A Goal Model Elaboration for Localizing Changes in Software Evolution

Hiroyuki Nakagawa, Akihiko Ohsuga, Shinichi Honiden

RE 2013, July 18th, 2013, Rio de Janeiro

Background: Software evolution

- Software evolution: activity for adapting to requirements changes
  → Play central role in overall software lifecycle
  - Recent topics: **continuous** software evolution
    - Software product lines [Northrop01]
    - Continuous delivery [Humble10]

- Requirements and specifications [Zave, Jackson97]
  - W = world, R = requirements, S = specifications

$$W, S \vdash R$$

- (Continuous) software evolution:

$$\cdots \rightarrow W, S \vdash R \xrightarrow{W \rightarrow W'} W', S \not\vdash R \xrightarrow{R \rightarrow R'} W', S \not\vdash R' \xrightarrow{S \rightarrow S'} W', S' \vdash R' \rightarrow \cdots$$

[REQ] Keep cost(S→S′) low under continuous evolution

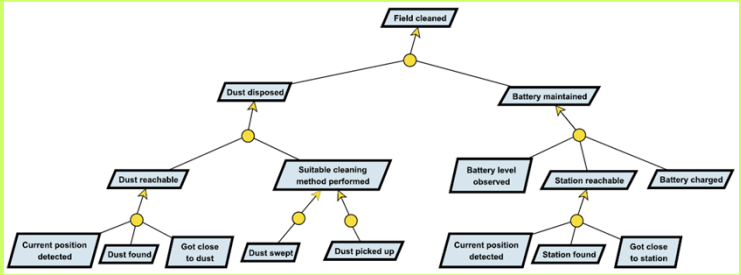
Example: cleaning robot evolution

- W: states of world
- R: goal model description
- S: system architecture

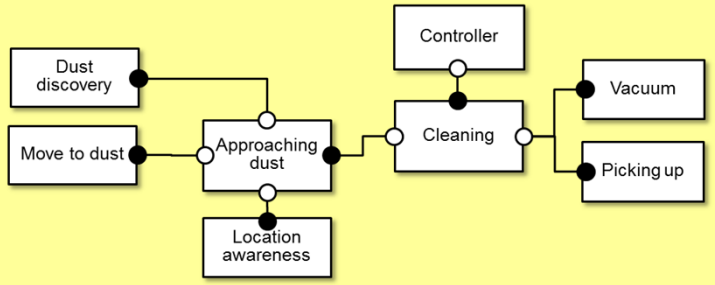
W

Dust items	{((x1, y1), empty can), ((x2, y2), litter), ((x3, y3), house moss)}
Position of robot	(0, 1800)

R

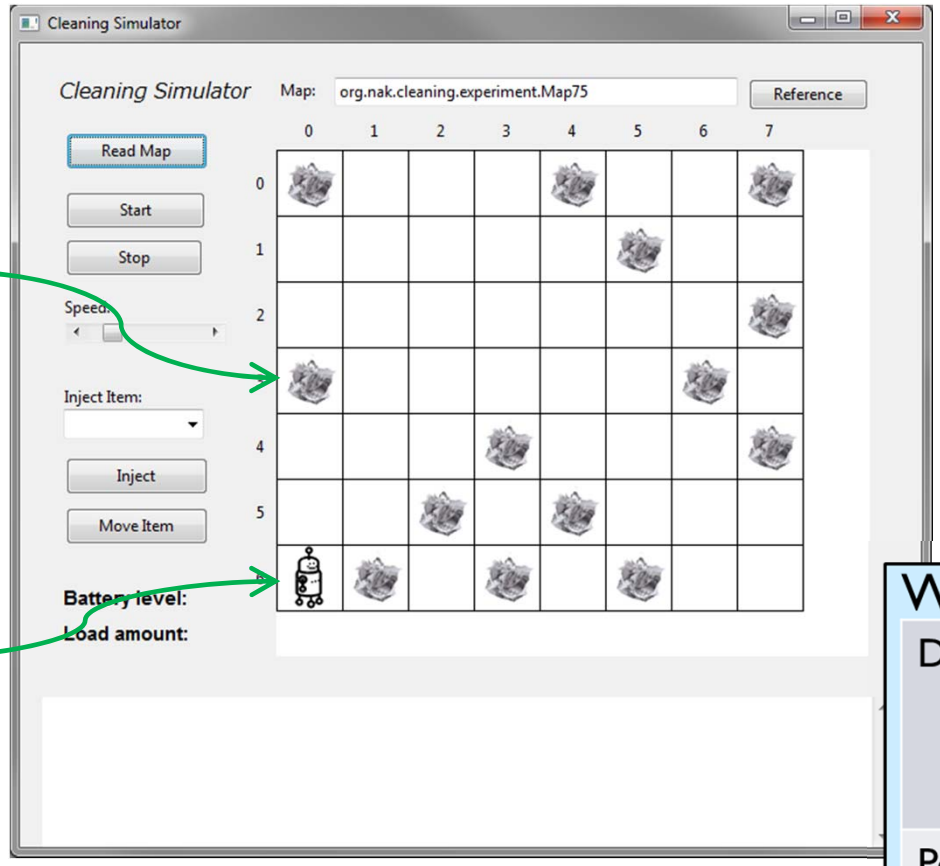


S



W in this study

- W: cleaning robot and dust items (in initial dev.)



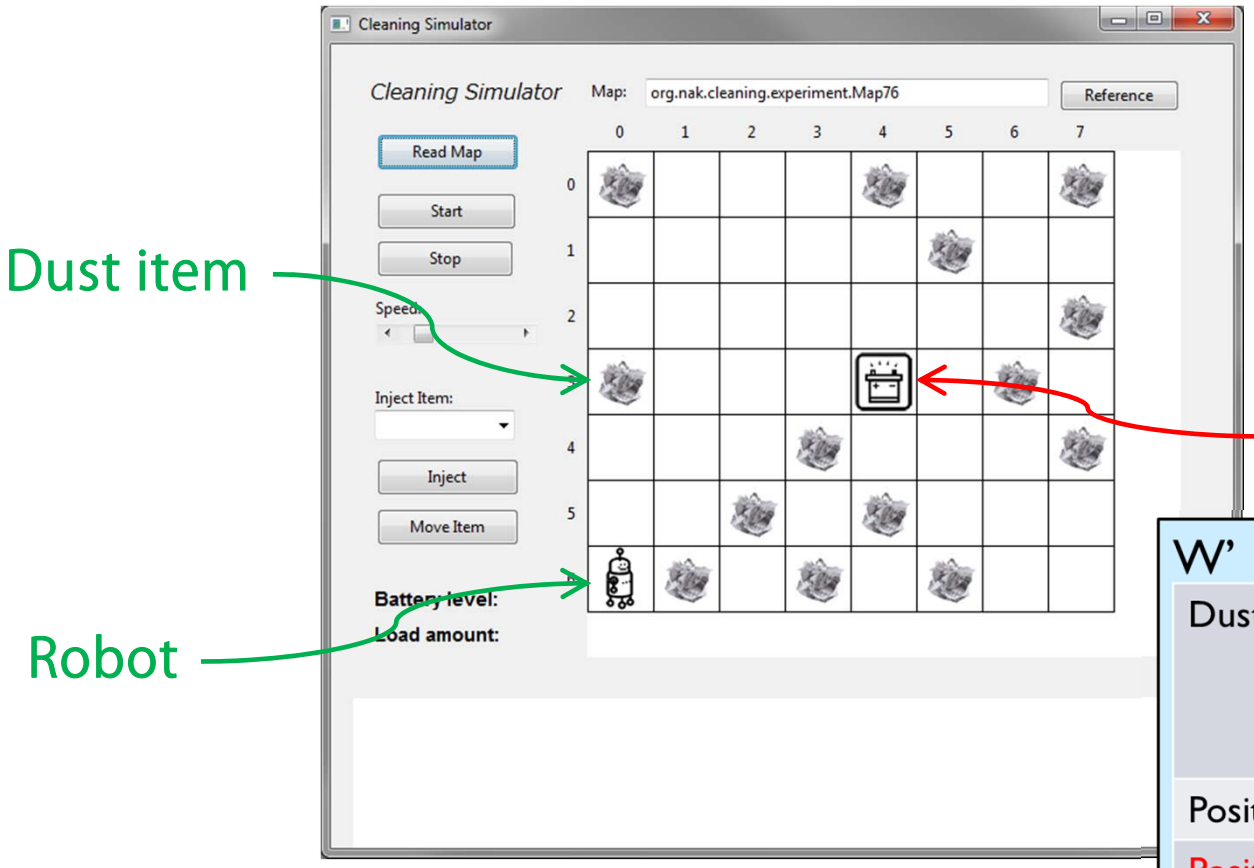
Dust item

Robot

W	
Dust items	{((x1, y1), empty can), ((x2, y2), litter), ((x3, y3), house moss)}
Position of robot	(0, 1800)

World has changed (W → W')

- W': (charge) station is installed



W'

Dust items	{((x1, y1), empty can), ((x2, y2), litter), ((x3, y3), house moss)}
Position of robot	(900, 600)
Position of station	(1200, 900)
Battery level	76 (%)

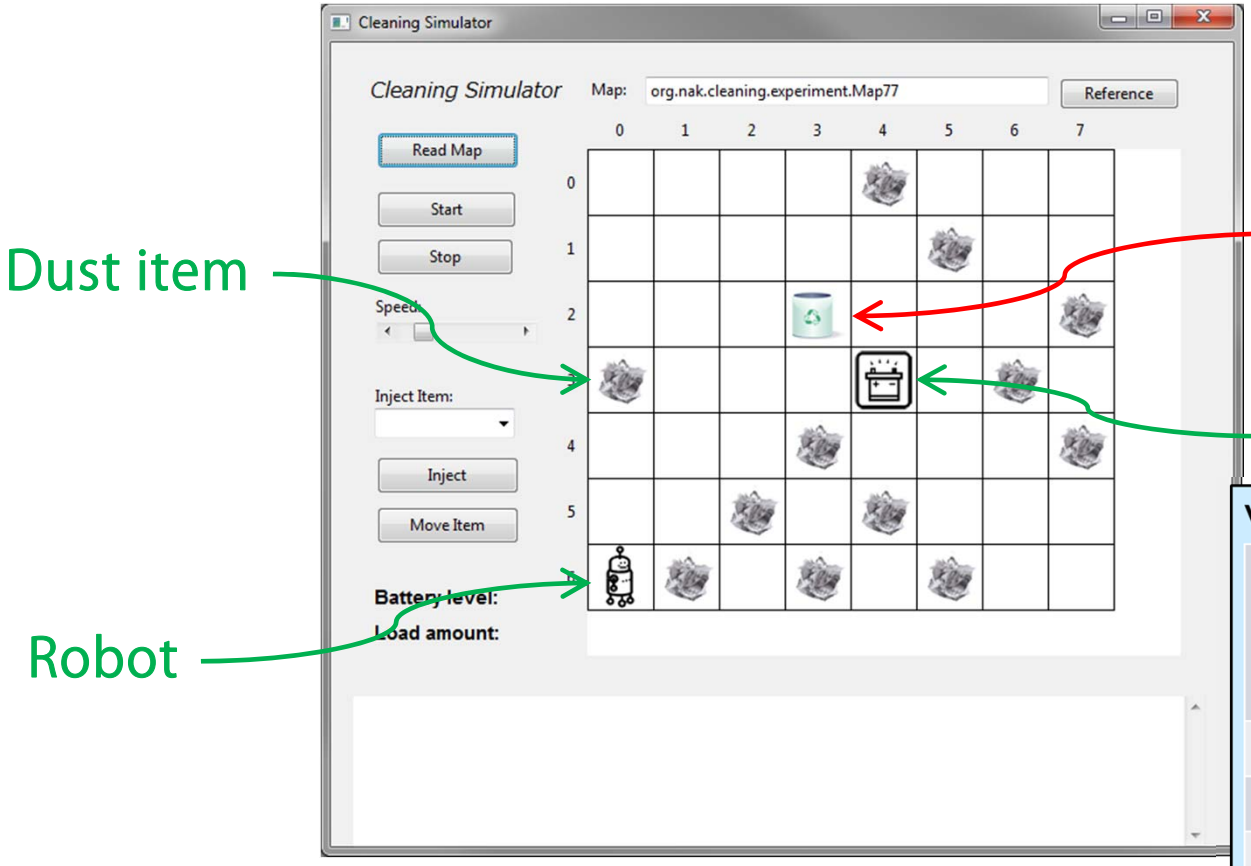
Wc

Wd

Wc = common(W', W), Wd = diff(W', W)

World has changed (W' → W'')

- W'': dustbin is installed



W''

Dust items	{((x1, y1), empty can), ((x2, y2), litter), ((x3, y3), house moss)}
Position of robot	(0, 1800)
Position of station	(1200, 900)
Battery level	76 (%)
Position of dustbin	(900, 600)
Load amount	40 (%)

