SSR基調講演

自己適応システムの研究動向

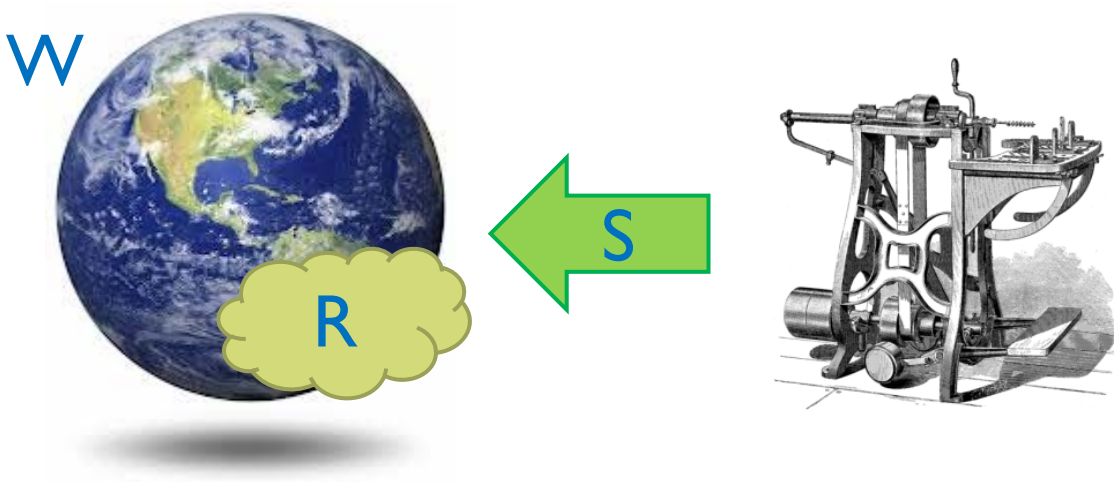
大阪大学 大学院情報科学研究科 中川 博之

2014.07.18

要求とソフトウェア仕様との関係

- Requirements and specifications [Zave97]
  - R = requirements, S = specifications, W = world

$$W \wedge S \vdash R$$



2 [Zave97] P. Zave and M. Jackson, "Four dark corners of requirements engineering", Transactions on Software Engineering and Methodology, 1997.

ソフトウェア進化と自己適応システム

- Requirements and specifications [Zave97]
  - R = requirements, S = specifications, W = world

$$W \wedge S \vdash R$$

- ソフトウェア進化(software evolution):

$$W \wedge S \vdash R \xrightarrow{R \to R'} W \wedge S \not\vdash R' \xrightarrow{S \to S'} W' \wedge S' \vdash R'$$

- 自己適応システム(self-adaptive systems):

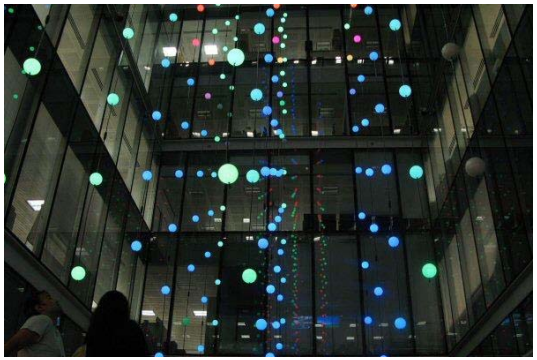
$$W \wedge S \vdash R \xrightarrow{W \to W'} W' \wedge S \not\vdash R \xrightarrow{S \to S'} W' \wedge S' \vdash R$$

自己適応システム

- 自己適応システム (self-adaptive systems)：

$$W \wedge S \vdash R \xrightarrow{W \to W'} W' \wedge S \nvdash R \xrightarrow{S \to S'} W' \wedge S' \vdash R$$

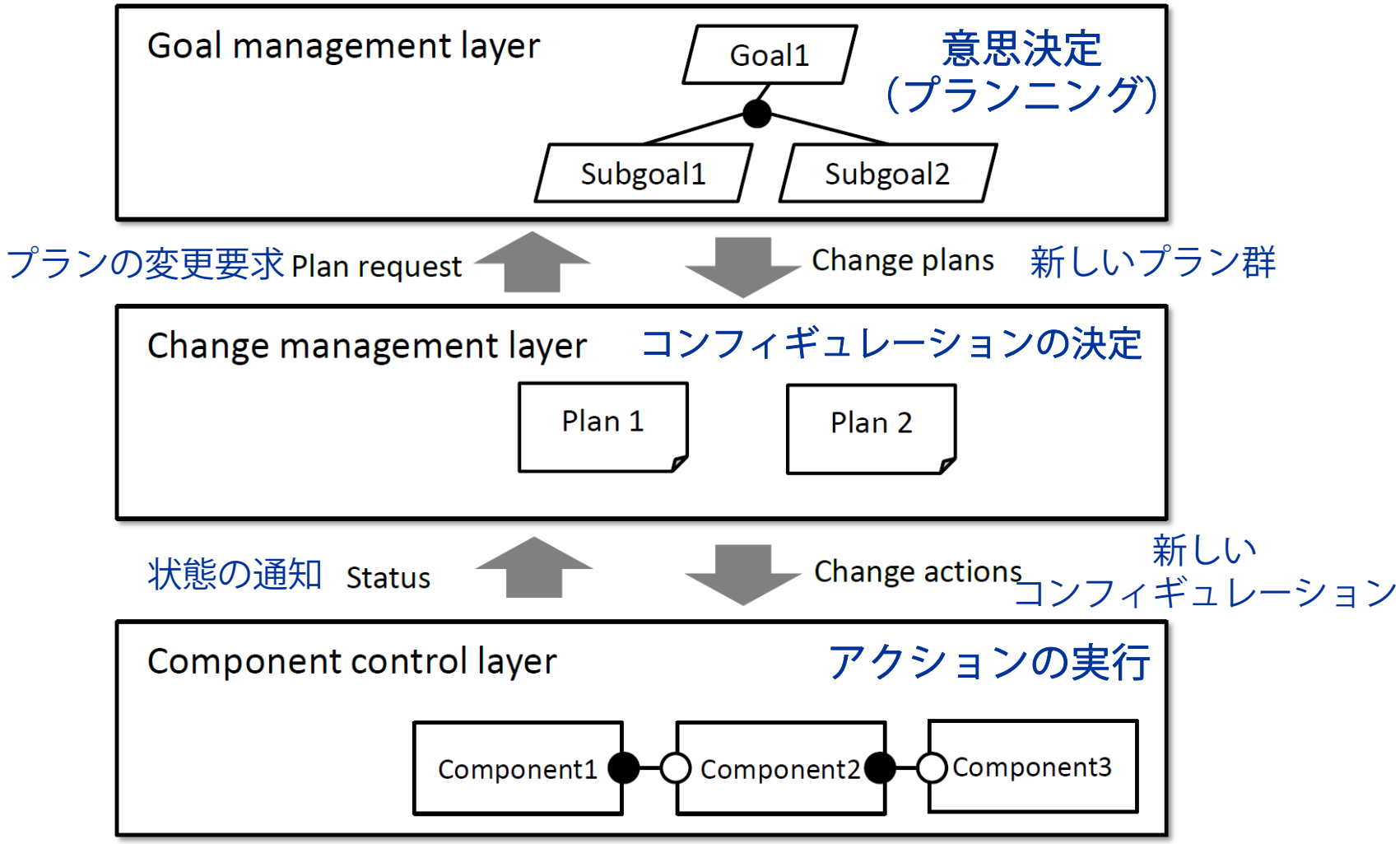
  - 環境や自身の状態変化に対して，構成や振る舞いを自発的に変えることで，変化に適応するシステム
    - パフォーマンスが下がる場合もある
  - 期待されている適用ドメイン
    - クラウドコンピューティング，ユビキタスコンピューティング，制御システム・自律ロボット制御 など


自己適応システムに対する研究アプローチ

- 現在ソフトウェア工学の分野でホットな研究領域の一つ
- 複数の研究アプローチがある
  - モニタリング → 要求工学
  - 構成変更 → ソフトウェアアーキテクチャ
    - 3層アーキテクチャ，MAPEループ
  - 振る舞いの保証 → 動的検証
    - Models@Run.time

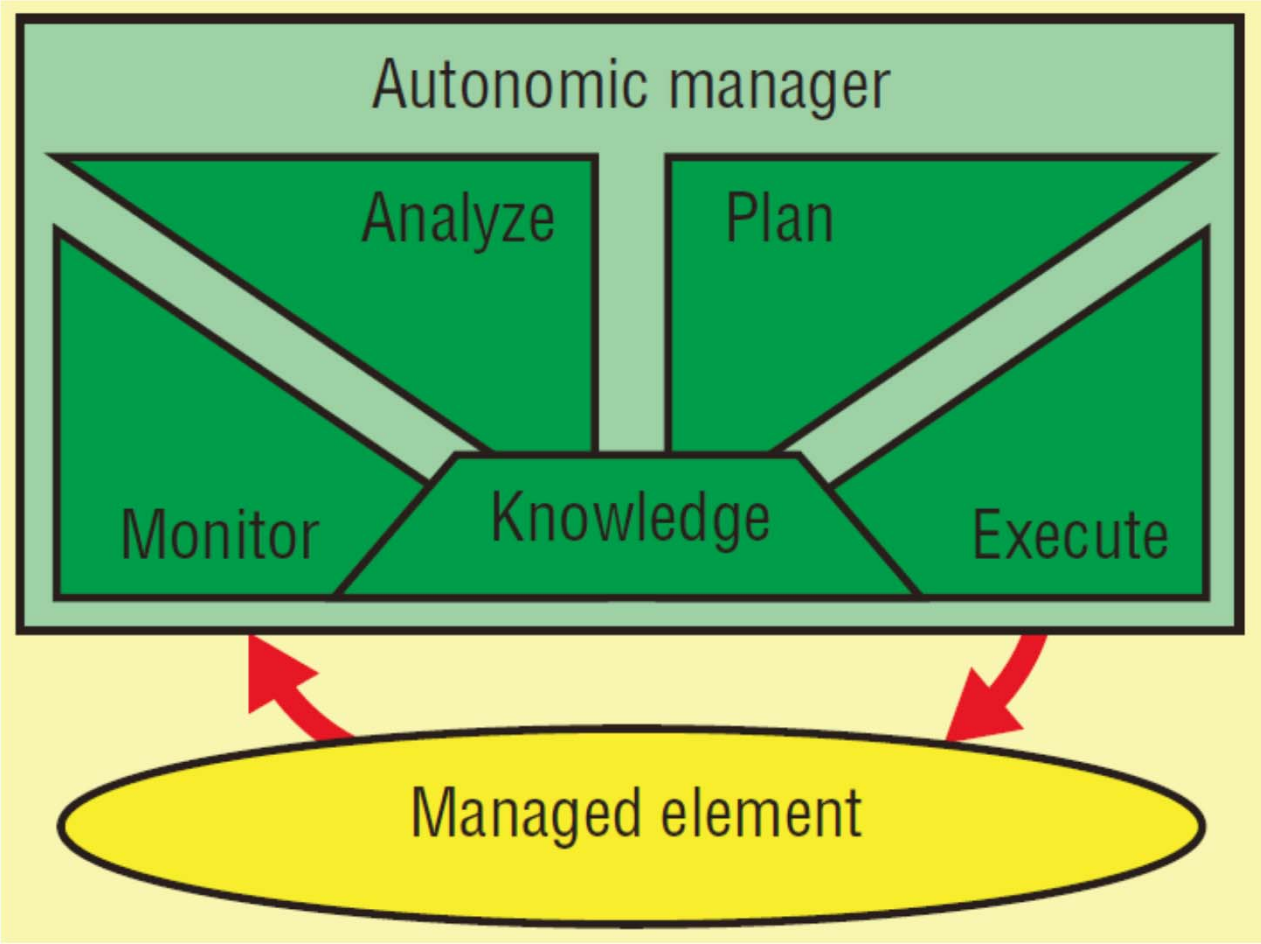
3層アーキテクチャ



Kramer, J., Magee, J. "Self-managed systems: an architectural challenge," ICSE/ FOSE '07.

