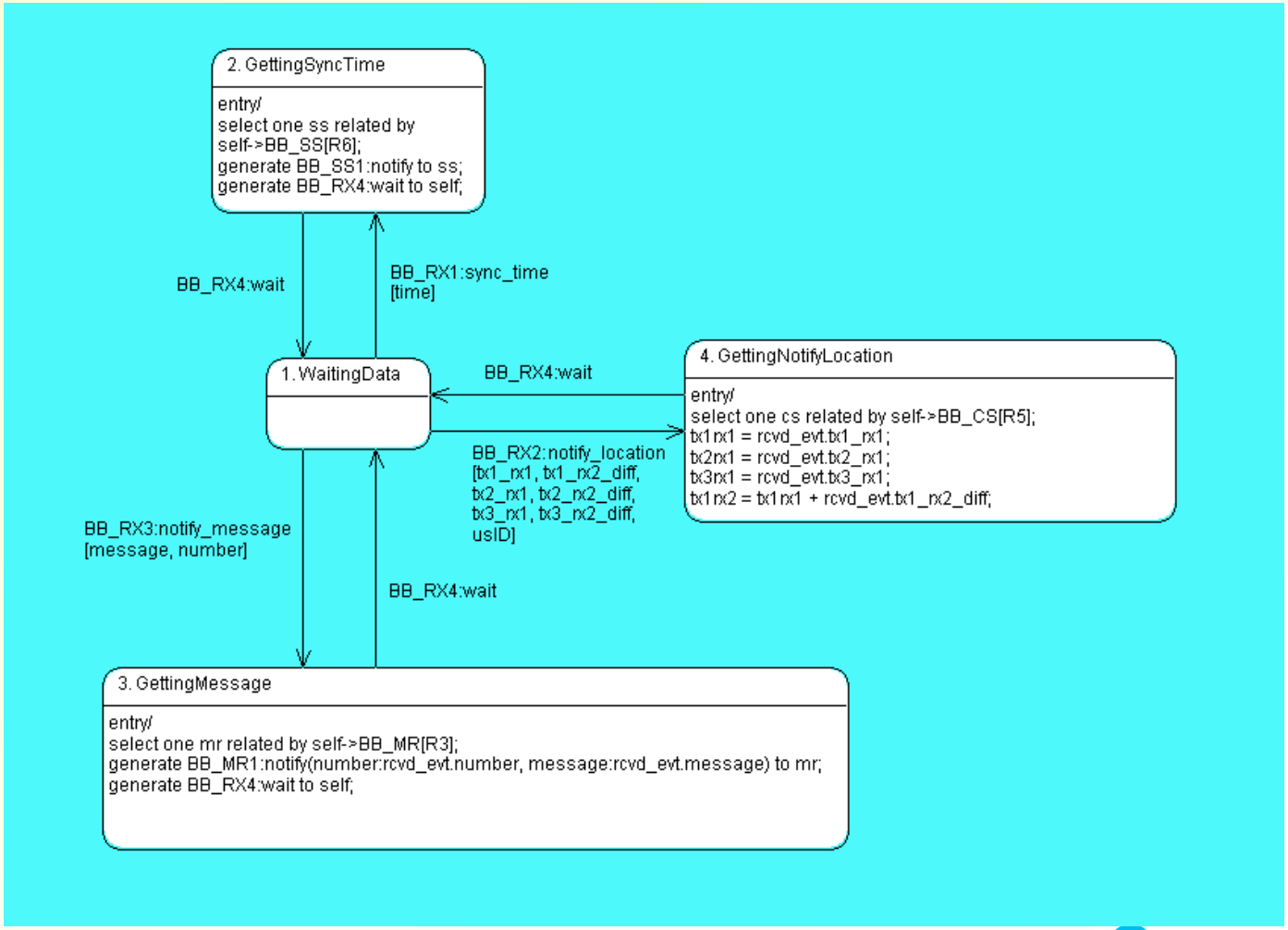


クラス図を作成


コントロール
を担うところ
バウンダリ
飛行プランを
考えるところ
Initialize
{1,BB_I}
bccID : integer {R1}
current_state : state<State_Model>
Finalize
{10,BB_F}
bccID : integer {R14}
current_state : state<State_Model>
MessageReceiver
{5,BB_MR}
mrID : integer {I}
number : integer
message : string
current_state : state<State_Model>
LocationDetectSignalSender
{2,BB_SS}
ssID : integer {I}
txID : integer {R8}
current_state : state<State_Model>
BlimpControlSignalSender
{3,BB_CS}
csID : integer {I}
txID : integer {R7}
fpID : integer {R4}
current_state : state<State_Model>
BaseControlCommand
{6,BB_BC}
bccID : integer {I}
current_state : state<State_Model>
DataReceiver
{8,BB_RX}
csID : integer {R5}
mrID : integer {R3}
ssID : integer {R6}
current_state : state<State_Model>
CommandSender
{7,BB_TX}
txID : integer {I}
current_state : state<State_Model>
FlightMap
{11,BB_FM}
fmID : integer {I}
coord_x : integer
coord_y : integer
coord_z : integer
fmID : integer {R17}
TrackingData
{12,BB_TD}
tdID : integer {I}
coord_x : integer
coord_y : integer
coord_z : integer
tdID : integer {R18}
FlightPlanner
{4,BB_FP}
fpID : integer {I}
current_state : state<State_Model>
fmID : integer {R15}
tdID : integer {R16}
R17
pre
post
R18
post
pre
R15
R16
is referenced by
reference
R3
R6
R5
R8
R7
R4
use
is used by
is planned by
plan