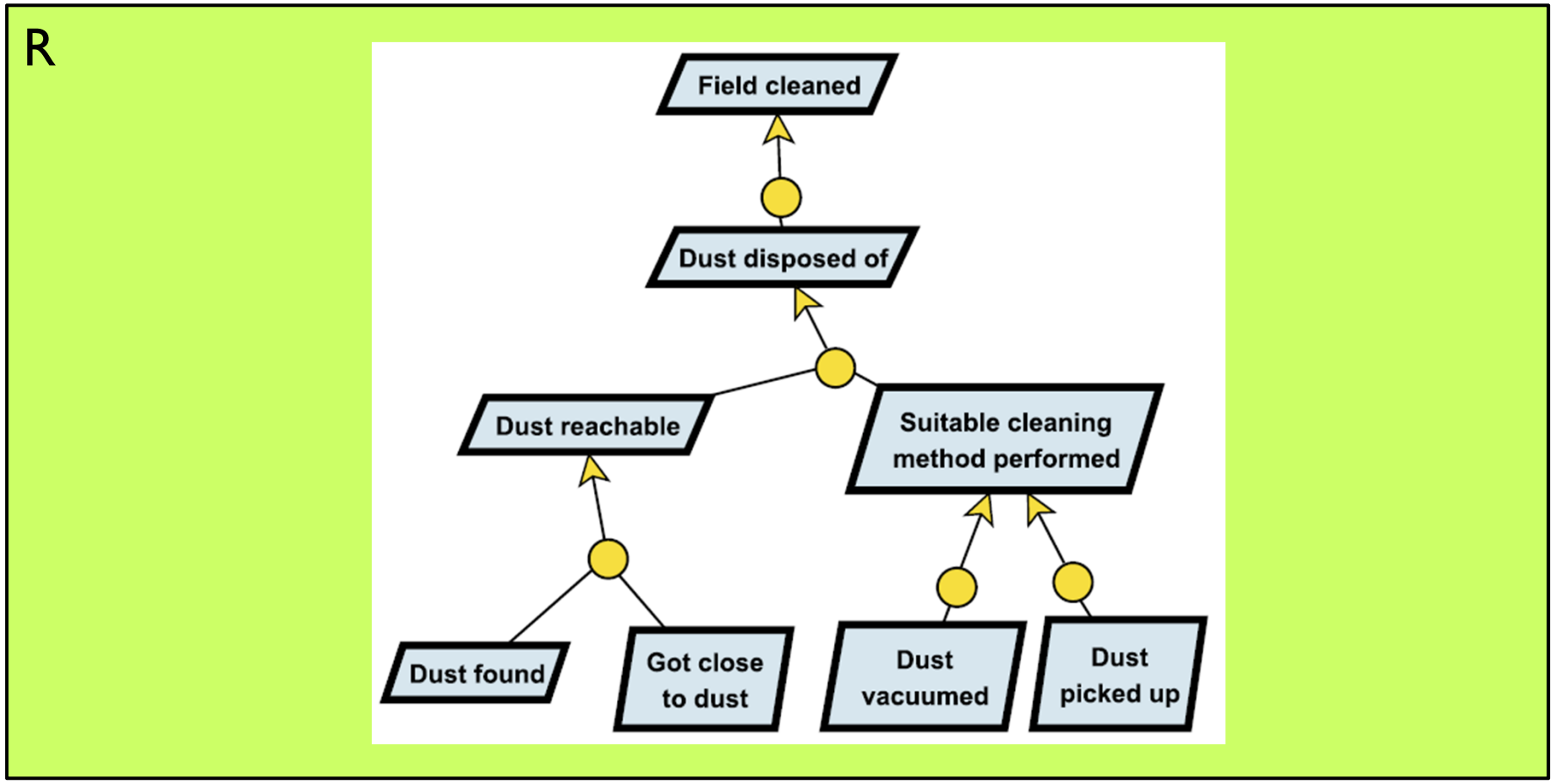
W'c

W'd

W'c = common(W'', W'), W'd = diff(W'', W')

R: requirements

- R: Initial goal model for a cleaning robot



R′: R changes to R′ (R → R′)

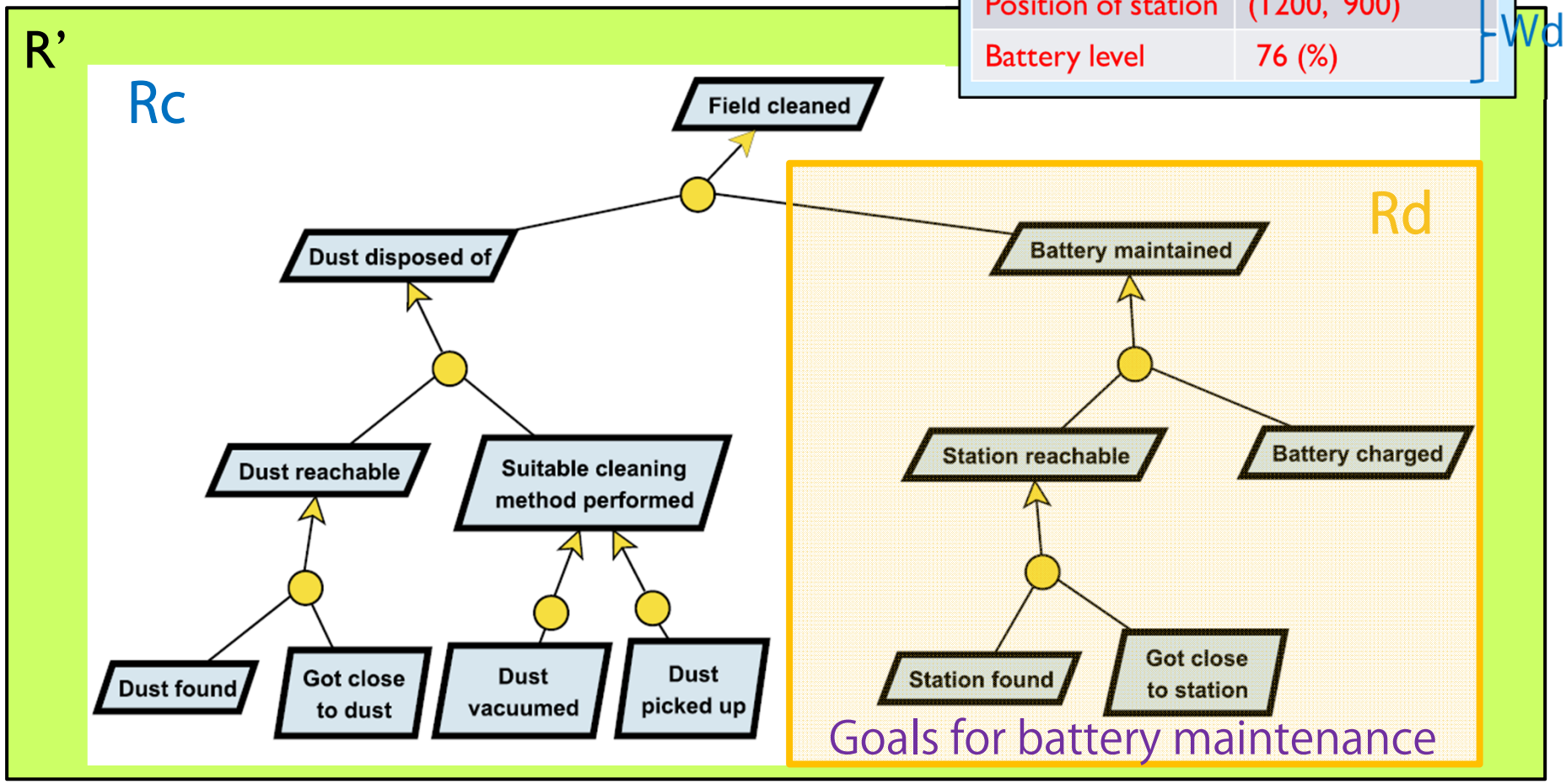
- R for battery maintenance is added

W′

Dust items	{((x1, y1), empty can), ((x2, y2), litter), ((x3, y3), house moss)}
Position of robot	(900, 600)
Position of station	(1200, 900)
Battery level	76 (%)

Wc

Wd



Rc = common(R′, R), Rd = diff(R′, R)

R'': R' changes to R'' (R' → R'')

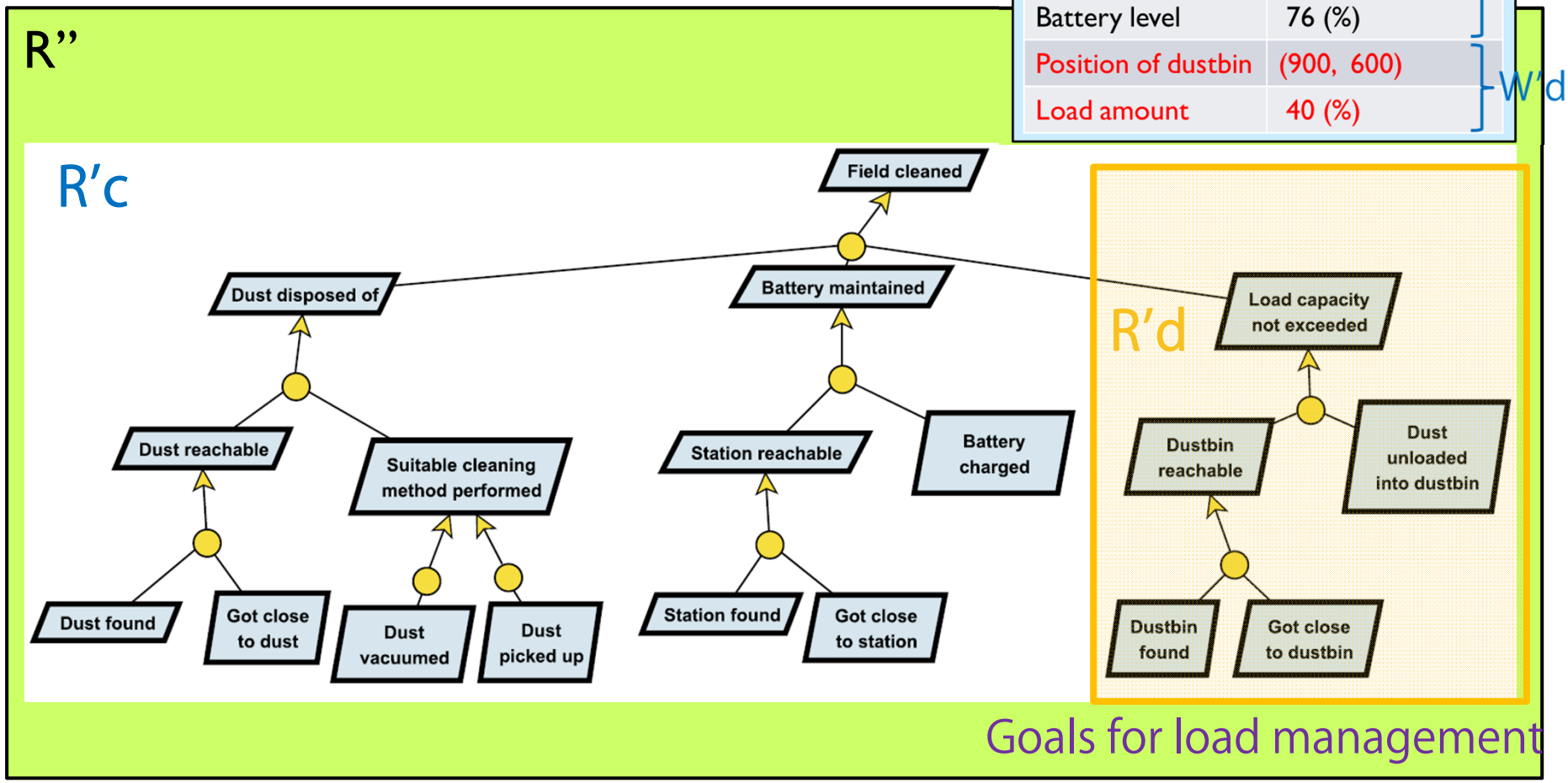
- R for load management is added

W''

Dust items	{((x1, y1), empty can), ((x2, y2), litter), ((x3, y3), house moss)}
Position of robot	(0, 1800)
Position of station	(1200, 900)
Battery level	76 (%)
Position of dustbin	(900, 600)
Load amount	40 (%)

W'c

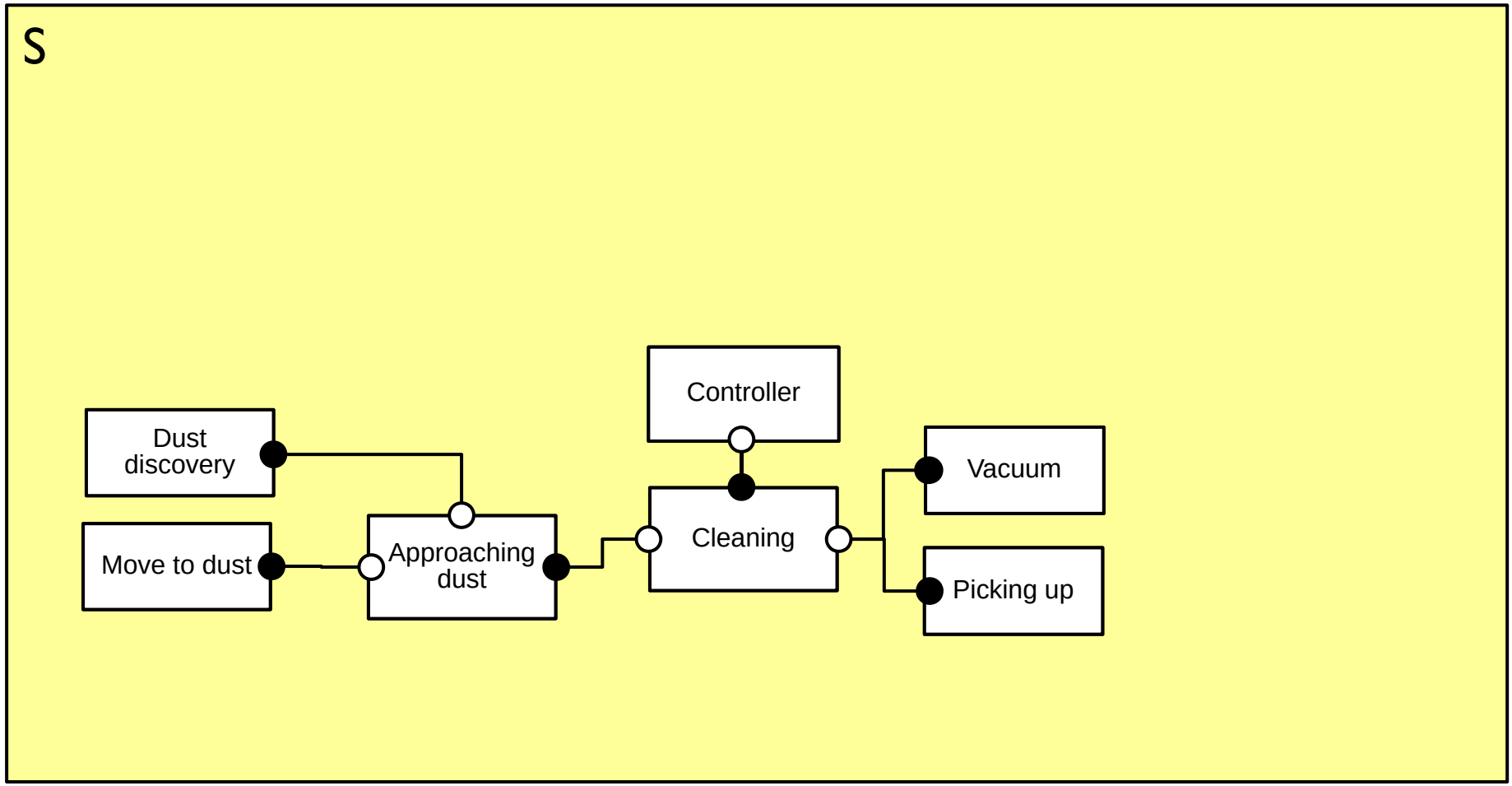
W'd



R'c = common(R'', R'), R'd = diff(R'', R')