MAPEループ

- IBMが提唱：Control loopの一種



Kephart, J.O., Chess, D.M. "The vision of autonomic computing," Computer 36(1), pp. 41–50 (2003)

中川のアプローチ

- 自己適応システム(self-adaptive systems):

$$W \wedge S \vdash R \xrightarrow{W \to W'} W' \wedge S \nvdash R \xrightarrow{S \to S'} W' \wedge S' \vdash R$$

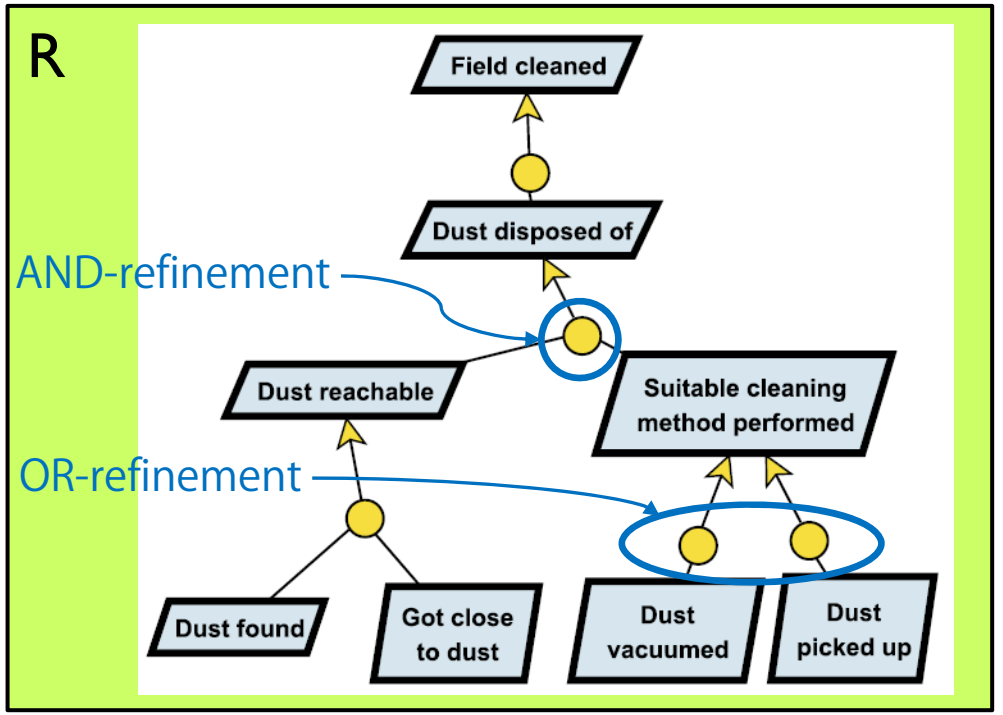
→ W(World), R (Requirements), S(Specification)の関係を強化する

- ゴールモデルによりRequirementsを記述
- W, Sを意識してゴールモデルを整形することで, 自己適応システムと親和性の高いSpecificationを切り出す

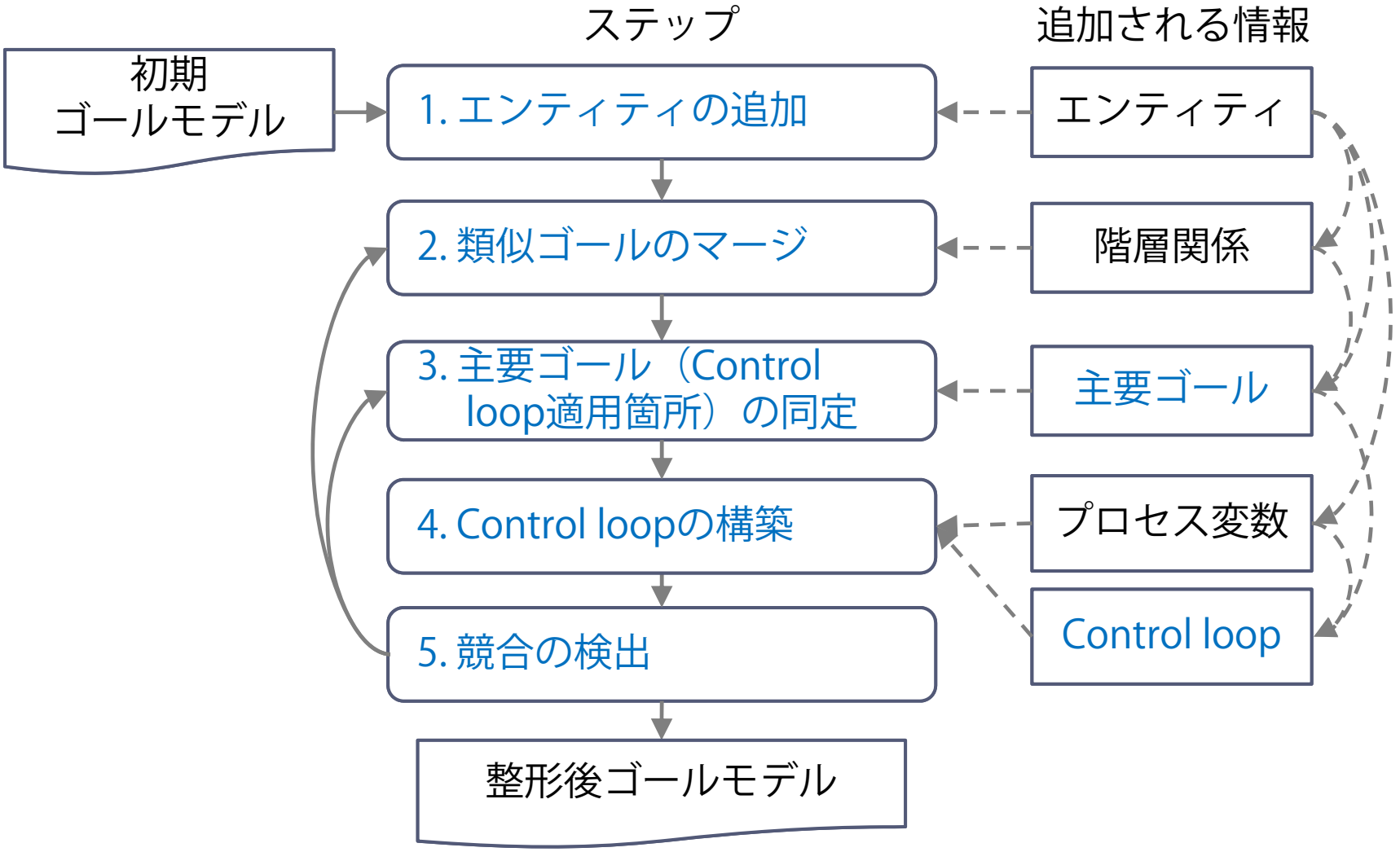
→ ゴールモデルからのControl loopの抽出

R: Requirements

- ゴールモデル：システムに対する要求 (ゴール) を構造化したモデル
  - (AND | OR) -refinement による詳細化
  - システムが達成すべき状態，果たすべき責務を明確化
  - R: 清掃ロボットに関する初期ゴールモデル