状態遷移図を作成

- DataReceiver クラスの状態遷移図



コードを作成

```
/*****************************************************************************
 * State [3]: 'GettingMessage'
 *****************************************************************************/
void
MS_BB_RX_Action_3( MS_BB_RX_s * self, const OoaEvent_t * const event )
{
  MS_BB_RX_Event3_s * rcvd_evt = (MS_BB_RX_Event3_s *)event;
  MS_BB_MR_s * v28;  /* mr */

  /* SELECT ONE mr RELATED BY SELF->BB_MR[R3] */
  v28 = self->mc_BB_MR_R3;


  /* GENERATE BB_MR1:'notify'(number:RCVD_EVT.number, message:RCVD_EVT.message) TO mr */
  {
    MS_BB_MR_Event1_s * event29 = (MS_BB_MR_Event1_s *) Escher_NewOoaEvent( (void *) v28, &MS_BB_MR_Event1_sc )
    Escher_strcpy( event29->m_message, rcvd_evt->m_message );
    event29->m_number = rcvd_evt->m_number;
    Escher_SendEvent( (OoaEvent_t *)event29 );
  }


  /* GENERATE BB_RX4:'wait'() TO SELF */
  {
    MS_BB_RX_Event4_s * event30 = (MS_BB_RX_Event4_s *) Escher_NewOoaEvent( (void *) self, &MS_BB_RX_Event4_sc
    Escher_SendSelfEvent( (OoaEvent_t *)event30 );
  }


}

/*****************************************************************************
 * State [4]: 'GettingNotifyLocation'
 *****************************************************************************/
```

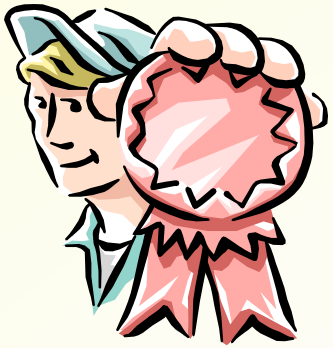
あとはビルドしてデバッグ！

- モデルに描いた振る舞いを実機上で確認する

- クラスや振る舞いがすべて洗い出せていなくても、モジュールを作成して実機で試すことができる
- あとは、徐々に機能を充実させていけばよい

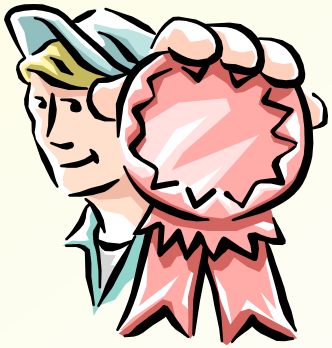
くみこまーはもっと外へ（1）

- もっとコミュニケーションを！
  - 企業・業種横断の「場」に参加する・作る
    - メーカーを越えたコミュニティへの参加
    - ツールのユーザー会
    - 情処ソフトウェア工学研究会
    - SESSAME、TOPPERS
  - UMLもコミュニケーションのツール



- くみこまーの誇り・やりがい・働きやすさを向上しよう

くみこまーはもっと外へ（2）



- より働きやすい職場にしよう
  - 対象領域の知識や経験の幅をひろげよう
    - いろいろな製品・機能の実現
    - いろいろなデバイスの制御
  - ソフトウェア技術者としての知識を高めよう
    - モデリング技術
    - オブジェクト指向技術
    - 分析・設計技術
    - プロジェクトマネジメント
    - 使えるプログラミング言語

まとめ

- 組込みシステムにオブジェクト指向を活用するには
  - 自分たちのポジションを認識する
  - 身近なできることから始め、日常化する

- UMLの活用も身近なところから
  - ホワイトボードでいいじゃない
  - 細かいことまで描くことがモデリングの目的じゃない
  - シーケンス図活用してるんですね、いいですねぇ
    - でもシーケンス図は、仕様書というよりも、仕様を整理する過程で振る舞いを検討するための道具です
  - くみこまーなら状態遷移図に強くなったほうがよいですね

2004/10/18

TECHNOLOGIC ARTS INCORPORATED

「実践！組込みUML～UMLの現状とこれから～」

組込みエンジニアのための UMLモデリング入門

― おしまい ―

久保秋　真（kuboaki@tech-arts.co.jp）

株式会社テクノロジック アート

113-0033 東京都文京区本郷4-1-4コスモス本郷ビル9F

Tel:03-5803-2788 Fax:03-5803-2989

http://www.tech-arts.co.jp/