S: specifications

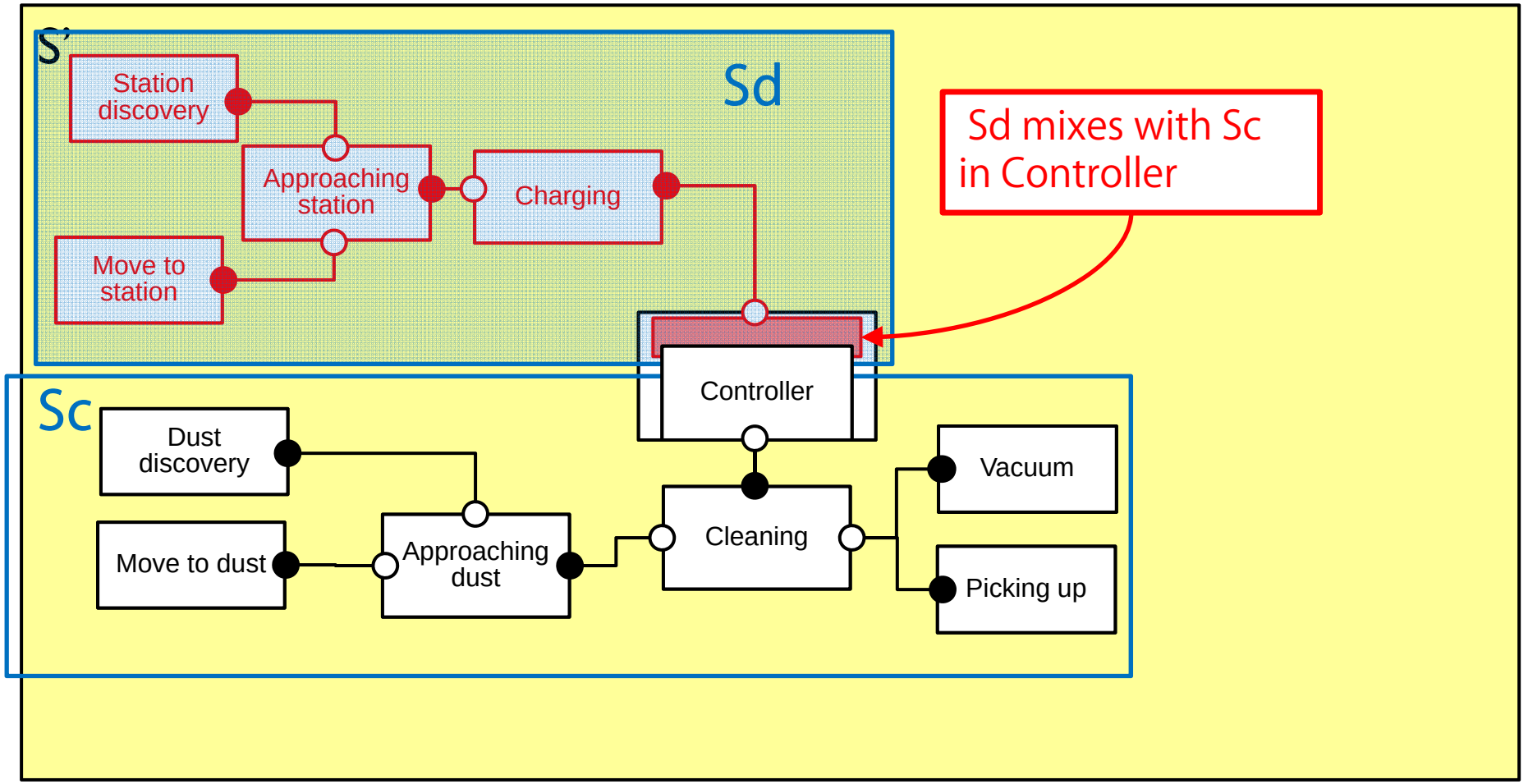
- S: Architectural configuration for cleaning dust items



Provided service port — Component — Required service port

S′: specifications should be changed (S → S′)
In a centralized control manner

- S′: Behaviors for battery maintenance are added

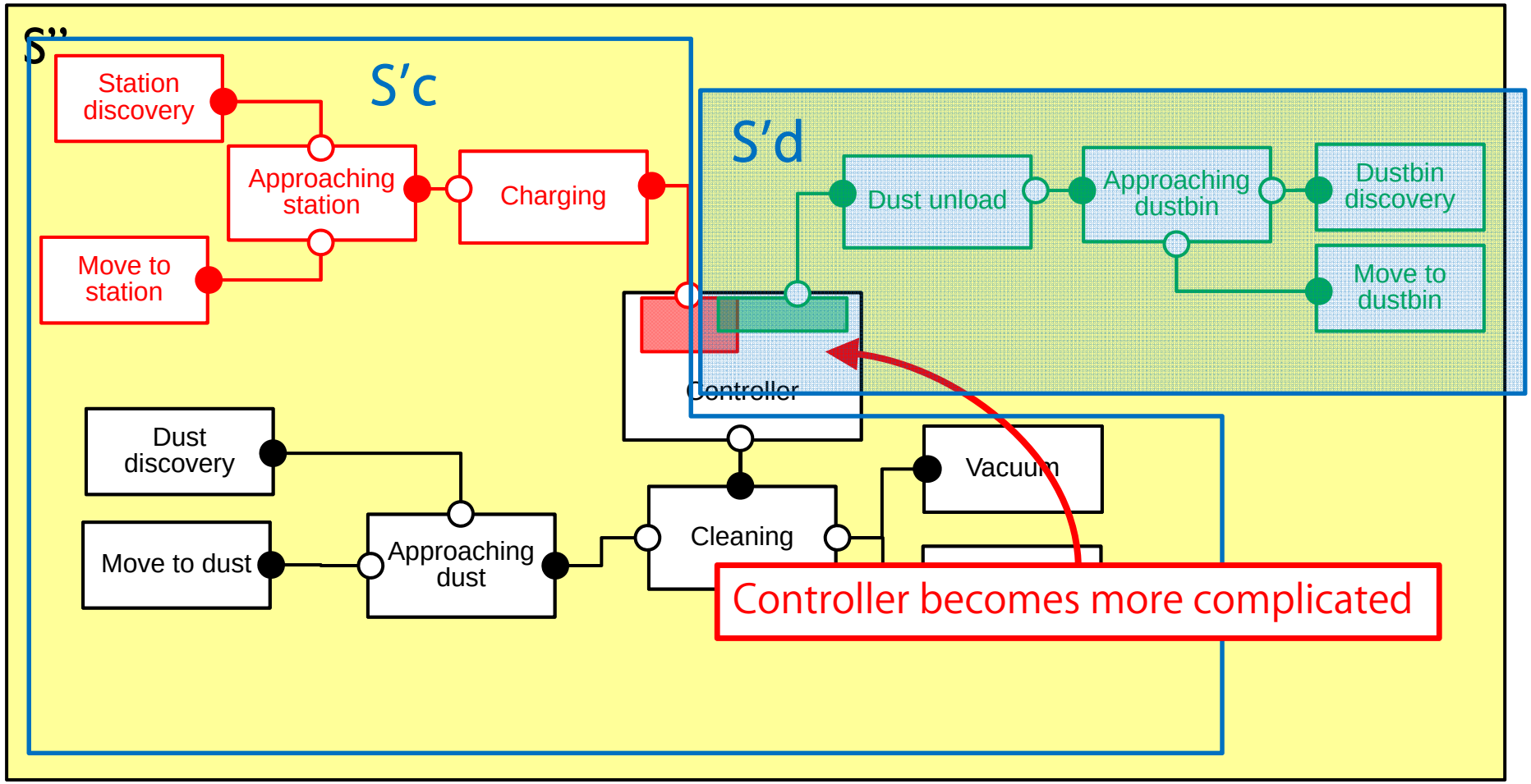


Provided service port — Component — Required service port

Sc = common(S′, S), Sd = diff(S′, S)

S″: specifications should be changed (S′ → S″)
In a centralized control manner

- S″: Behaviors for load management are added



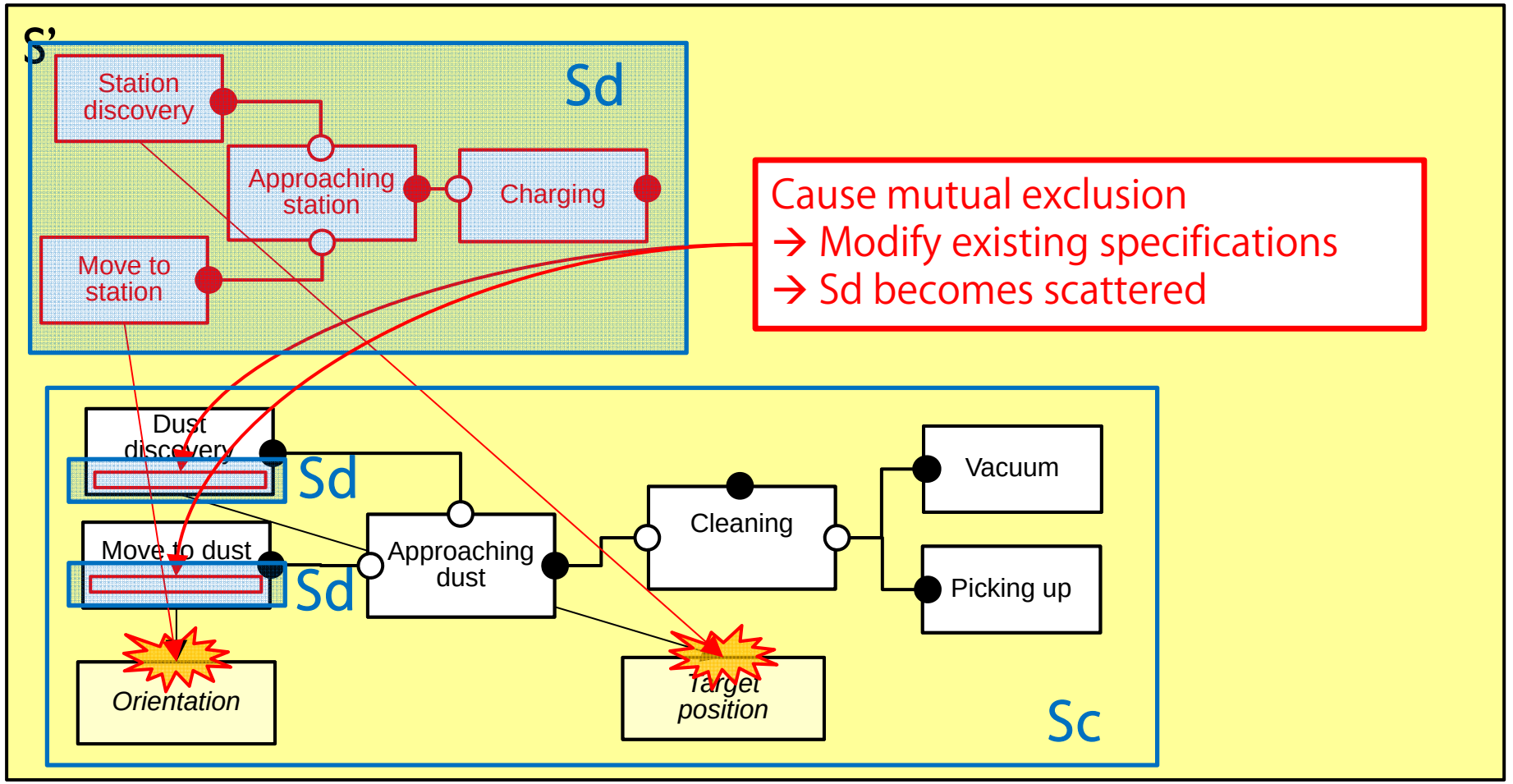
Provided service port | Component | Required service port

$S'c = \mathrm{common}(S'', S'),\ S'd = \mathrm{diff}(S'', S')$

S': specifications should be changed (S → S')