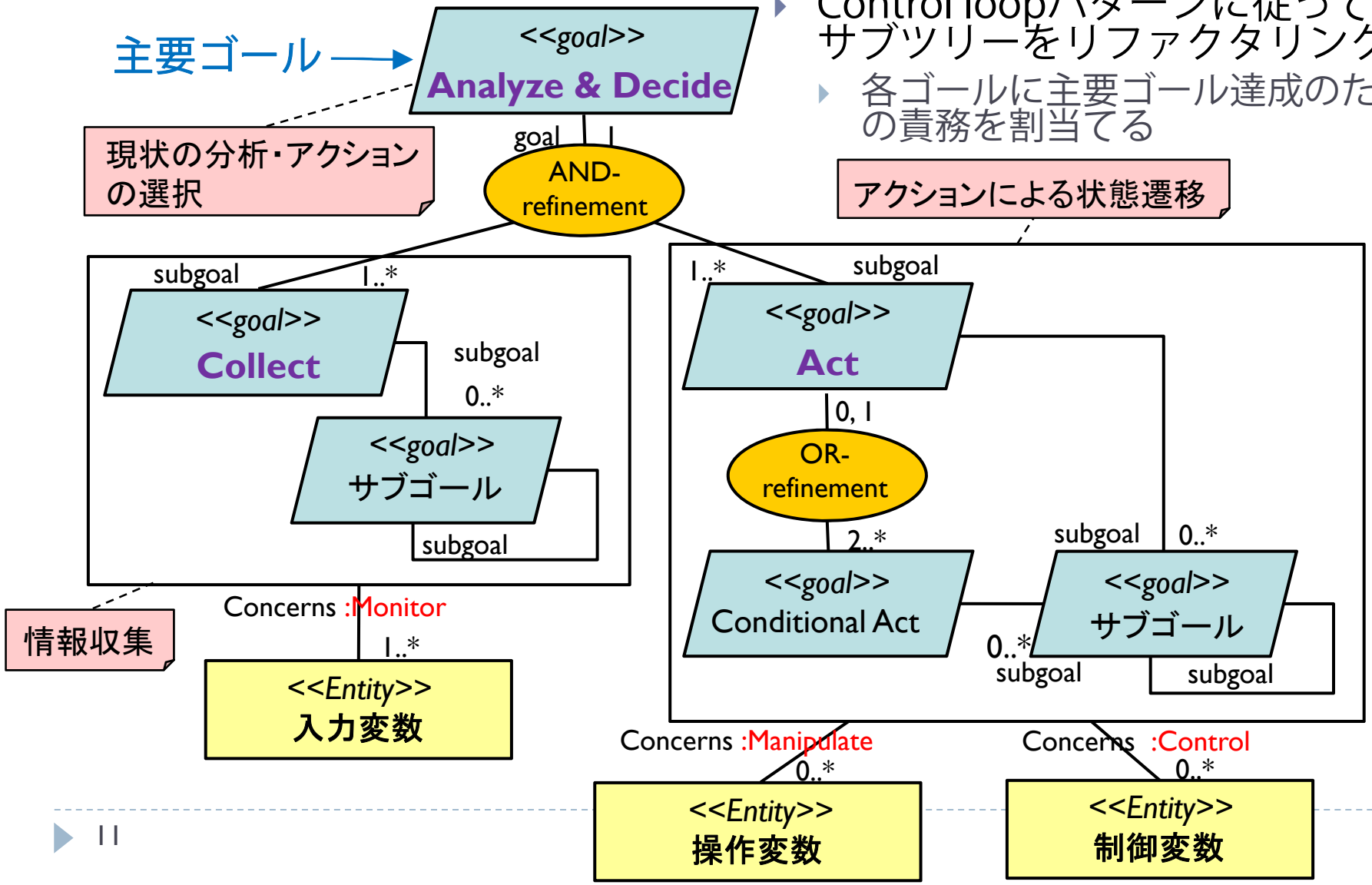


ゴールモデル整形プロセス



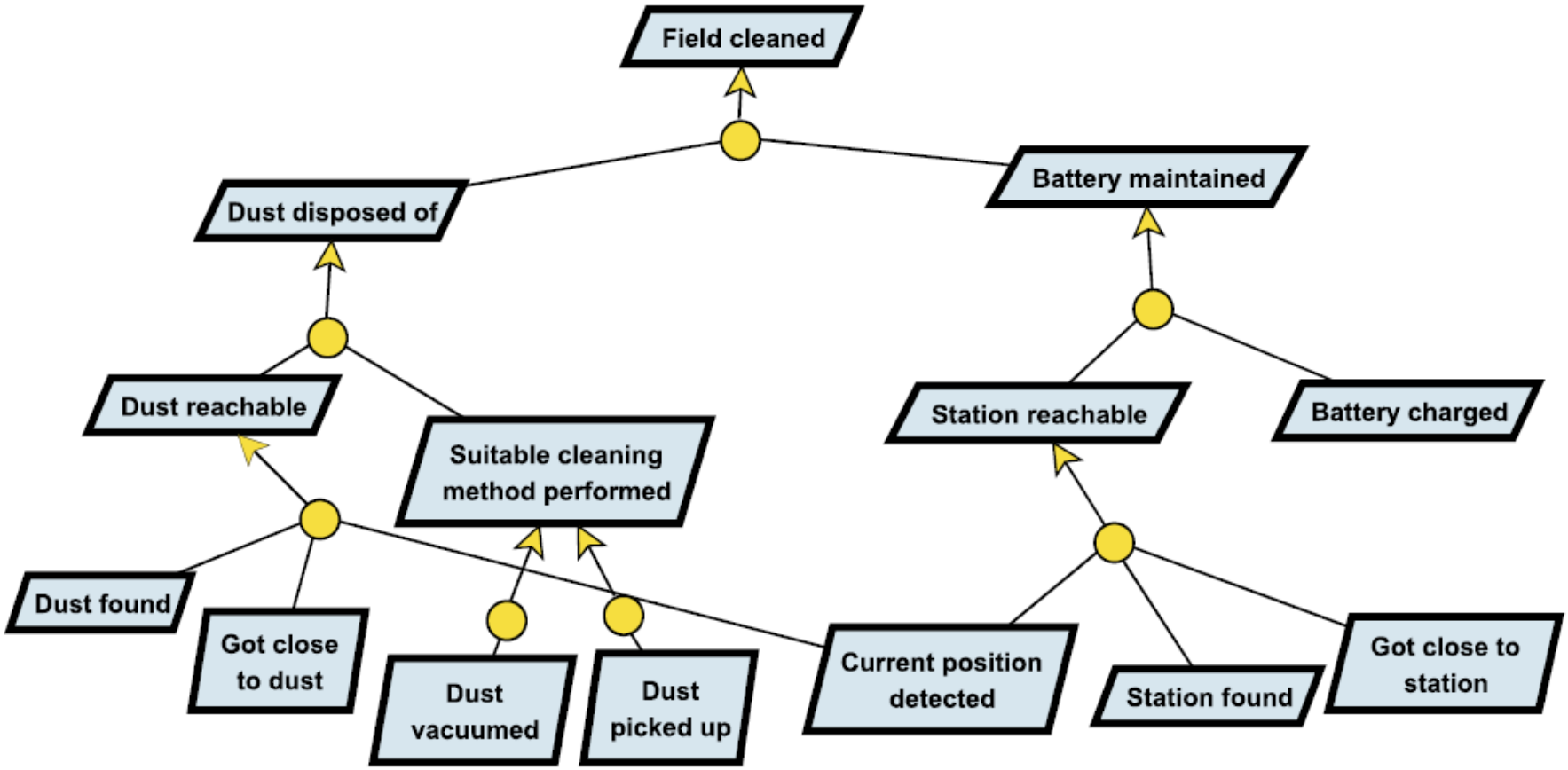
H. Nakagawa, A. Ohsuga, S. Honiden, "A Goal Model Elaboration for Localizing Changes in Software Evolution," RE2013, IEEE, 2013.

Control loopパターン

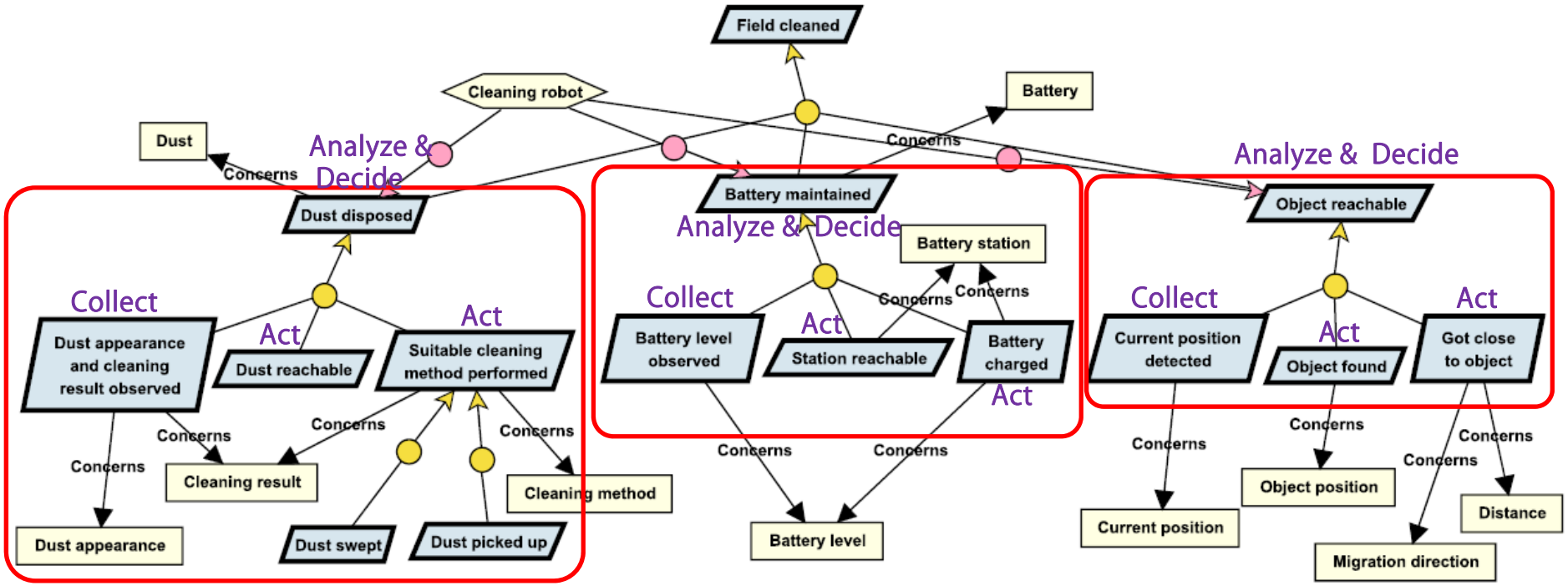


- Control loopパターンに従ってサブツリーをリファクタリング
  - 各ゴールに主要ゴール達成のための責務を割当てる

整形前ゴールモデル


Field cleaned
Dust disposed of
Battery maintained
Dust reachable
Suitable cleaning method performed
Station reachable
Battery charged
Dust found
Got close to dust
Dust vacuumed
Dust picked up
Current position detected
Station found
Got close to station