In a distributed control manner

- S': Behaviors for battery maintenance are added



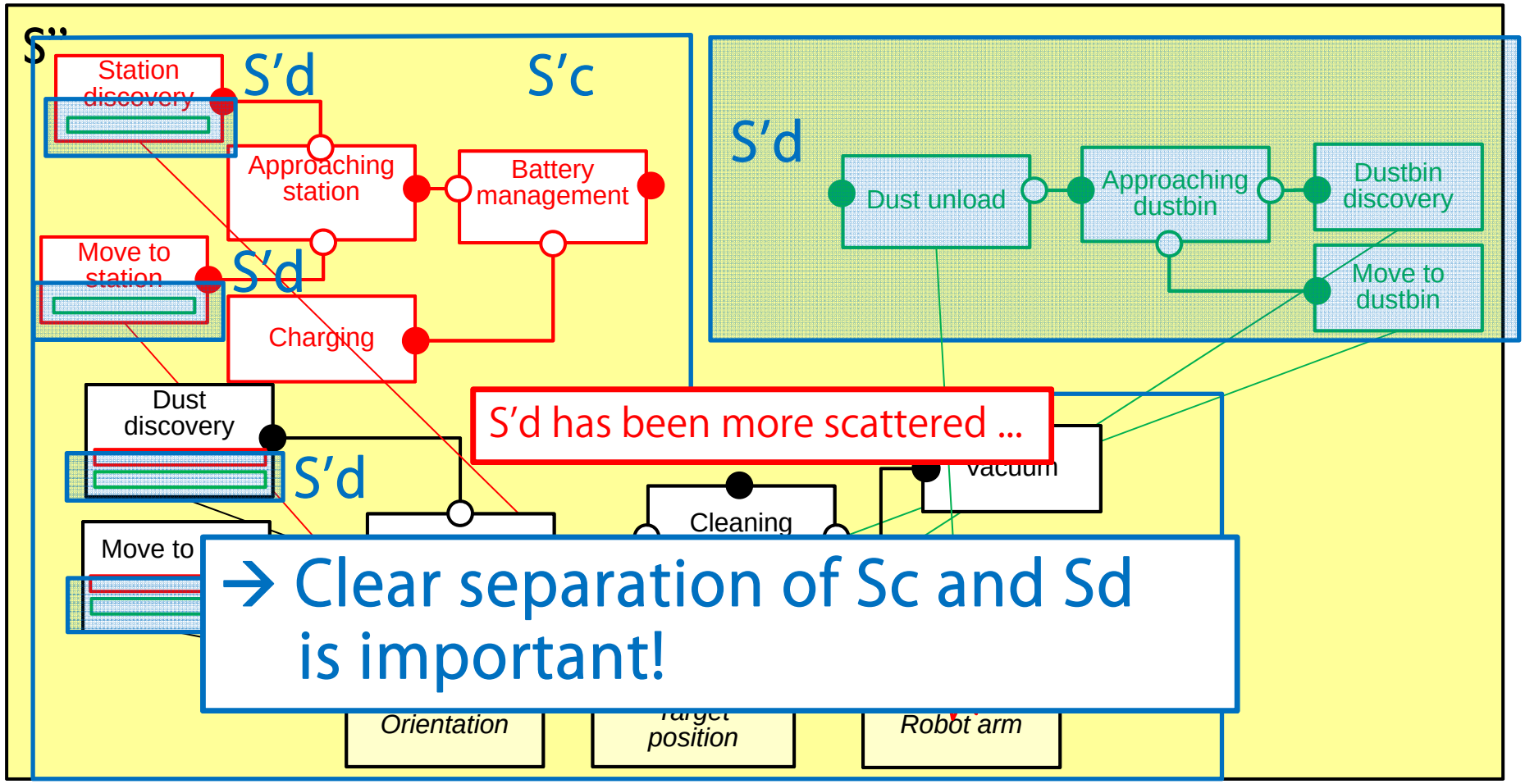
Provided service port | Component | Required service port

Global variable

S″: specifications should be changed (S′ → S″)

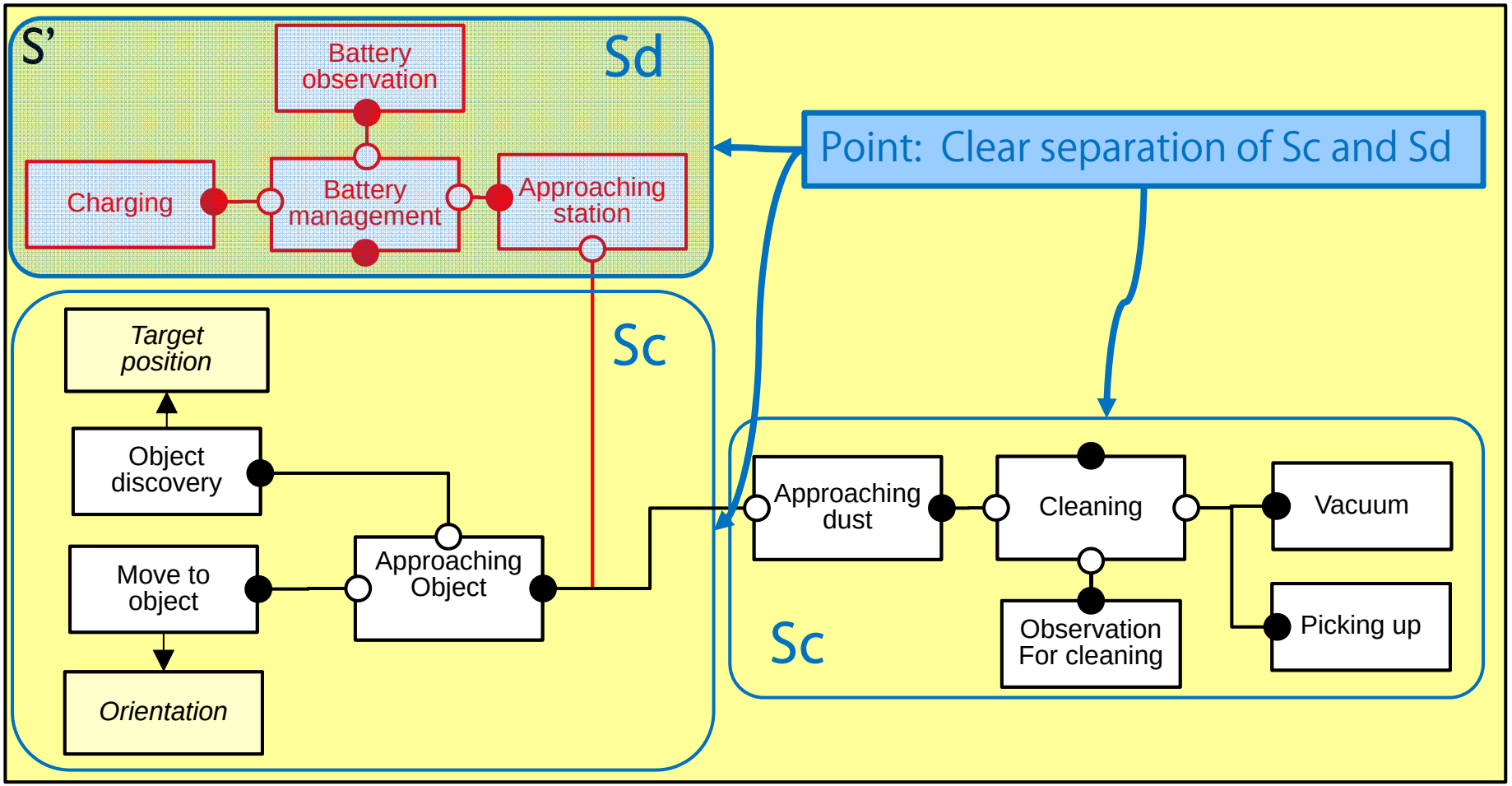
In a distributed control manner

- S″: Behaviors for load management are added



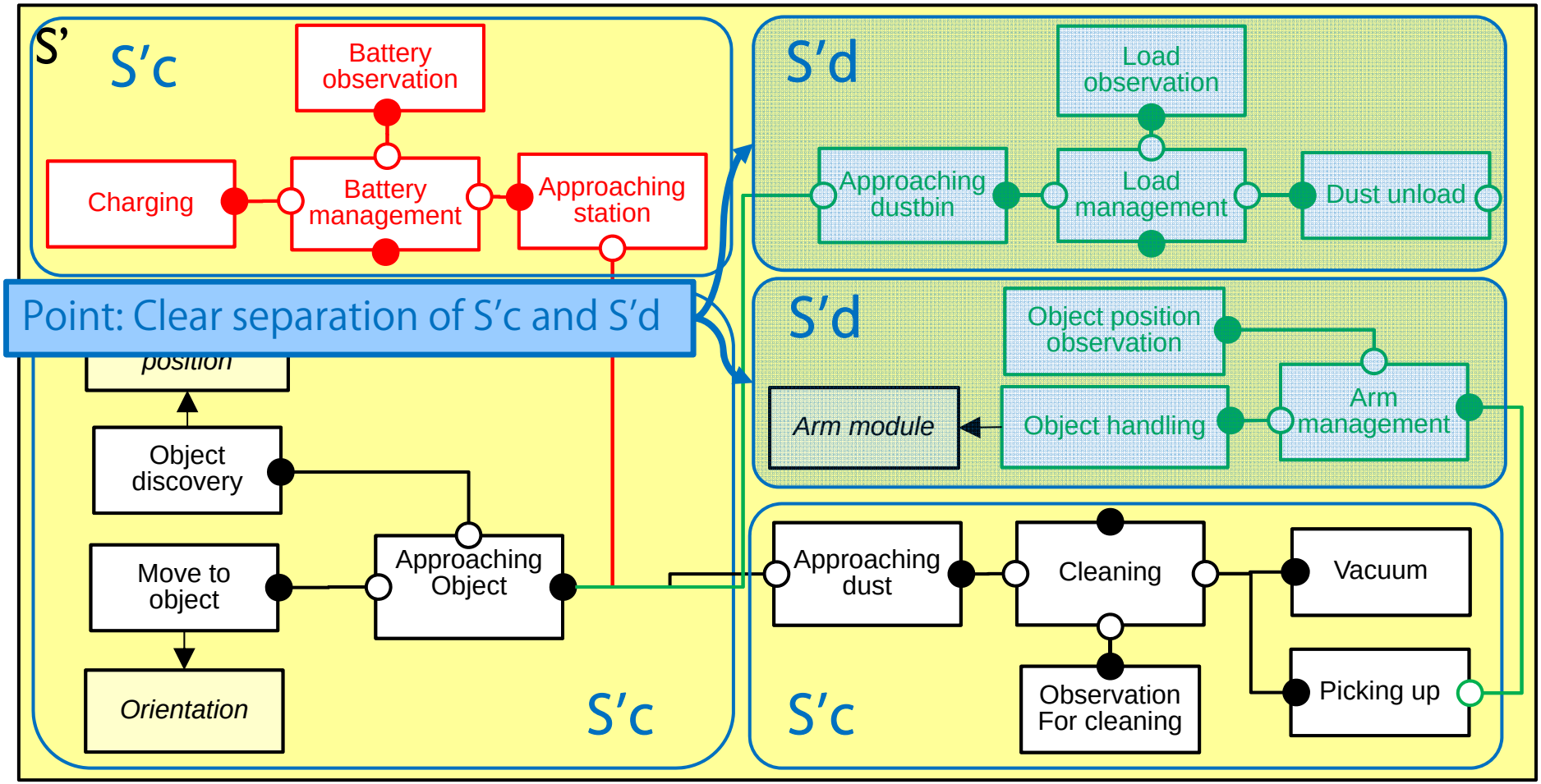
Desirable S'

- S': Behaviors for battery maintenance are added



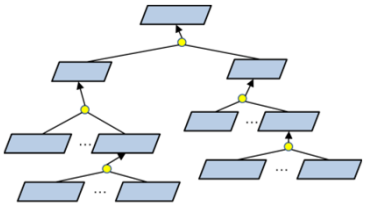
Desirable S''

- S'': Behaviors for load management are added



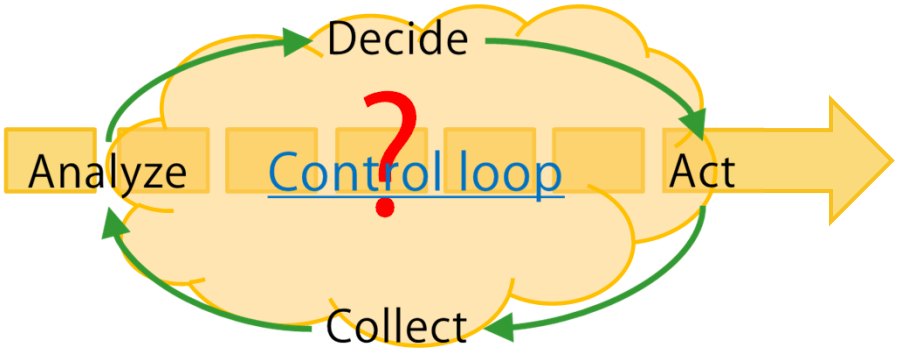
How do we construct desirable S' and S''?

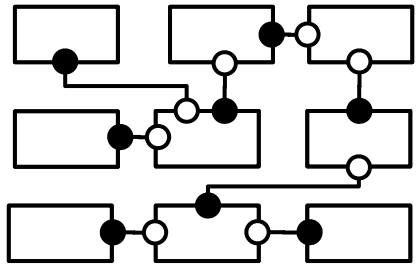
- Difficulty: Less traceability between R and S
  - S does not appear in the goal model
- Approach:
  - Design S in R → Introduce behavioral model for S
  - Clearly separate Sc and Sd → Independent behavioral modules



Goal model

R



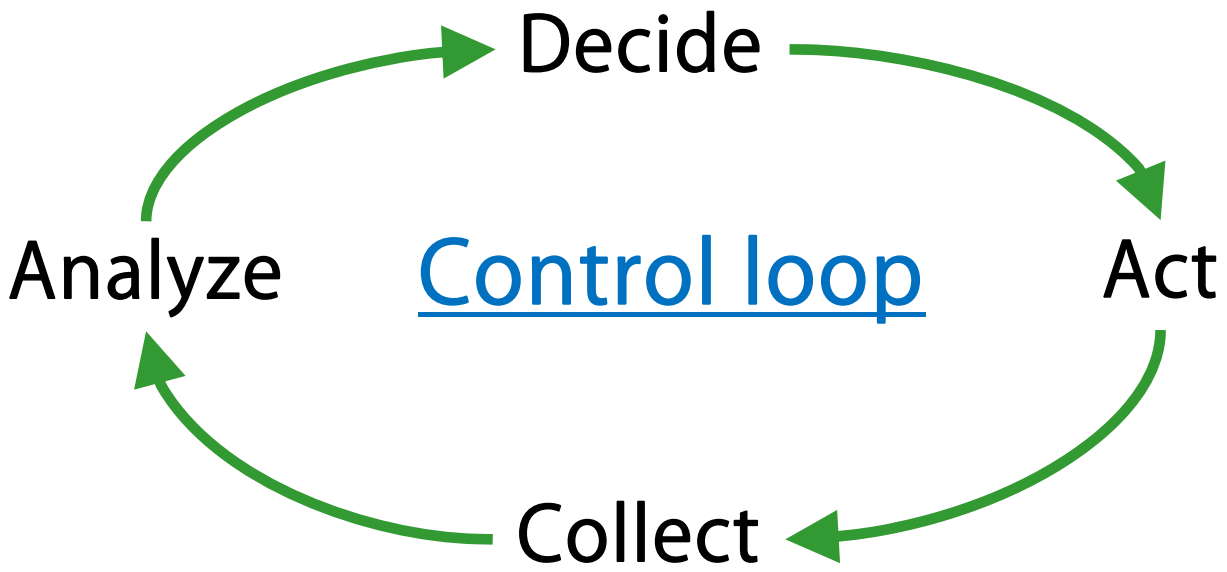


System architecture

Desirable S

Control loop

- Control loop [Shaw 95] [Dobson 06] :
  - Behavioral model that focuses on process control
    - Summarized as Collect, Analyze, Decide, Act
  - Process variables: Input, controlled, and manipulated variables

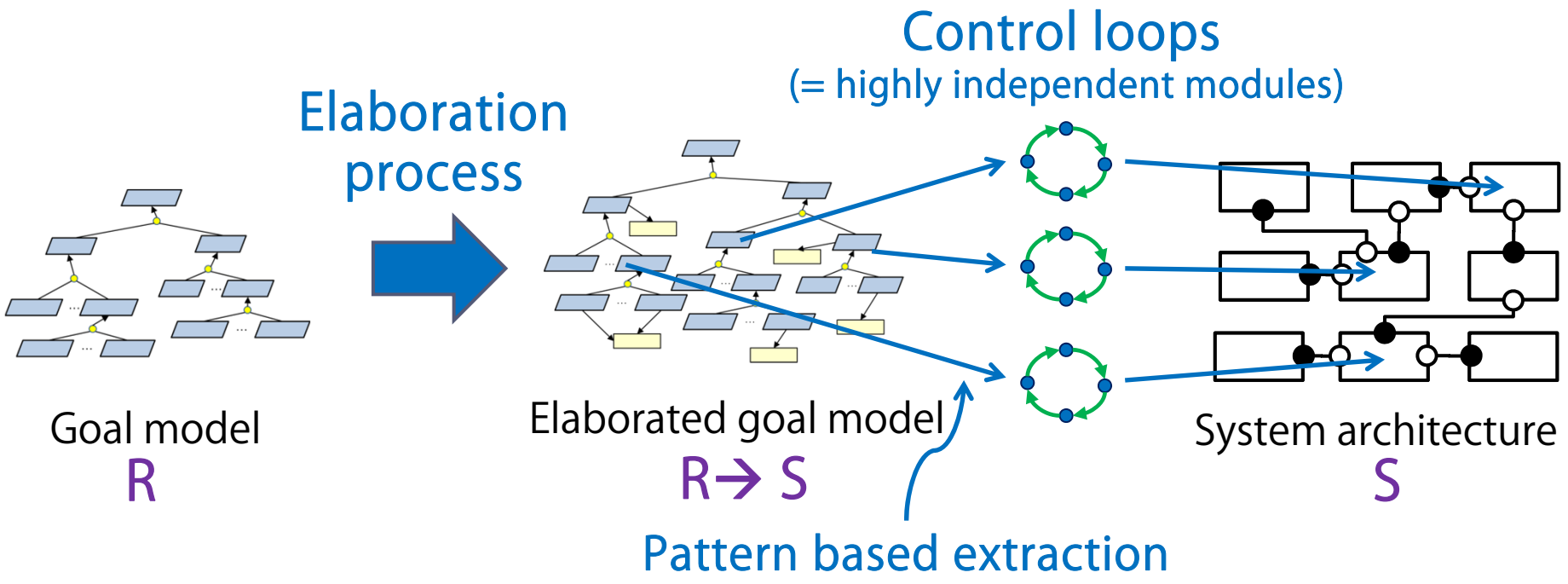


[Shaw 95] Mary Shaw, "Beyond Objects: A Software Design Paradigm Based on Process Control", ACM SIGSOFT Software Engineering Notes Homepage archive Volume 20 Issue 1, Jan. 1995

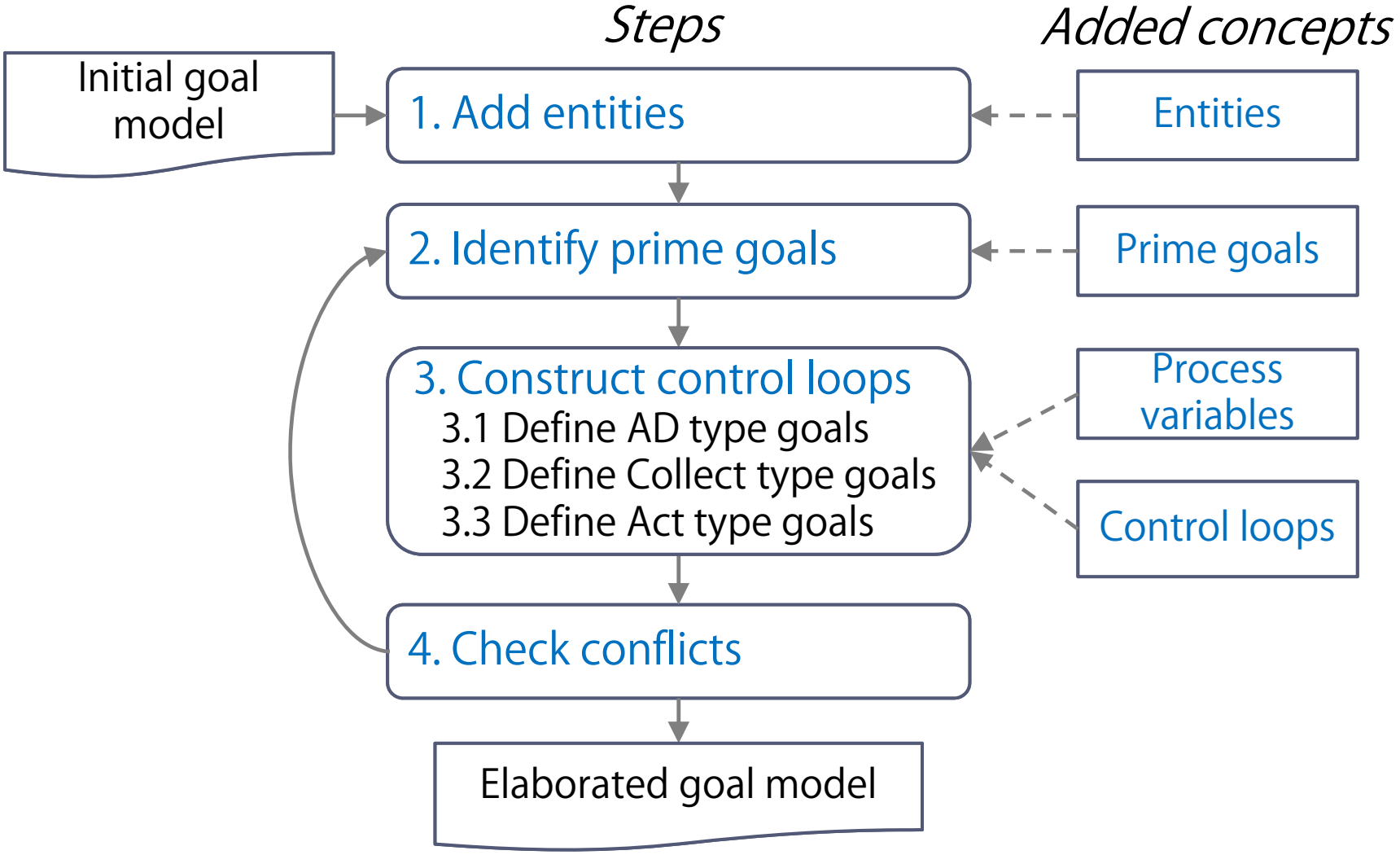
[Dobson 06] Dobson et. al, A survey of autonomic communications", ACM Transactions on Autonomous and Adaptive Systems, Vol. 1, Issue 2, 2006.

Our approach to constructing desirable S′ and S″

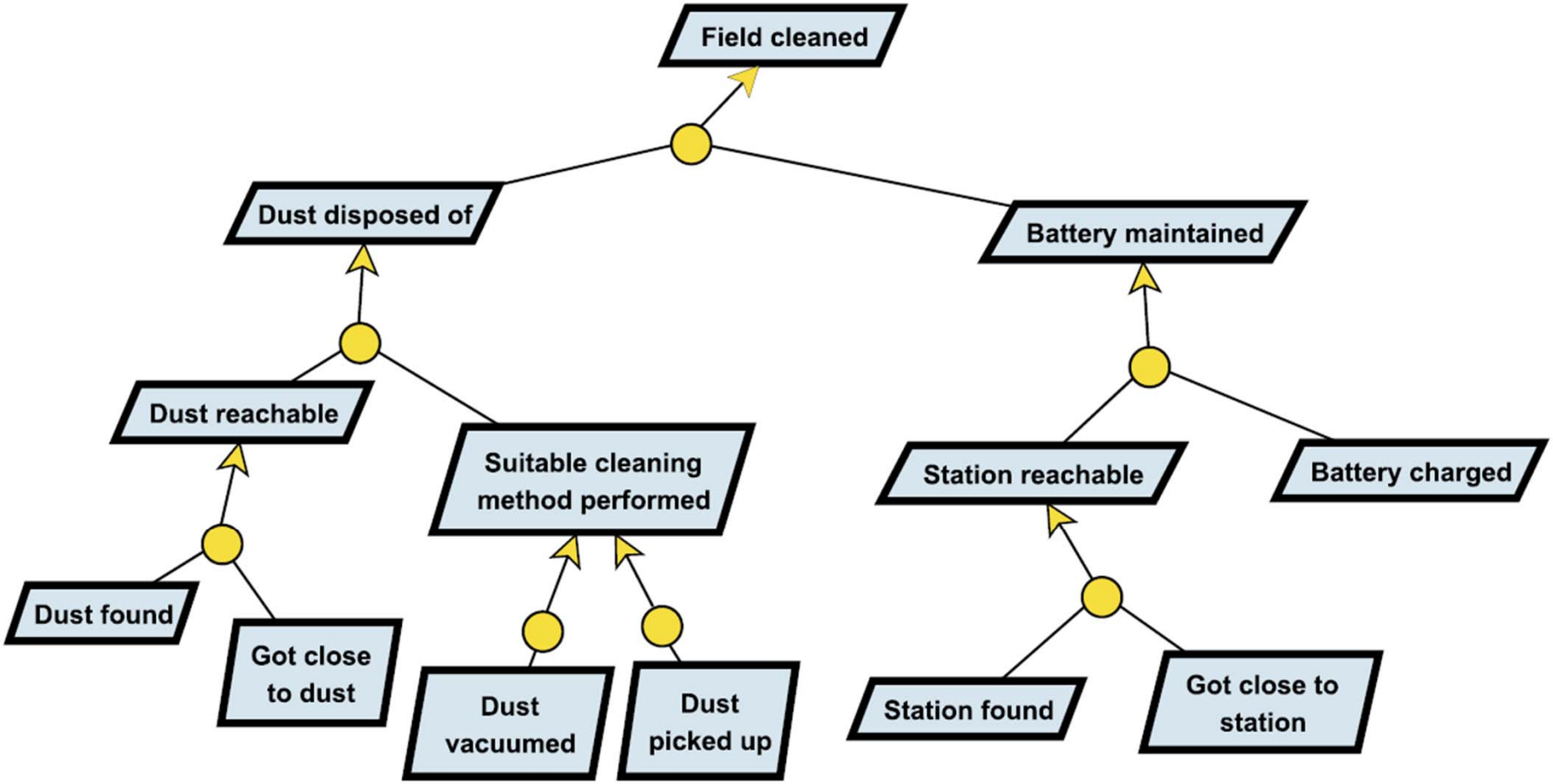
- Key points:
  - Design S based on control loops
  - Introduction of an elaboration process
    - Constructing control loops according to a description pattern