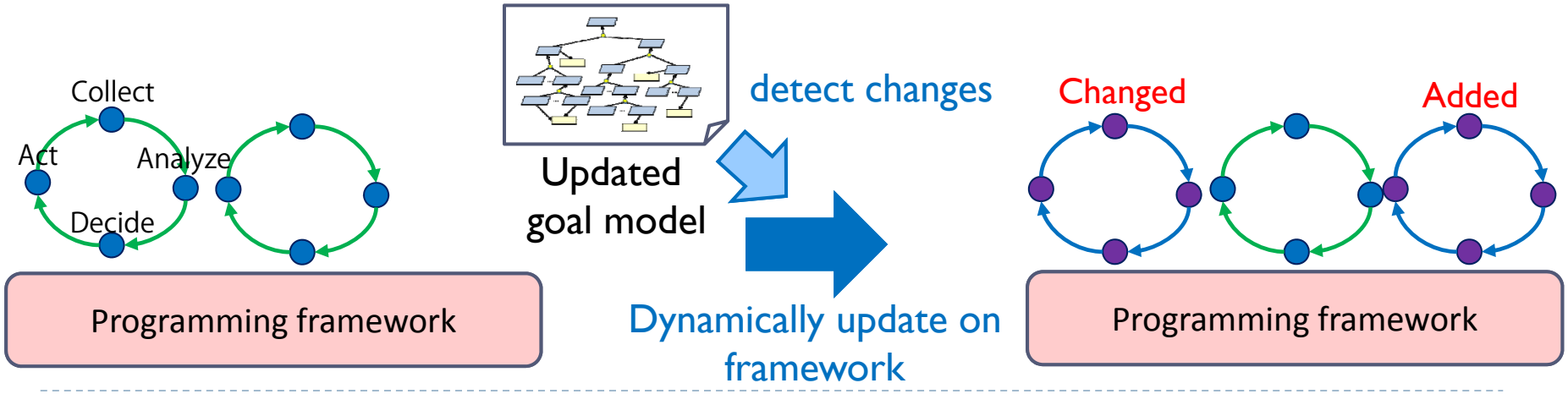
整形後ゴールモデル



Control loop

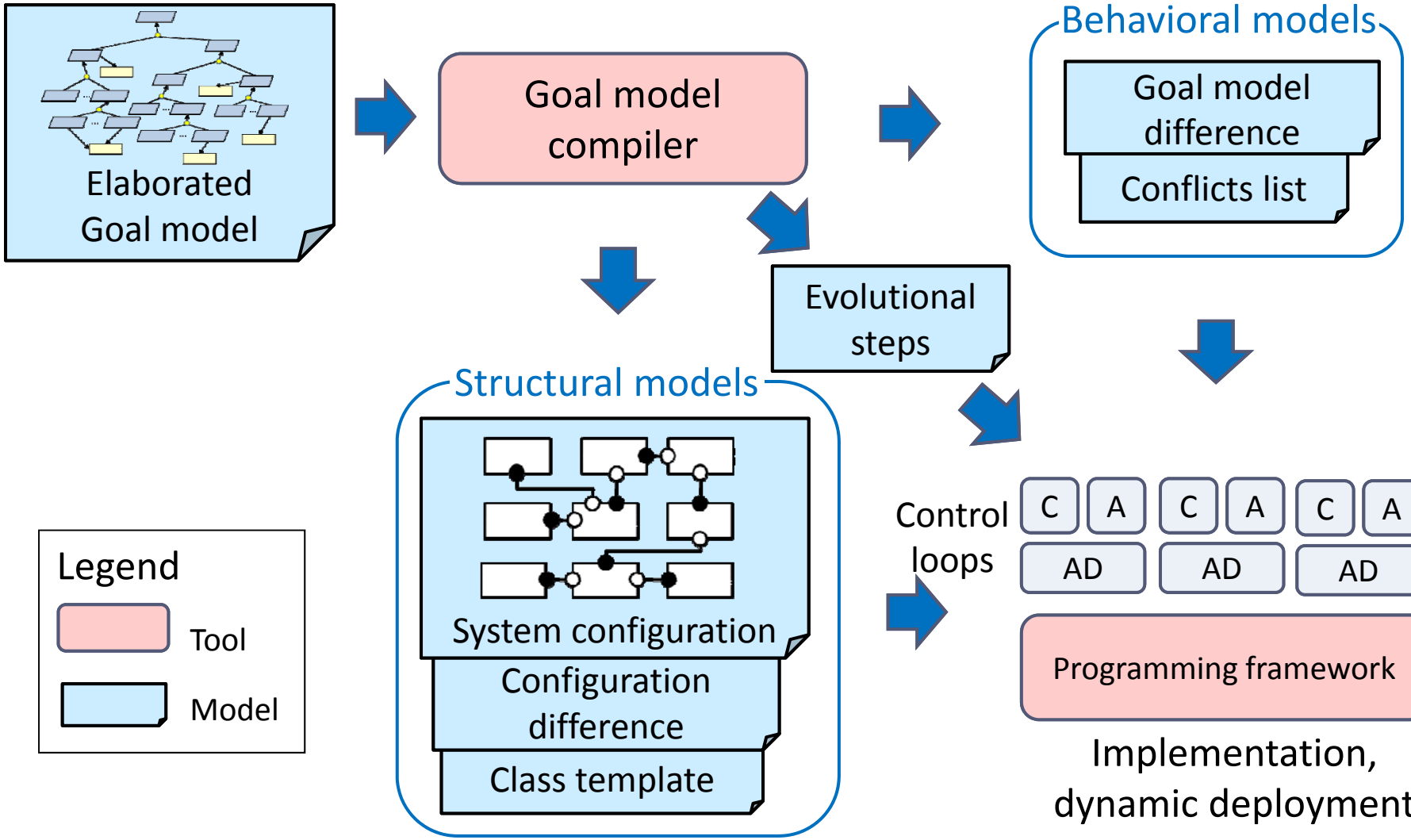
Approach

- Construct self-adaptive systems with multiple control loops
  - Embed or replace control loops for dealing with evolution
- REQ1 → Detect changes from requirements description
  - Identify the changes of control loops from updated goal model
  - Introduce goal model compiler to detect changes of control loops
- REQ2 → Introduce programming framework for supporting dynamic evolution
  - provide APIs for changing control loops dynamically



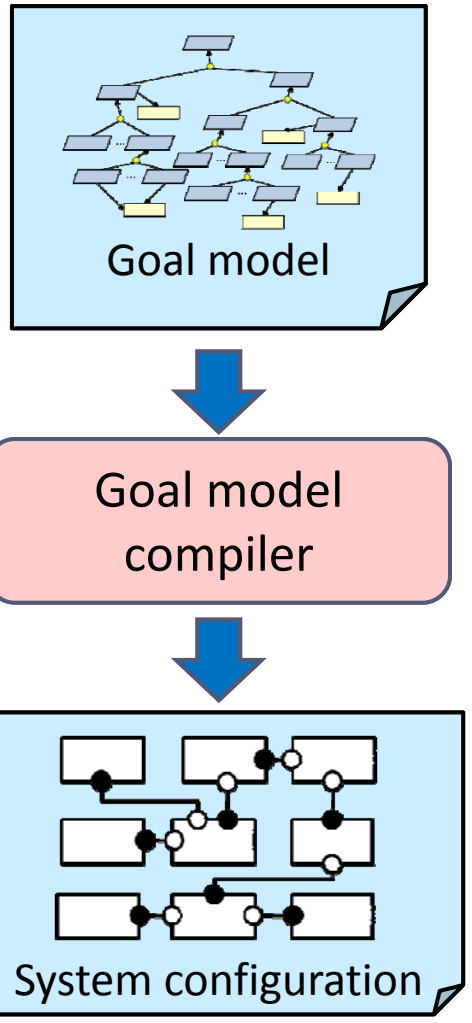
14 H. Nakagawa, A. Ohsuga, S. Honiden, "Towards Dynamic Evolution of Self-adaptive Systems Based on Dynamic Updating of Control Loops," SASO 2012, IEEE, 2012.

Overview of our development process



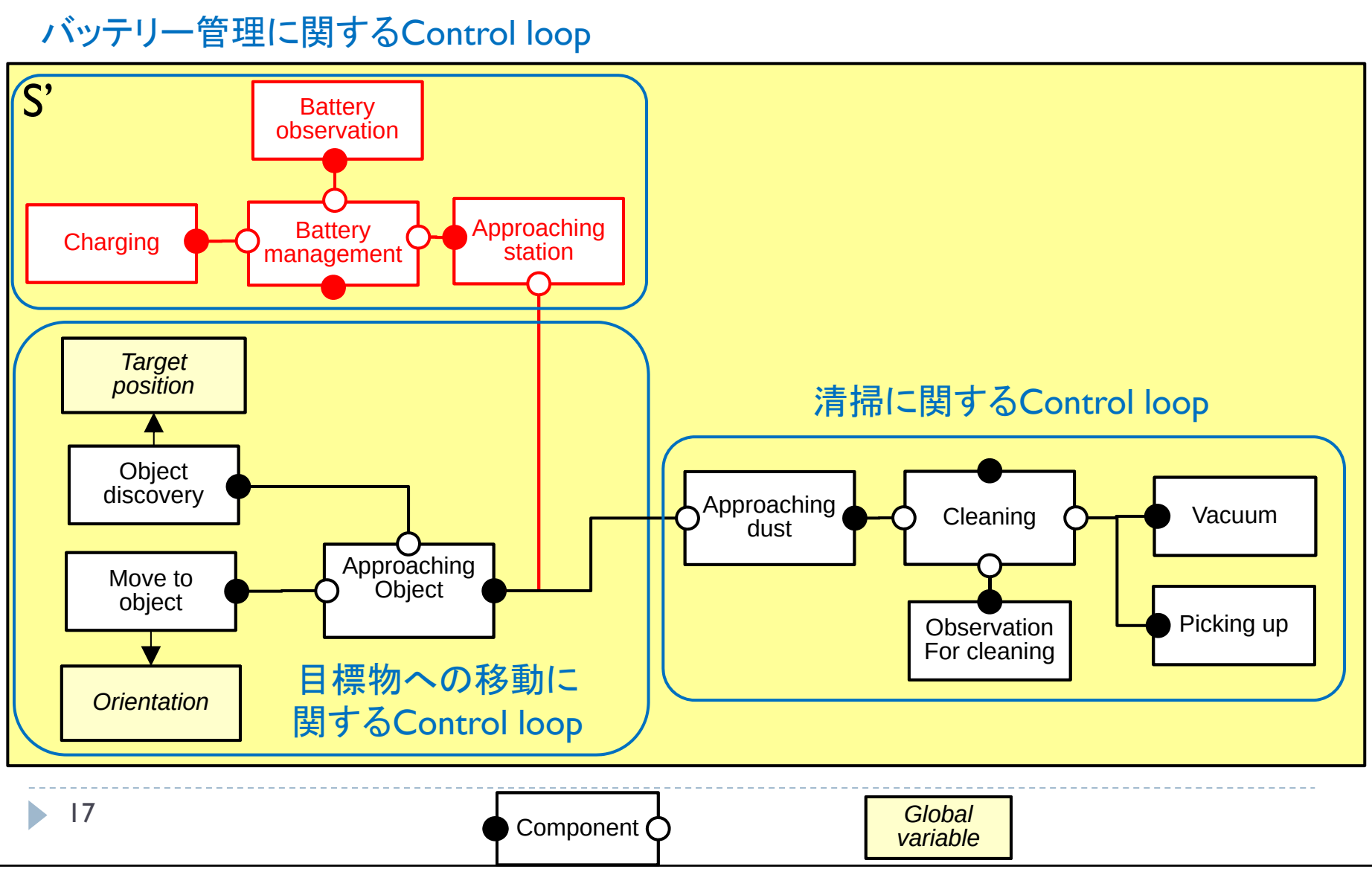
Configuration generation

- Generate architectural configuration from goal model
  - Step1: Construct preliminary configuration according to Control loop pattern
    - Combine goals representing control loop activities with components
    - Connect components according to the refinement links
  - Step 2: Elaborate configuration
    - Join components in accordance with the uses labels
    - Replace the connections of Conditional Act by eliminating intermediate Act