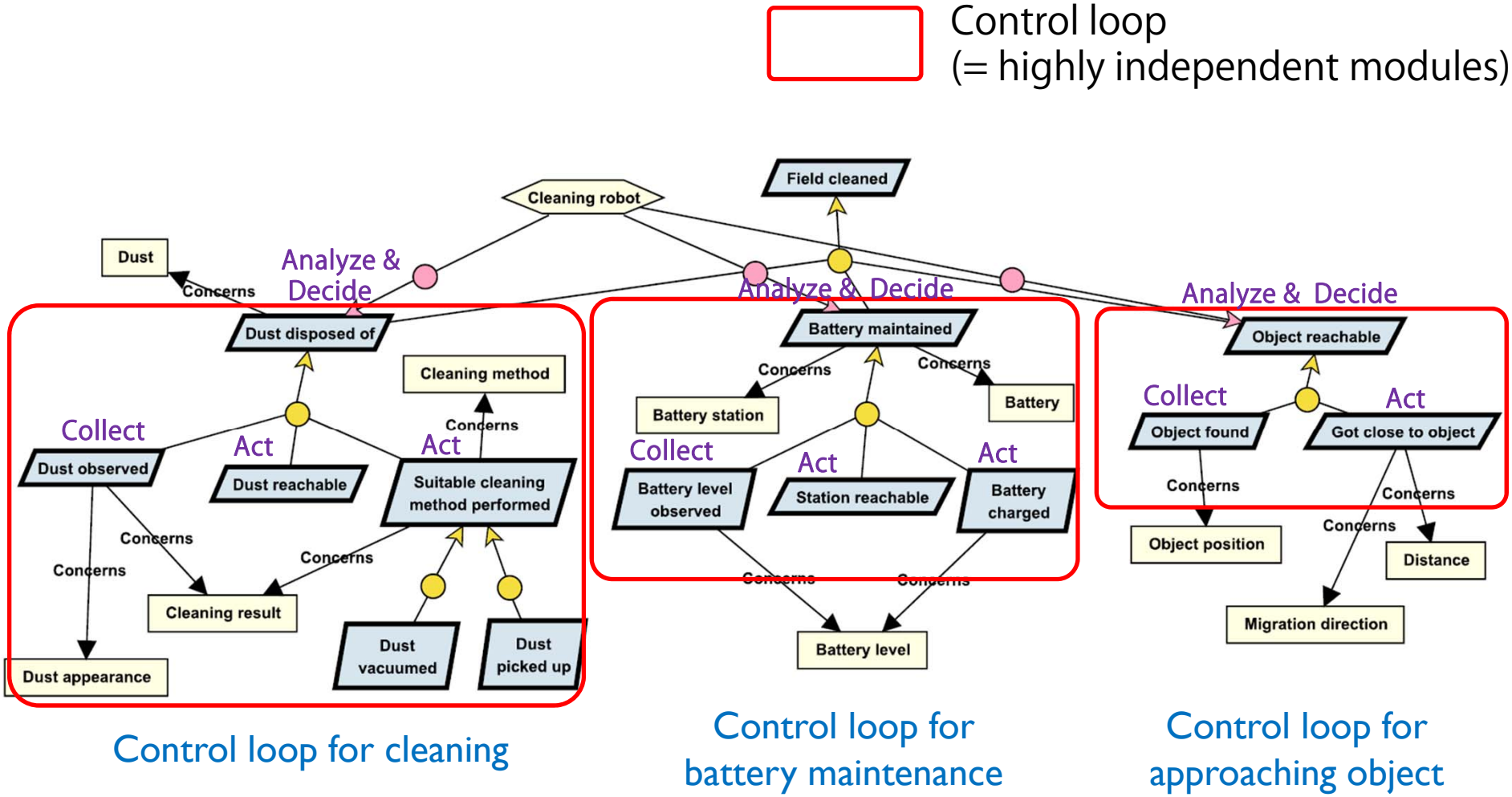
Elaboration process



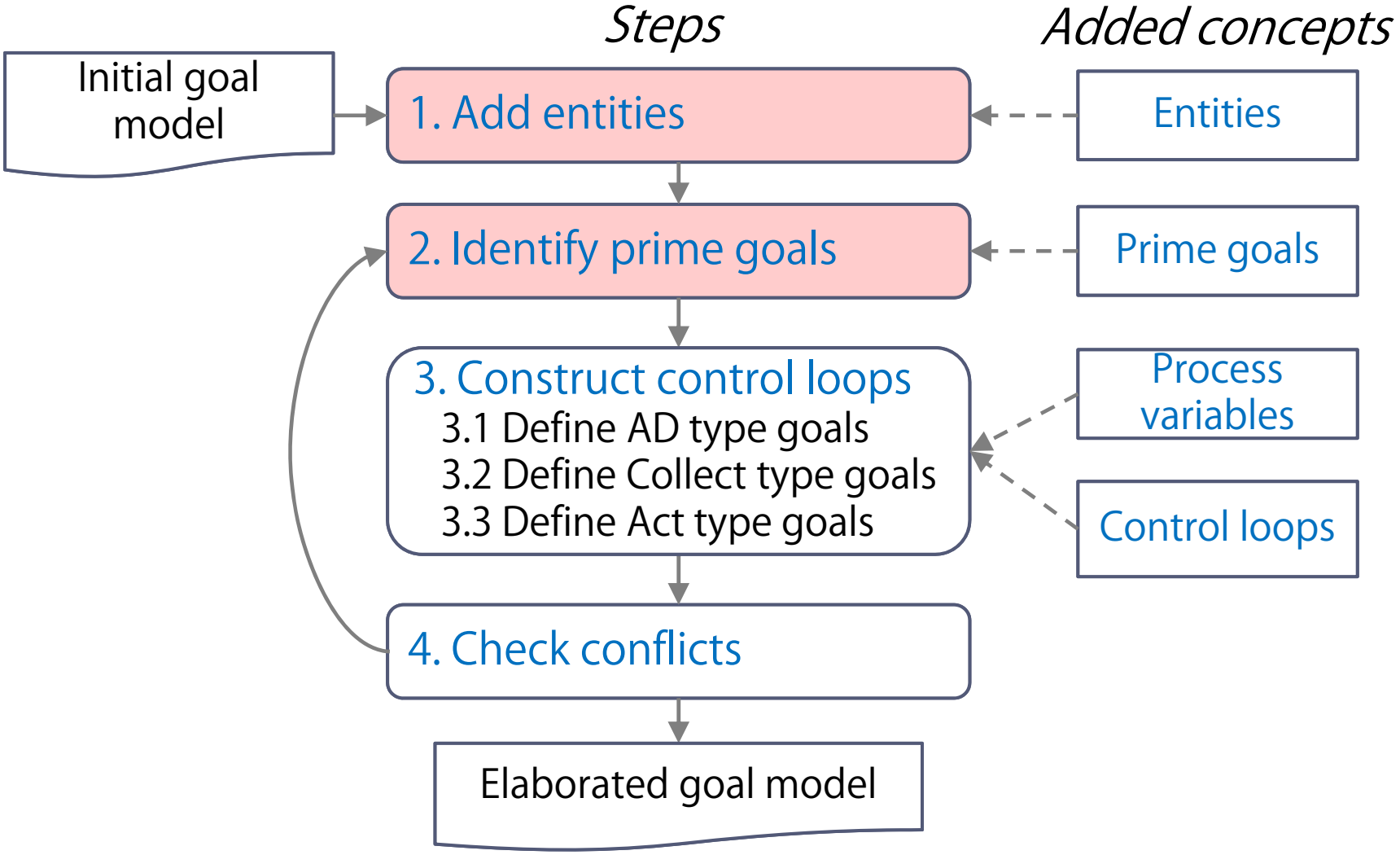
Goal model before elaboration



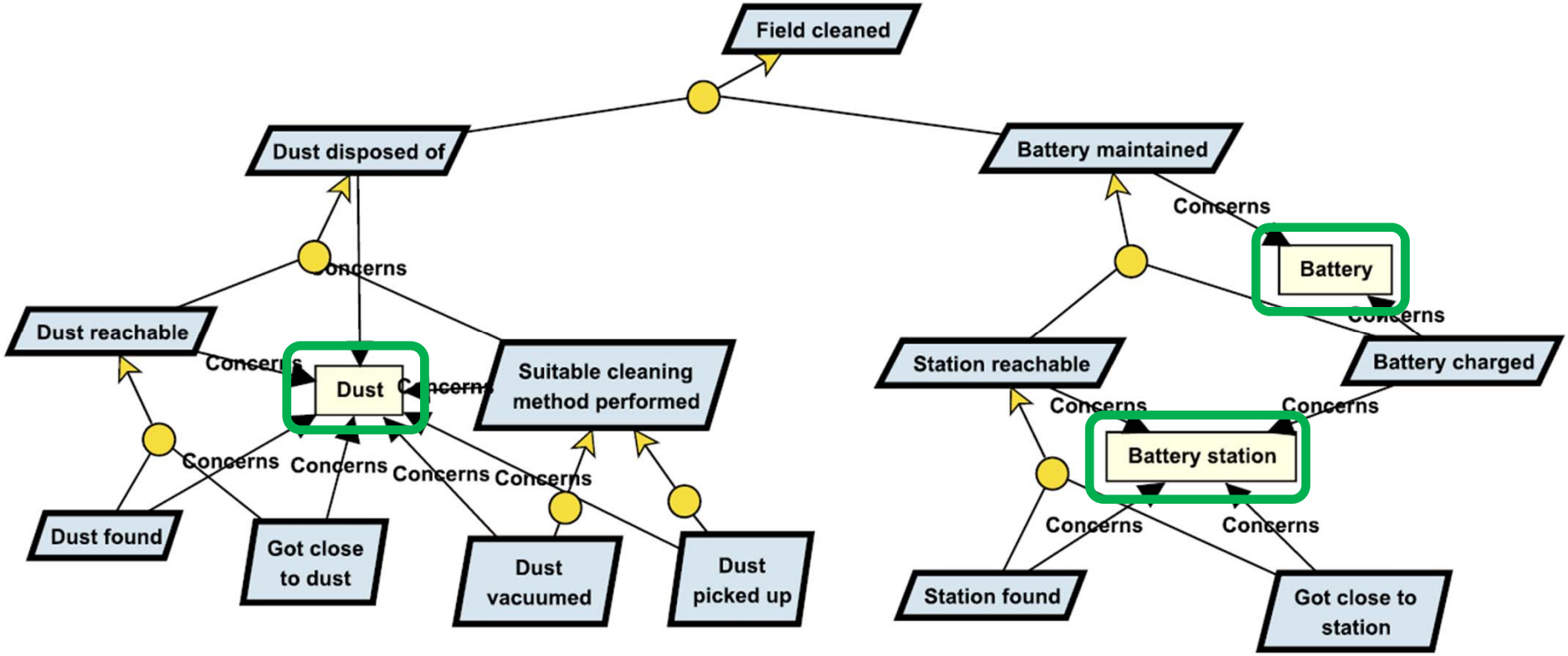
Goal model after elaboration



Elaboration process



Goal model for cleaning robot
1. Add entities



Entity

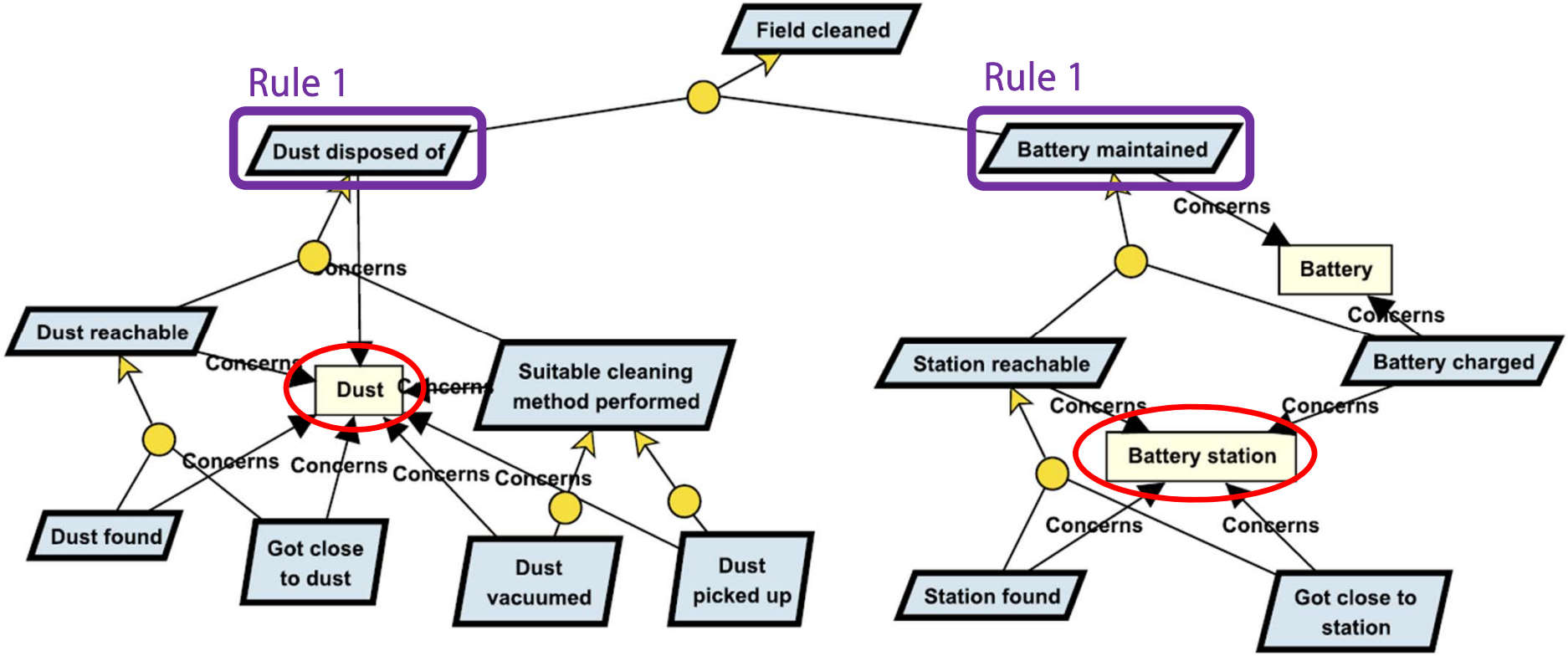
- Add relevant entities (objects) with concerns links
  Concerns links: relation between goals and entities

2. Identify prime goals

- Activities: determine prime goals by following guidelines
  - Prime goals: key goals for constructing highly independent modules
  - Individual control loops are assigned to prime goals
    - To independently design components achieving prime goals
    - To localize accesses of individual entities

- Guidelines: a goal g_i that satisfies one of the following rules is defined as a prime goal candidate
  - Rule 1) More than one child goal of g_i has **concerns links to the same entities**
  - Rule 2) Other goals depend on g_i through **Uses** labels

Goal model for cleaning robot
2. Identify prime goals (Extract candidates)



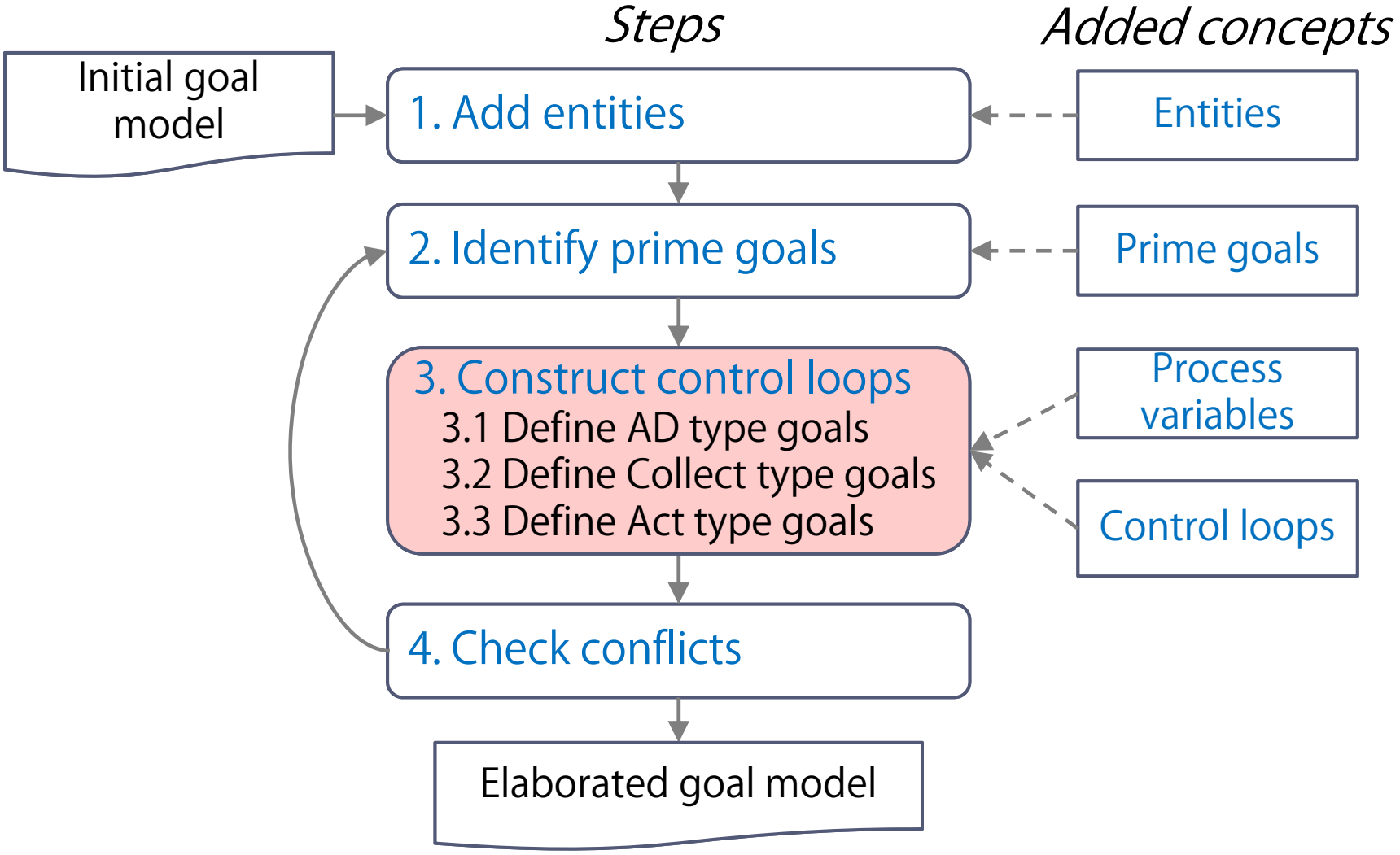
Prime goal (candidate)

Reason

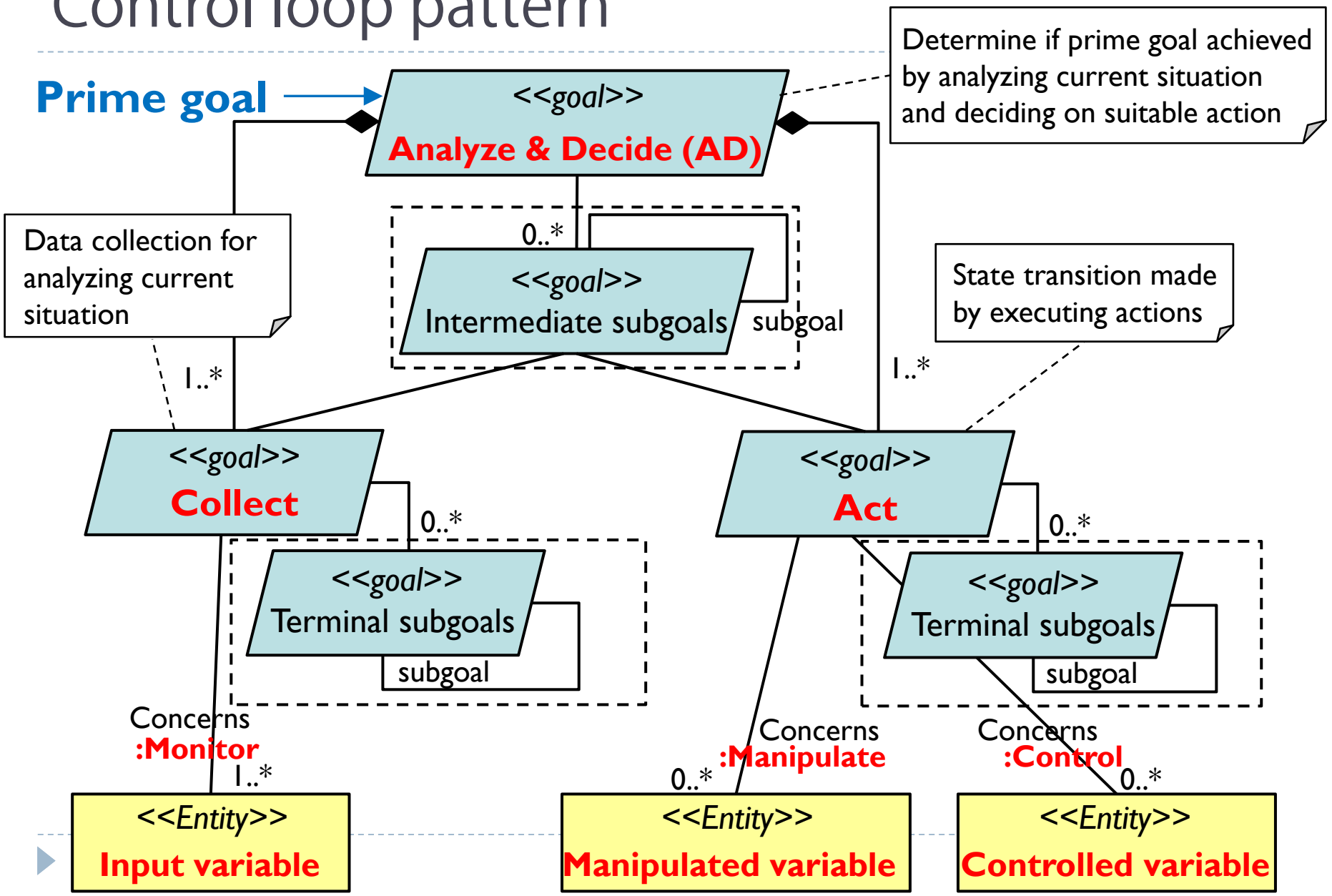
- Identify prime goals by applying Rule1