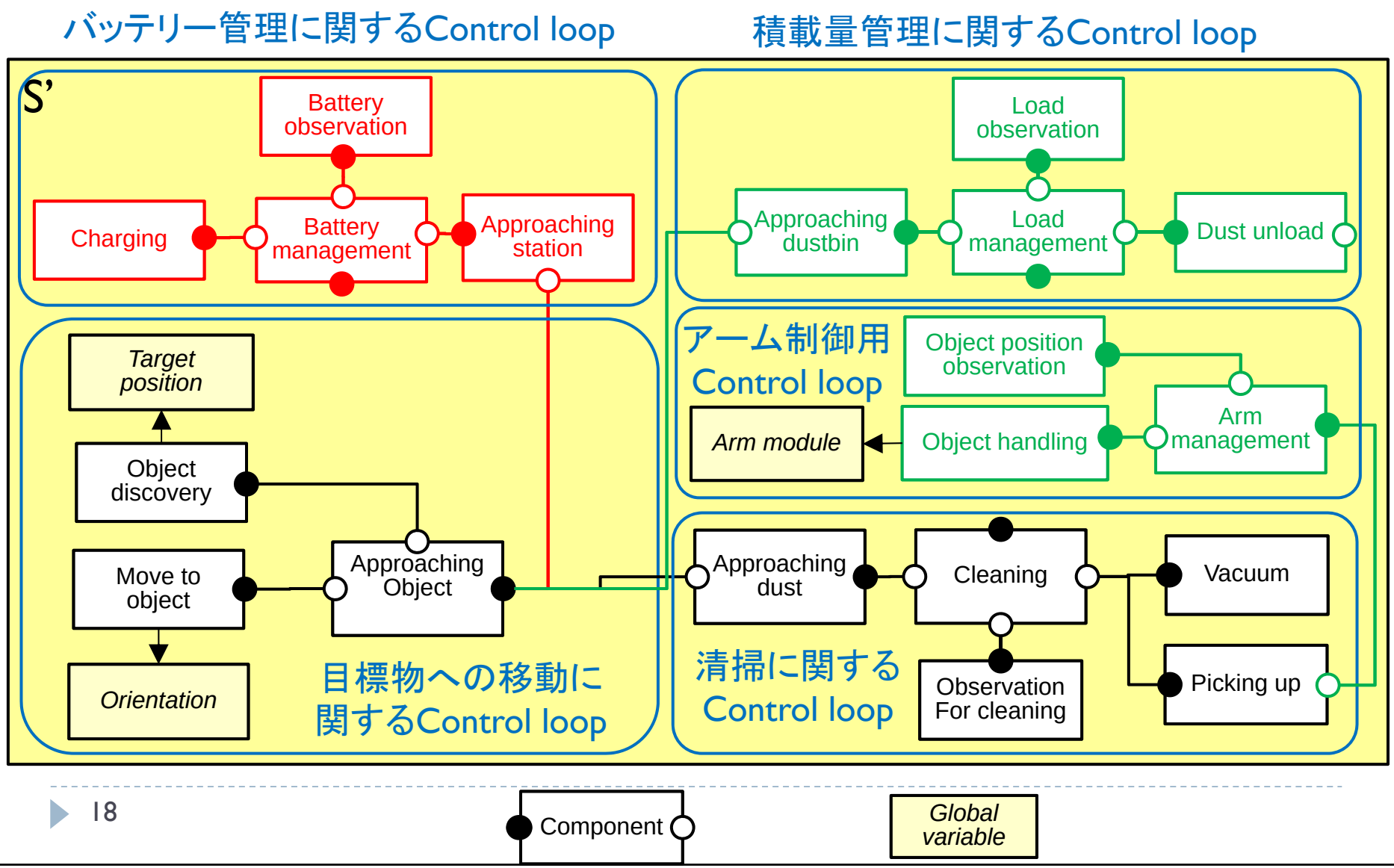


H. Nakagawa, A. Ohsuga, S. Honiden, "gocc: A configuration compiler for self-adaptive systems using goal-oriented requirements description", SEAMS2011, ACM, 2011.

整形後ゴールモデルから生成されるS'

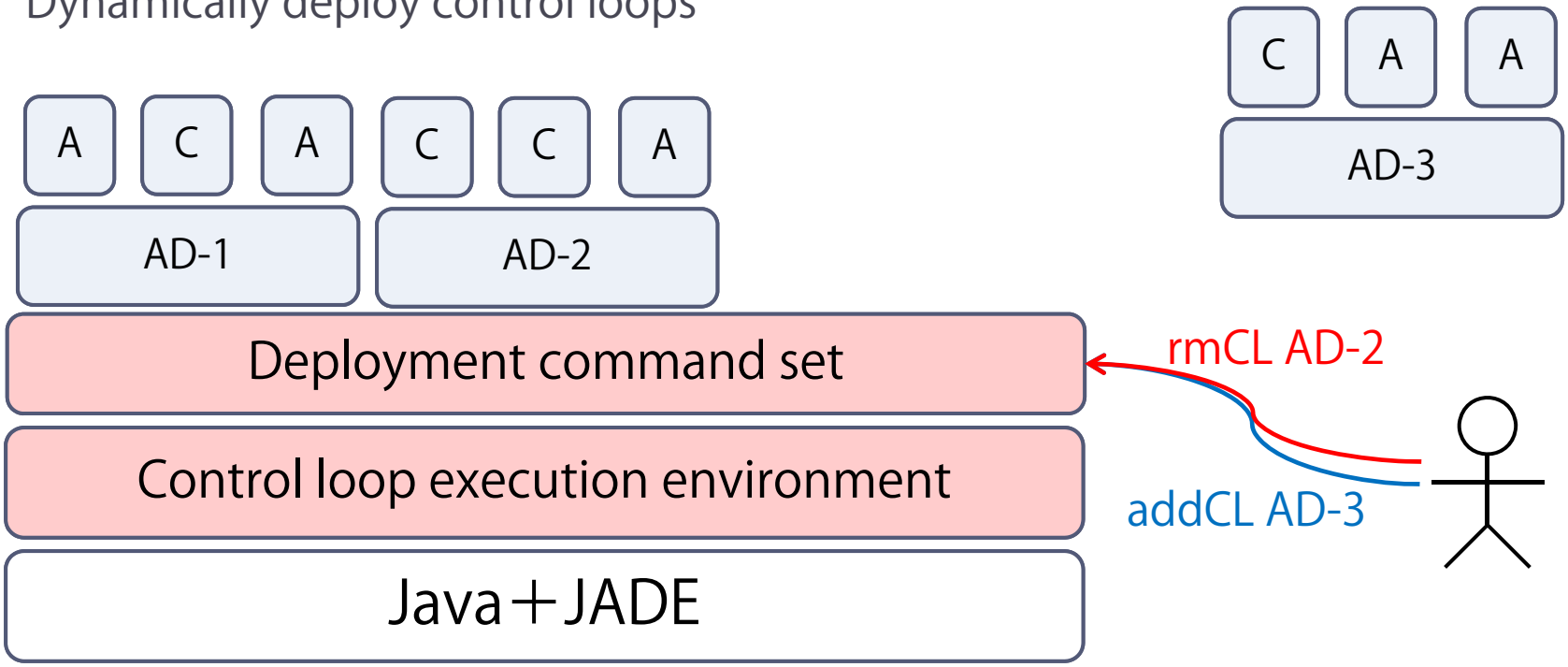


整形後ゴールモデルから生成されるS''



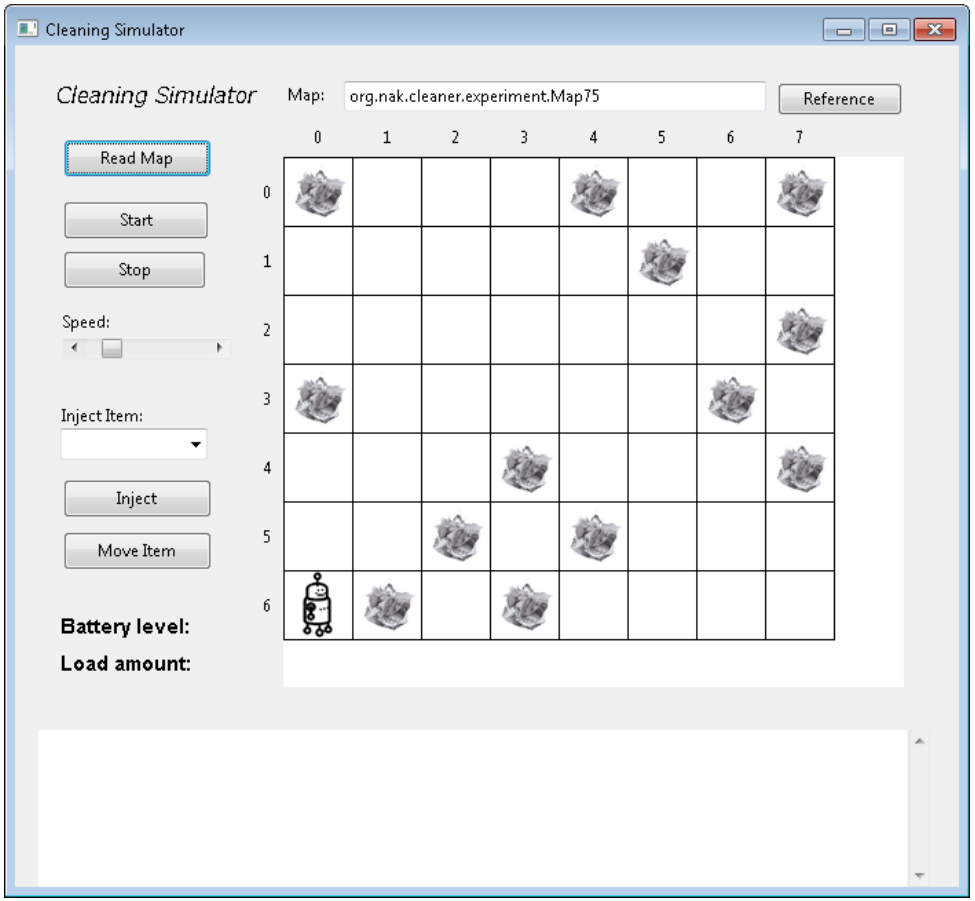
Programming framework

- Control loop execution environment: based on JADE framework [JADE]
  - Concurrent control loop execution environment
  - APIs for implementing three types of components (AD, C, and A)
- Deployment command set
  - Dynamically deploy control loops



[JADE] Telecom Italia. JADE: Java agent development framework. http://jade.tilab.com/.

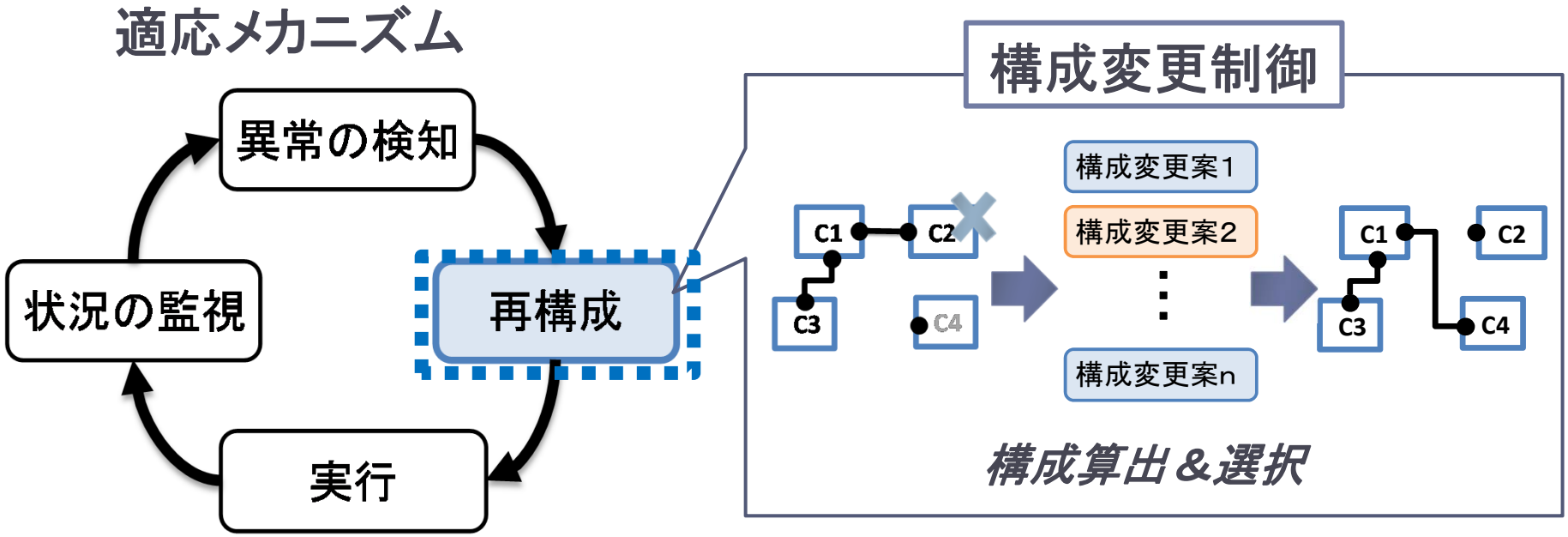
Demonstration:
Dynamic evolution of cleaning robot



- Evolution of cleaning robot
  1. Initial version
     - Does not have the battery Maintenance function
     - Battery level:
       - min=0, max=100
     - Load amount:
       - min = 0, max=200
  2. Add battery maintenance function
  3. Add load management and obstacle avoidance functions
- Dynamic evolution
  - Inject deployment command

大須賀・田原研での学生との研究

自己適応システム構築における課題

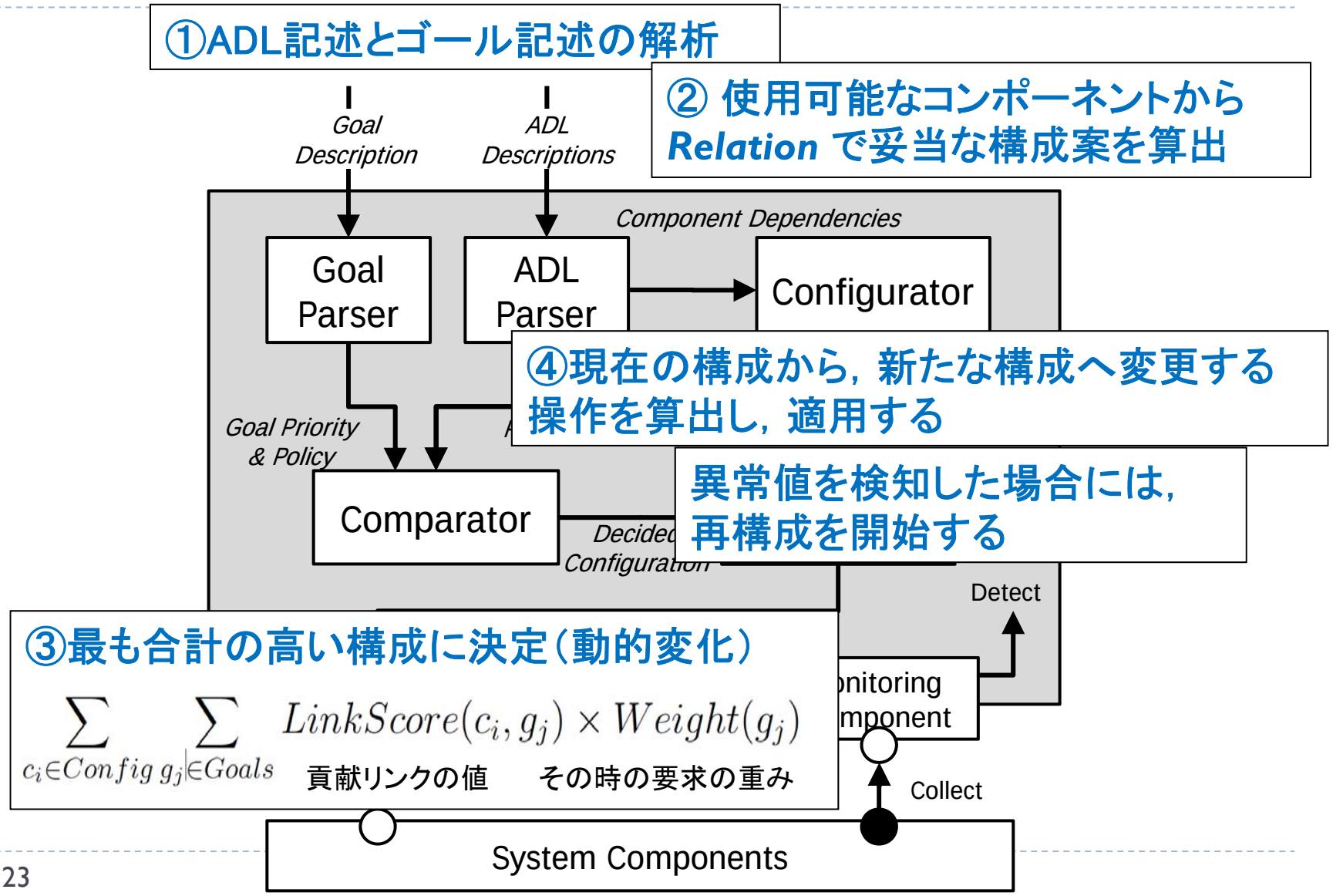


- 変化に対する適応案を<u>自律的に算出</u>する必要
- 状態や要求を考慮し，<u>状況に即した構成選択</u>が重要

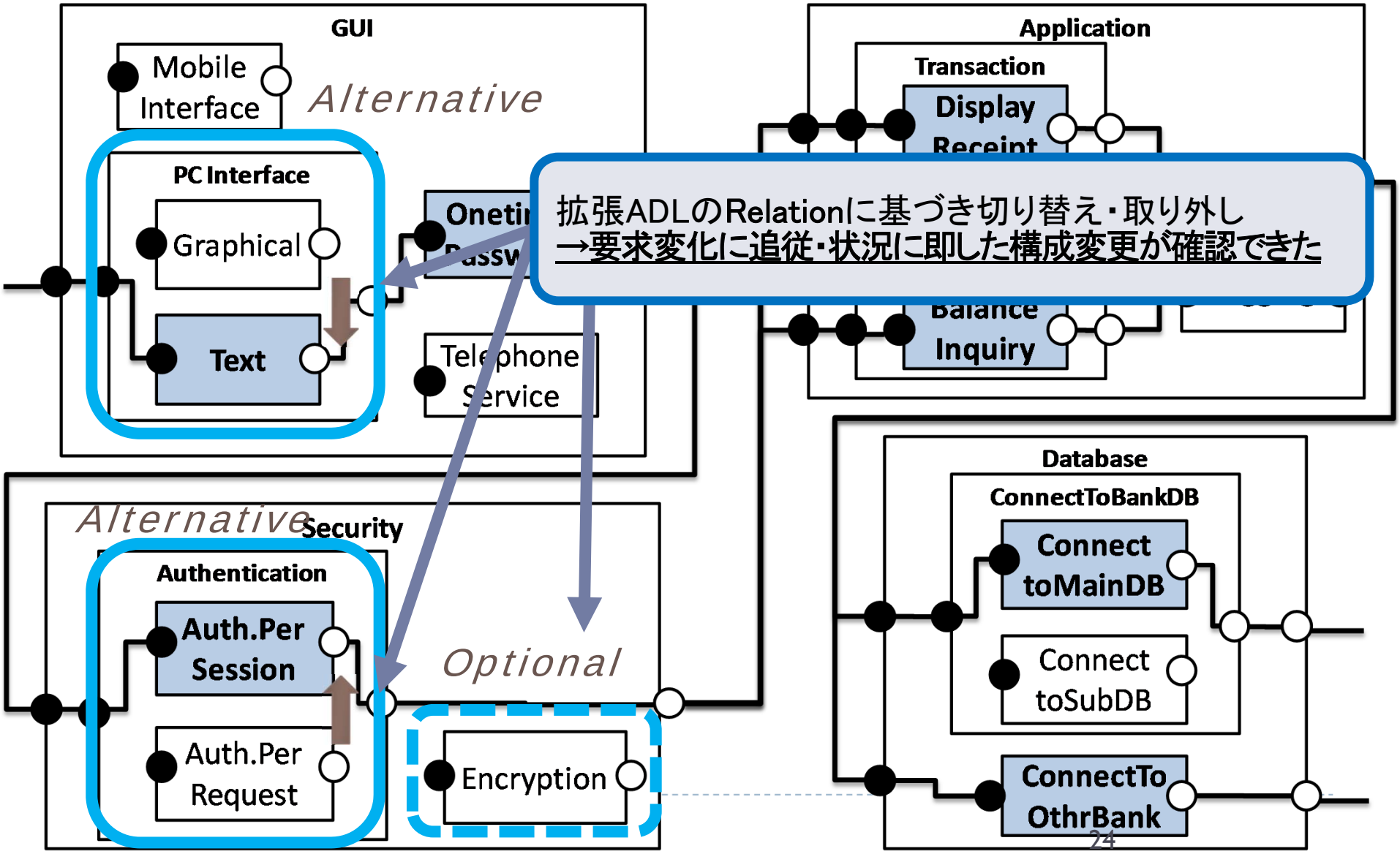
→ <u>自己適応システムの動的再構成手法 および 設計手法</u>

H. Horikoshi, H. Nakagawa, Y. Tahara, A. Ohsuga, “Dynamic Reconfiguration in Self-adaptive Systems Considering Non-functional Properties”, SAC2012, ACM, 2012.

自己適応フレームワーク

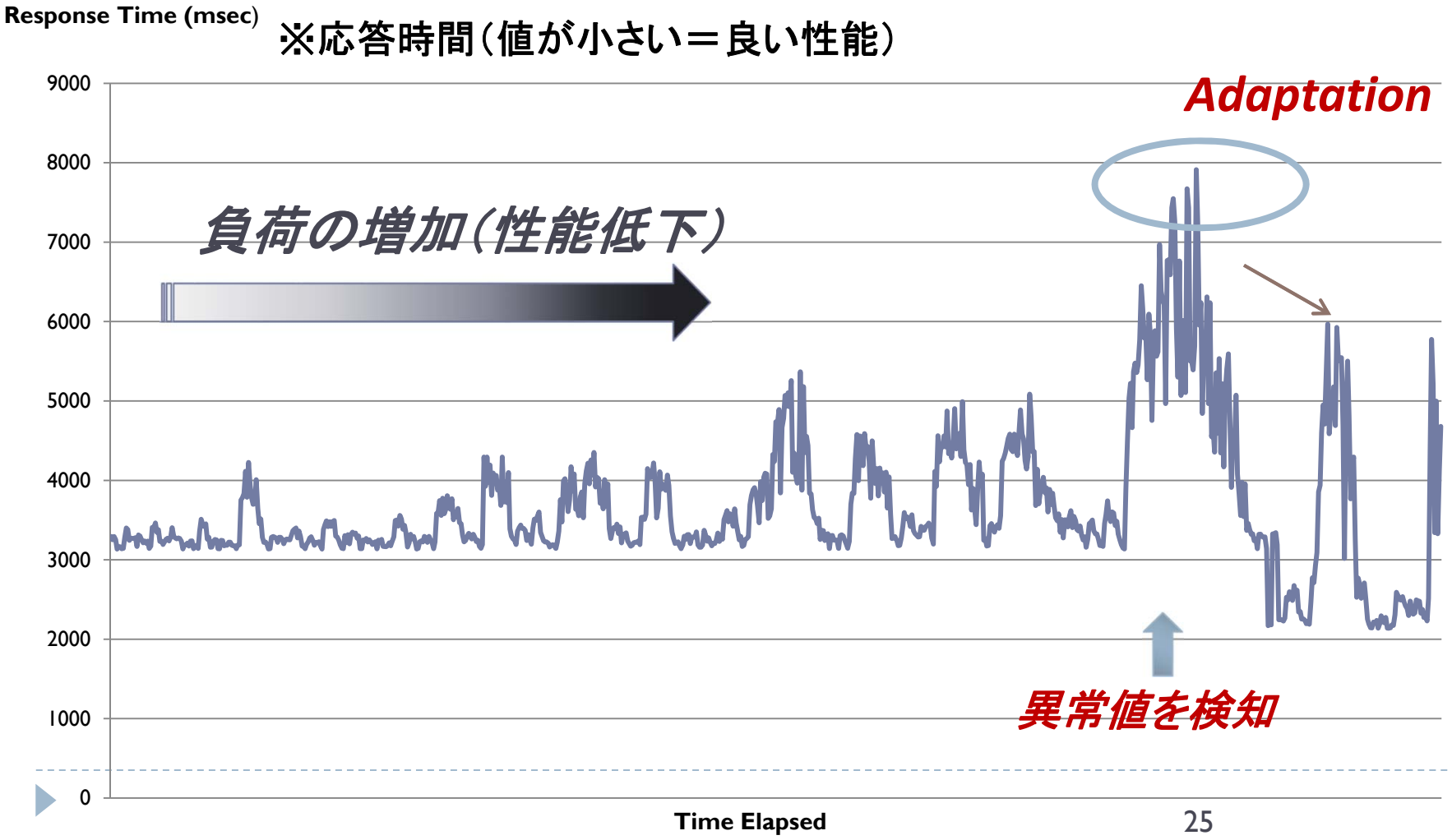


シナリオII　状況変化後



シナリオ II （実行環境変化）

ユーザアクセスを突発的に増加させる負荷実験



オンラインゲームへのMAPEループの適用

- 飽きさせないコンテンツの提供
- 多人数による同時接続
- タイムラグの軽減☆
- ゲーム環境の可用性
- 高いセキュリティの保持

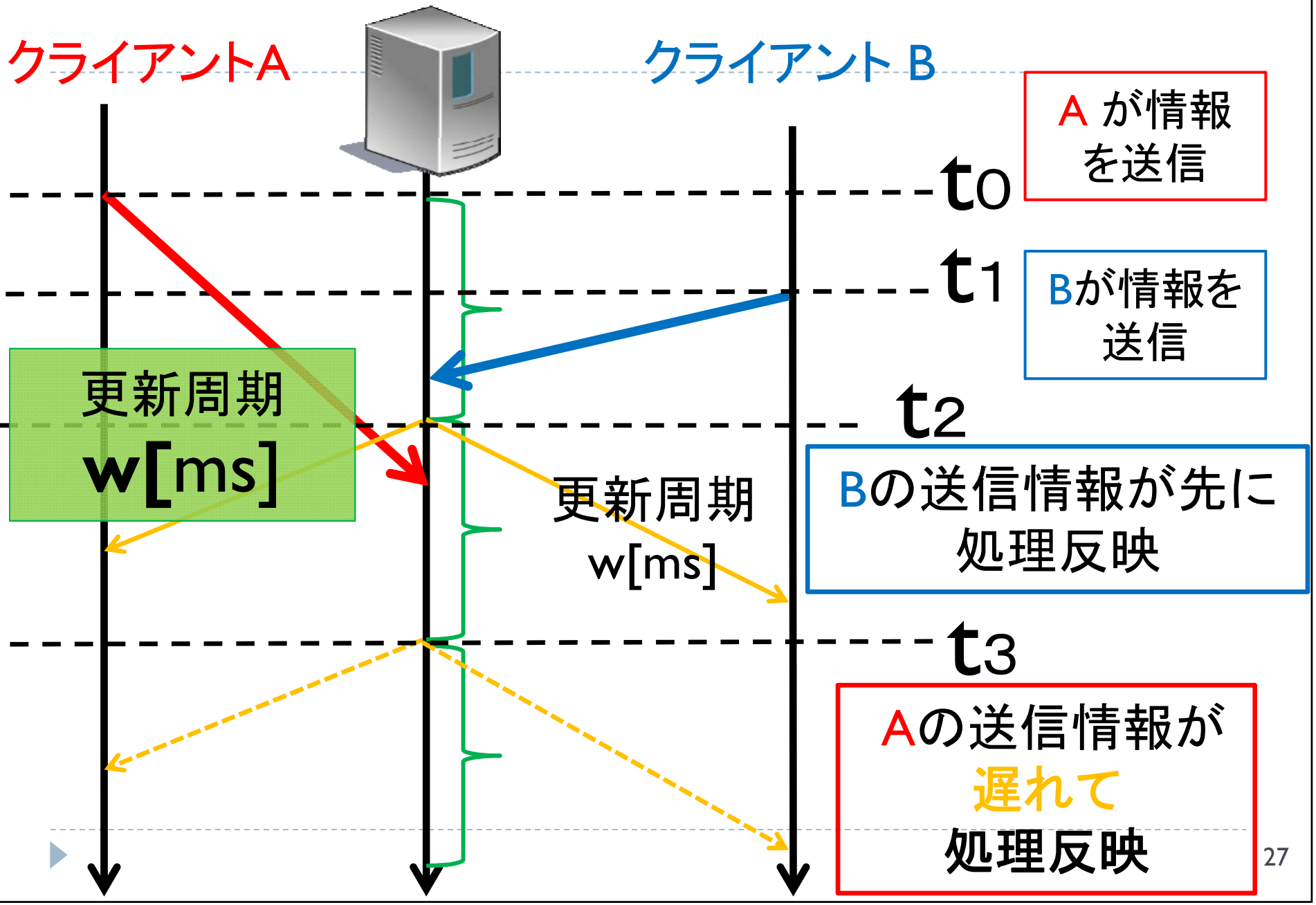
FPS（First Person shooter）

格闘ゲーム

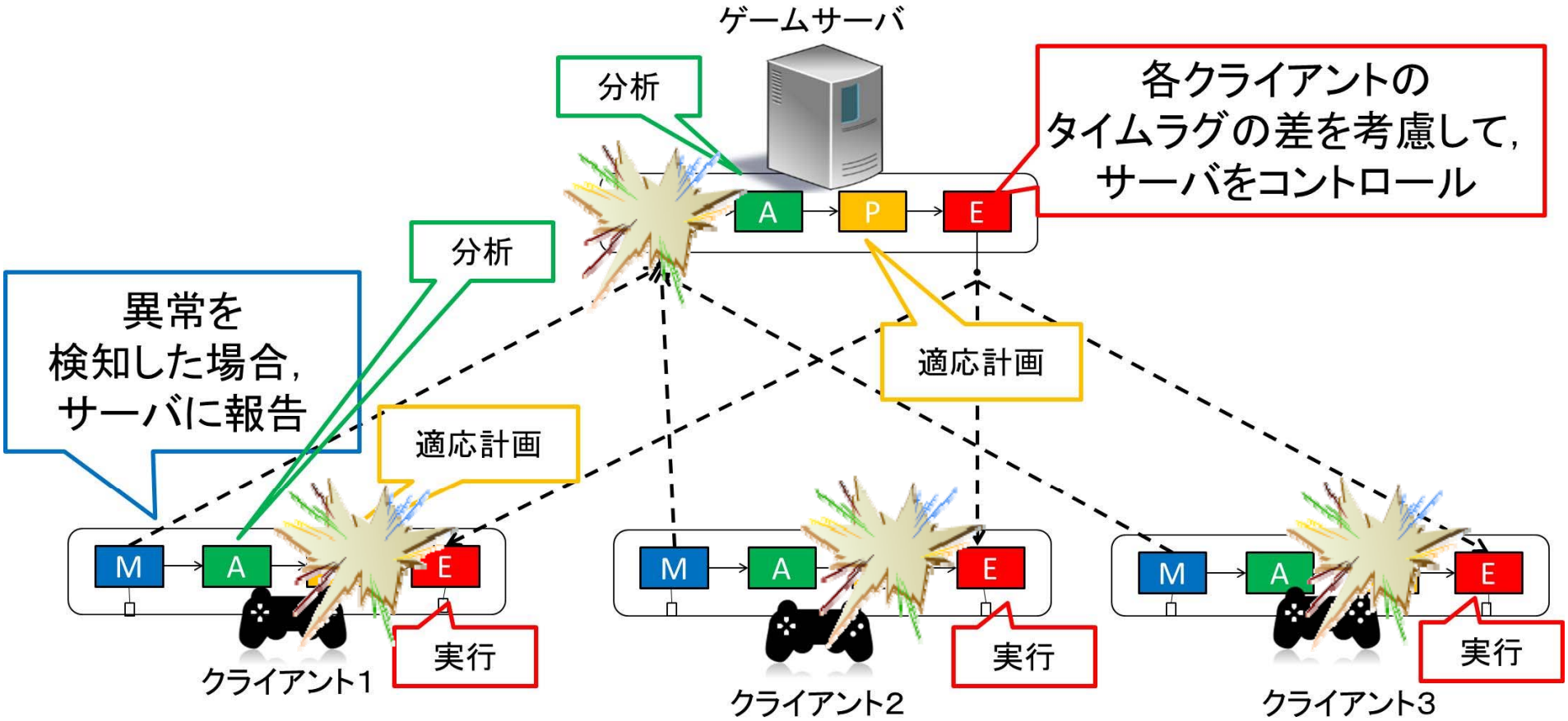
FPS（First Person shooter）や格闘ゲームなどの
リアルタイムで行う**対人型ゲームでは**
タイムラグの対策が重要！

26 S. Yamagata, H. Nakagawa, Y. Sei, Y. Tahara, A. Ohsuga, "A MAPE Loop Control Pattern for Heterogeneous Client/Server Online Games", SEKE2014, 2014.

順序がずれる原因（再掲）


クライアントA
クライアント B
t0
t1
t2
t3
A が情報を送信
Bが情報を送信
更新周期
w[ms]
更新周期
w[ms]
Bの送信情報が先に処理反映
Aの送信情報が
遅れて
処理反映

Hierarchical Controlパターンの課題点

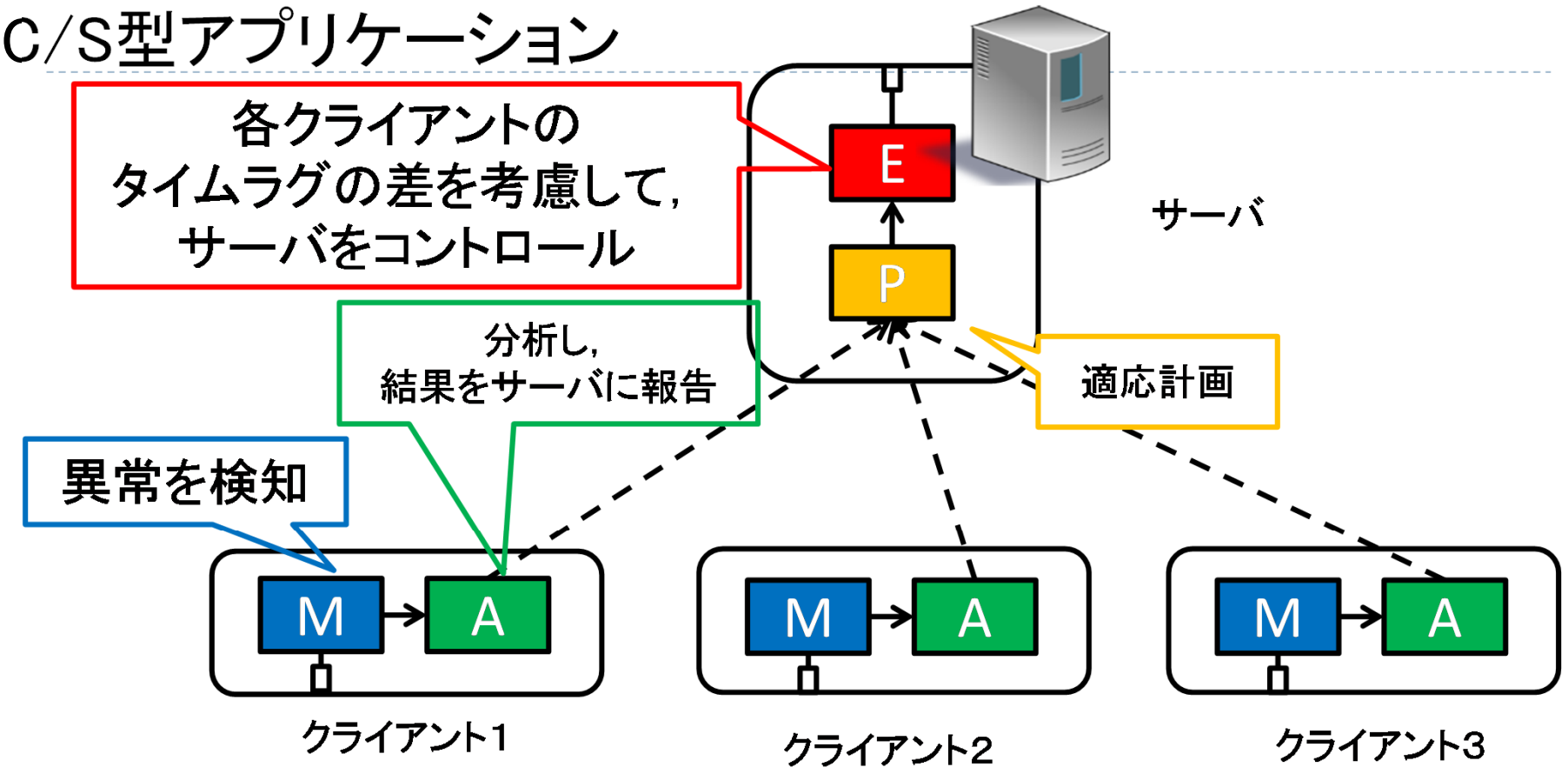


・MAPEループ間の調整が困難
・構成が複雑
・クライアントに負担

Danny Weyns, et al., "On Patterns for Decentralized Control in Self-Adaptive Systems", Software Engineering for Self-Adaptive Systems II, LNCS, 2013.

提案MAPEループ構成パターン

C/S型アプリケーション



・クライアントの環境を監視
・サーバの構成管理が可能
・単純なMAPEループ構成

適応方法の提案

w < w’ [ms]

クライアントA

クライアント B

t’ 0

A が情報を送信

t’ 1

Bが情報を送信

w’[ms]

更新周期変更

w⇒w’[ms]

t3

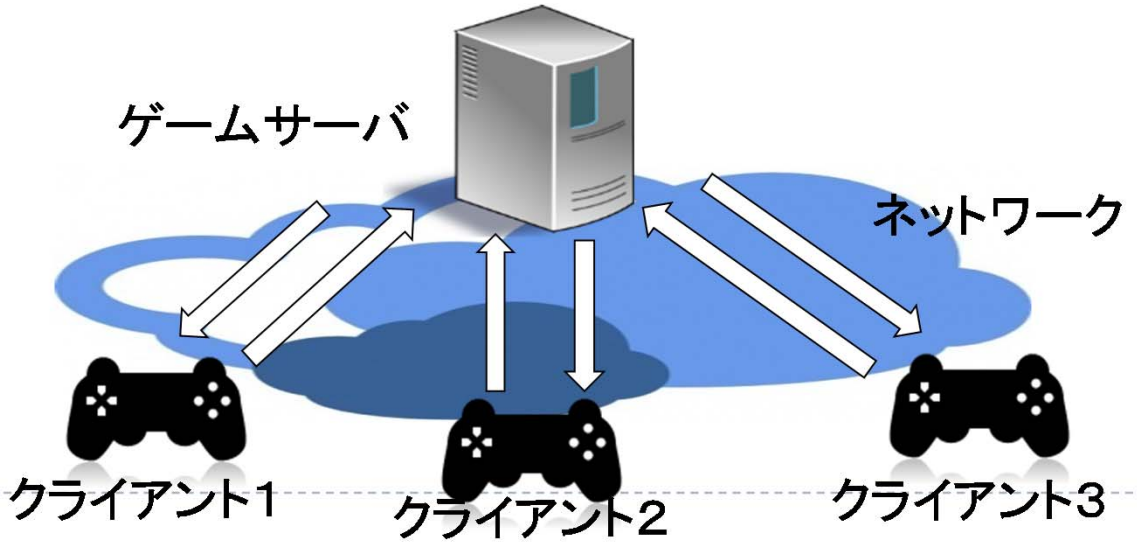
A とBの情報を並び変え処理

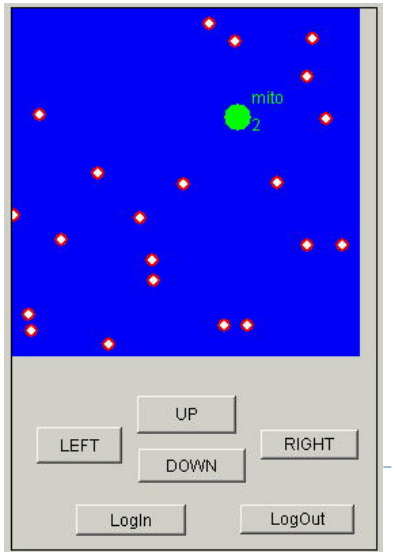
t’ 1（B）よりt’ 0（A）の方が先の行動と処理される

実験

- C/S型簡易オンラインゲーム：点取りゲーム
- TCP/IPを用いたC/S方式
- Javaアプリケーションを使った仮想ネットワーク
- Sleep time関数により，疑似的タイムラグを発生

- 各クライアントに同じタイミングで同じ行動させ，
  異なるデータの共有(データの矛盾)の回数を確認





結果

実験1：2者間(+40[ms])による実験
実験2：3者間(+40[ms],+80[ms])による実験

表1　平均更新周期 120[ms]

	自己適応なし	自己適応
実験1（矛盾検出数）	46	2
実験2（矛盾検出数）	18	3

表2　平均更新周期 150[ms]

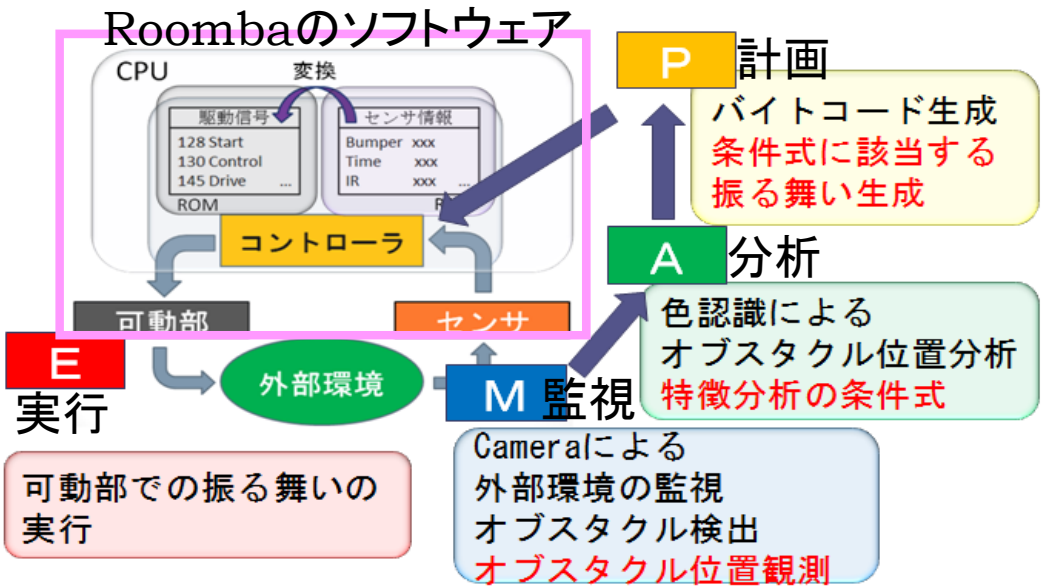
	自己適応なし	自己適応
実験1（矛盾検出数）	58	6
実験2（矛盾検出数）	15	7

自己適応することにより矛盾検出数が減少

自動清掃ロボットの機能拡張

- 自己適応のメカニズム(MAPEループ)を利用したRoombaの機能拡張
  - 課題:小物類やゴミなどのオブスタクル移動
    - オブスタクル移動のAndroidアプリを実装
  - 制約:組込みプログラムを変更できない
    - システムに外付けするMAPEループの特徴

移動できるようになったオブスタクル



工藤拓光, 自己適応のメカニズムを用いた自動清掃ロボットの機能拡張に関する研究, 電気通信大学 修士論文, 2014.

まとめ

- 自己適応(ソフトウェア)システム
  - 次世代のソフトウェア構築手法を目指すフラグシップ
  - ソフトウェア工学分野の各方面からのアプローチ
    - 要求工学, ソフトウェアアーキテクチャ, 動的検証
  - 自己適応システムに関する主要会議
    - SASO (IEEE International Conference on Self-Adaptive and Self-Organizing Systems)
    - SEAMS (International Symposium on Software Engineering for Adaptive and Self-Managing Systems)
    - その他, SE分野でのトップカンファレンス
      - ICSE, ASE, RE, MODELS, CAiSE, …

→ 現段階では，自己適応システム自身を構築するよりも，1つの例題として要素技術を深化させることが重要