Rule1) More than one child goal of g_i has concerns links to the same entities

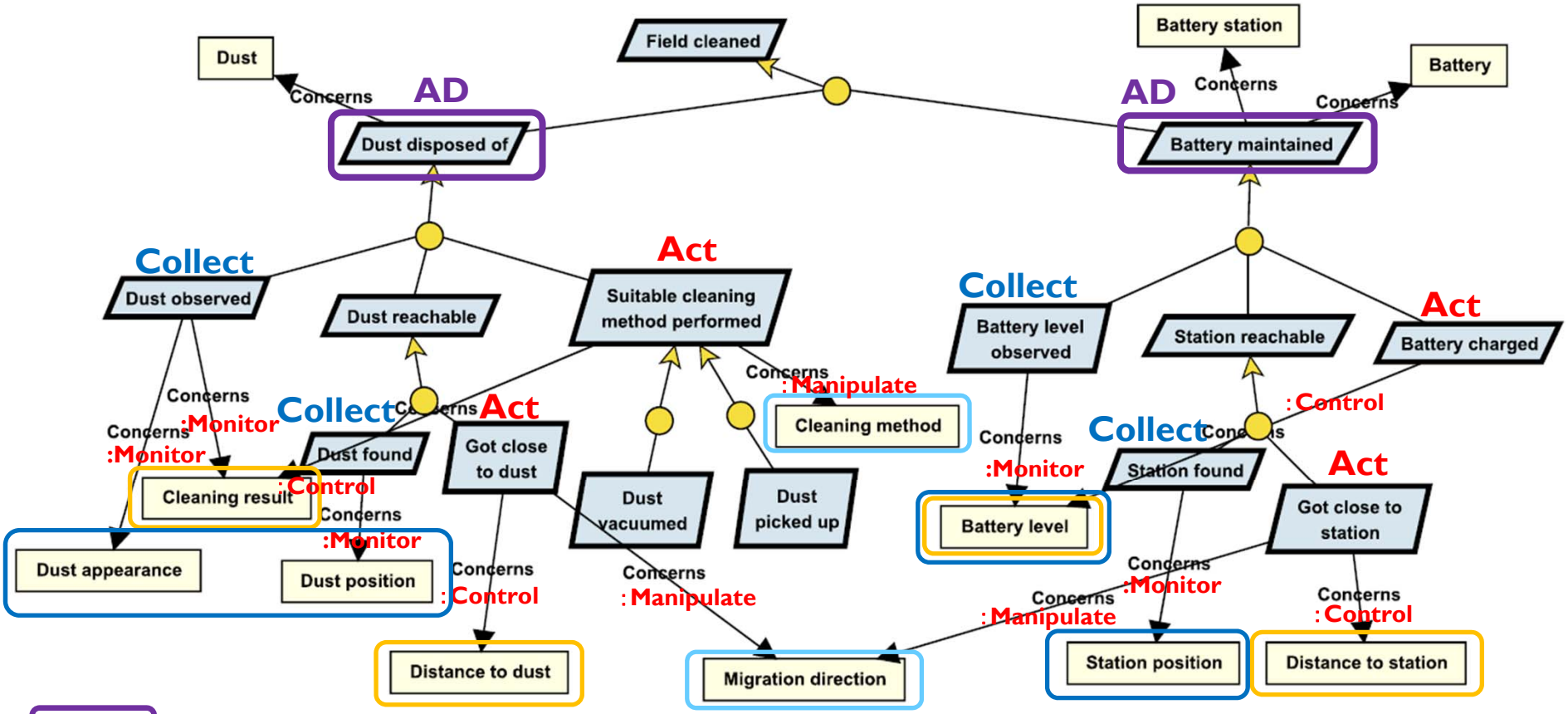
Elaboration process



Control loop pattern

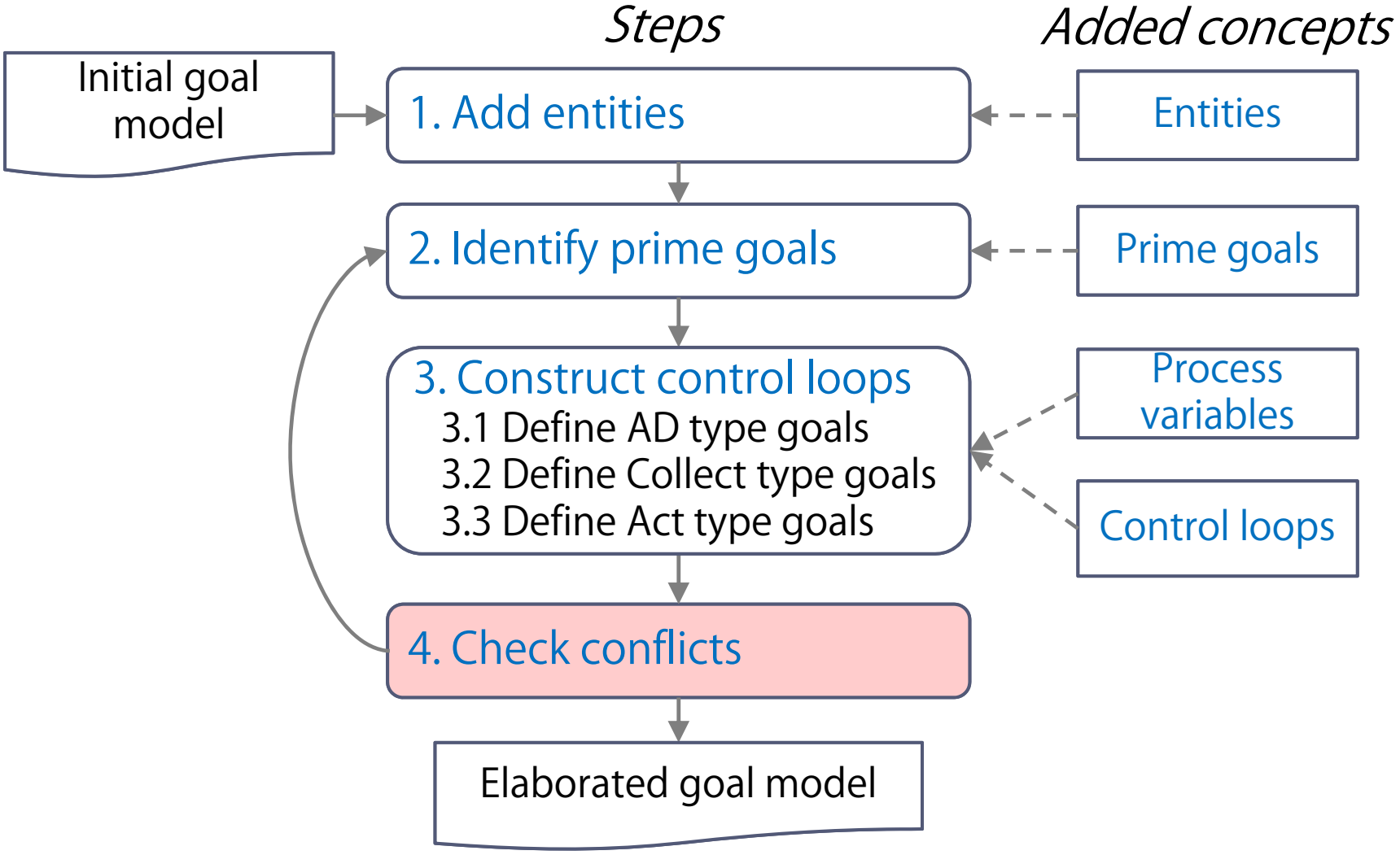


Goal model for cleaning robot
3. Control loop construction



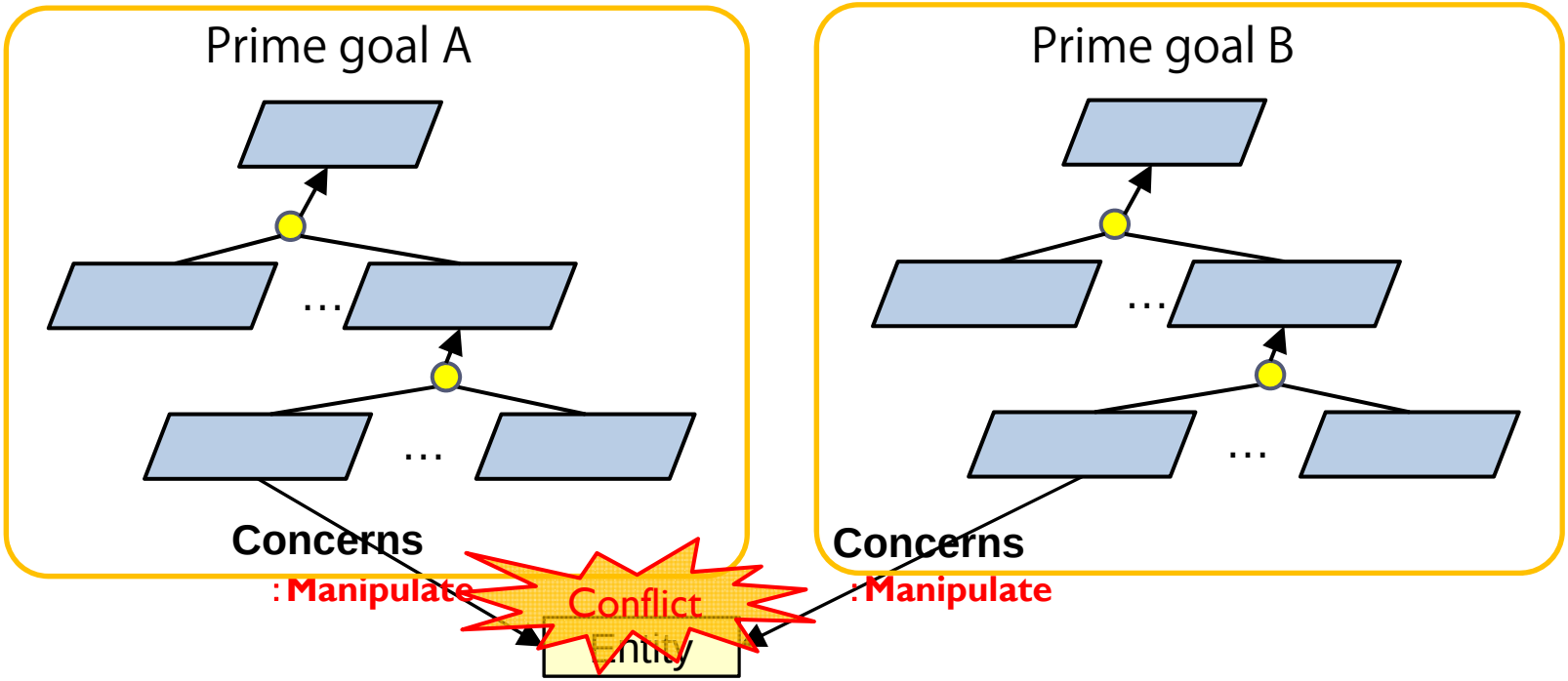
- Identify Act type goals
- Identify controlled and manipulated
  - Add them if not described in the goal model

Elaboration process



Conflict detection

- Embedding multiple control loops may cause conflicts
- Check concerns links to manipulated variables from multiple control loops

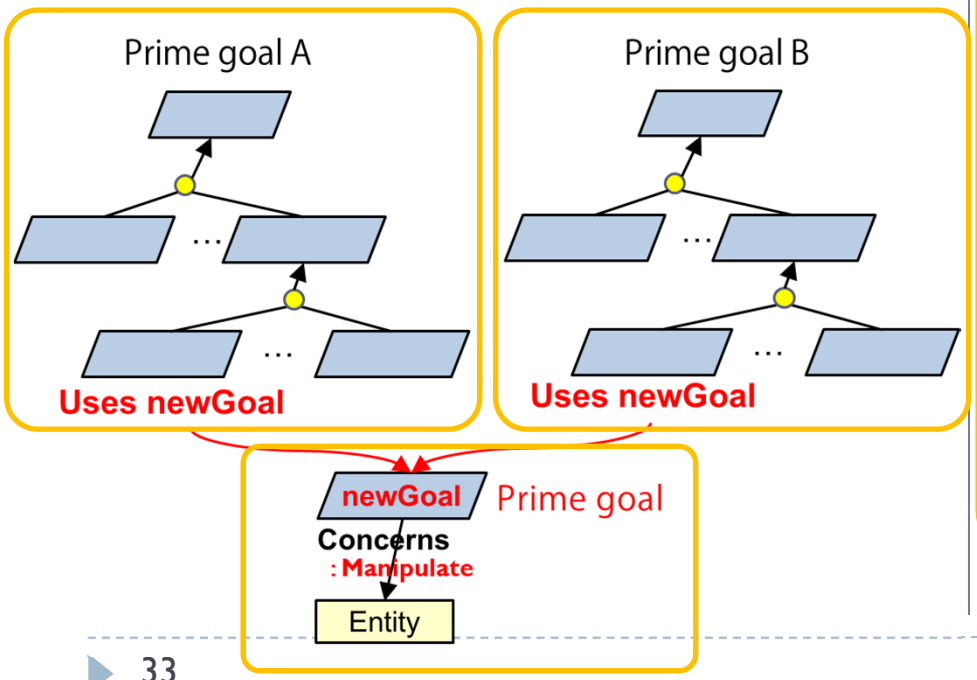


- Solve conflicts by modifying goal model structure

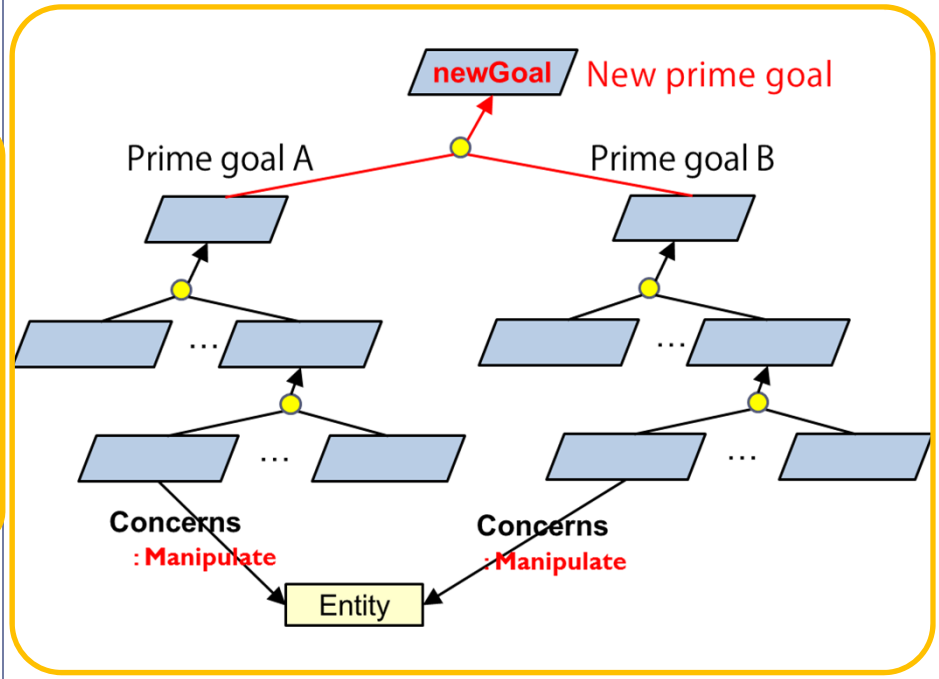
Solution

- Solution 1: Link aggregation
  - Aggregate concerns links into a new prime goal
  - Uses: dependencies on other prime goal
    - Goal A needs prime goal B to achieve it

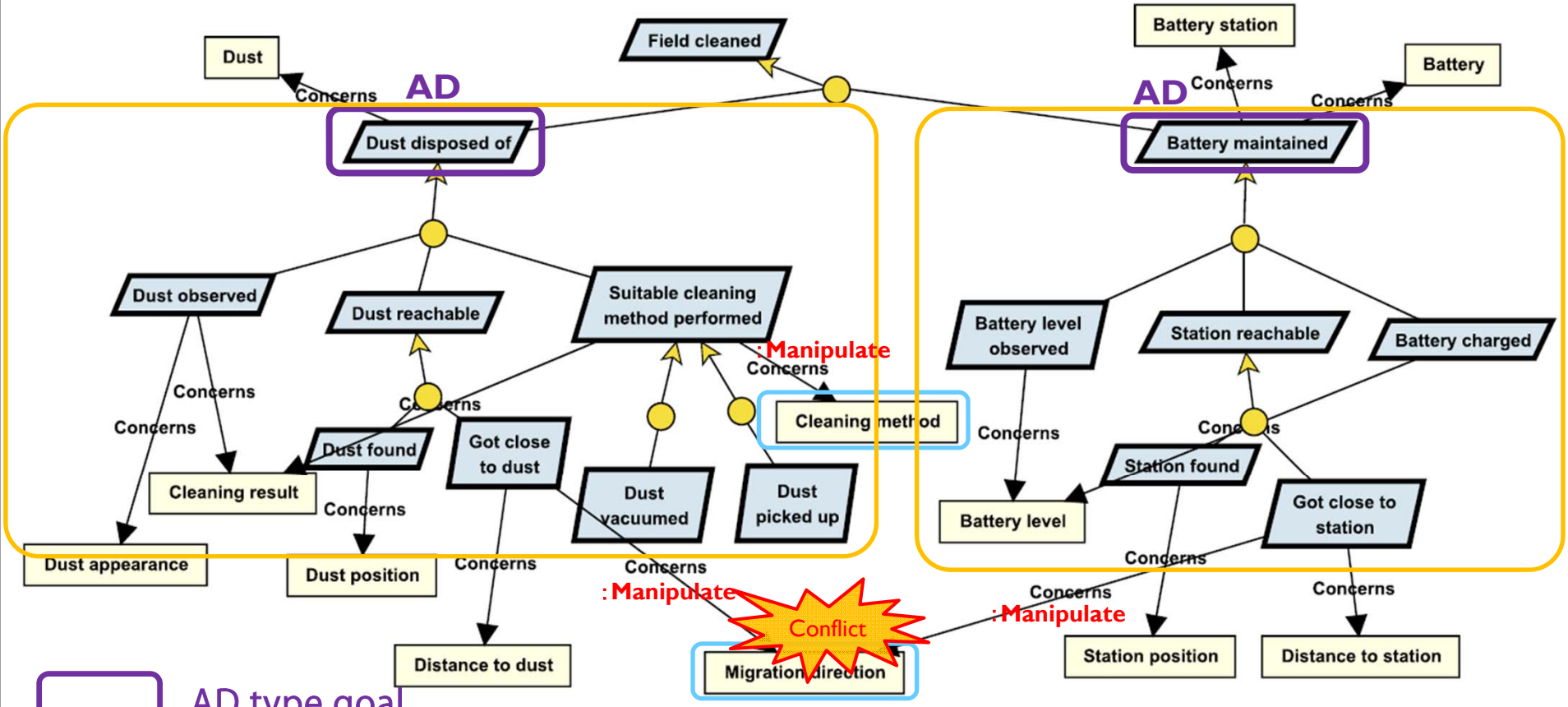
    → Label "Uses B" on goal A



- Solution 2: Goal aggregation
  - Aggregate conflicting prime goals into a tree
  - By adding a new root goal, which becomes a new prime goal



Goal model for cleaning robot
4. Conflict checking



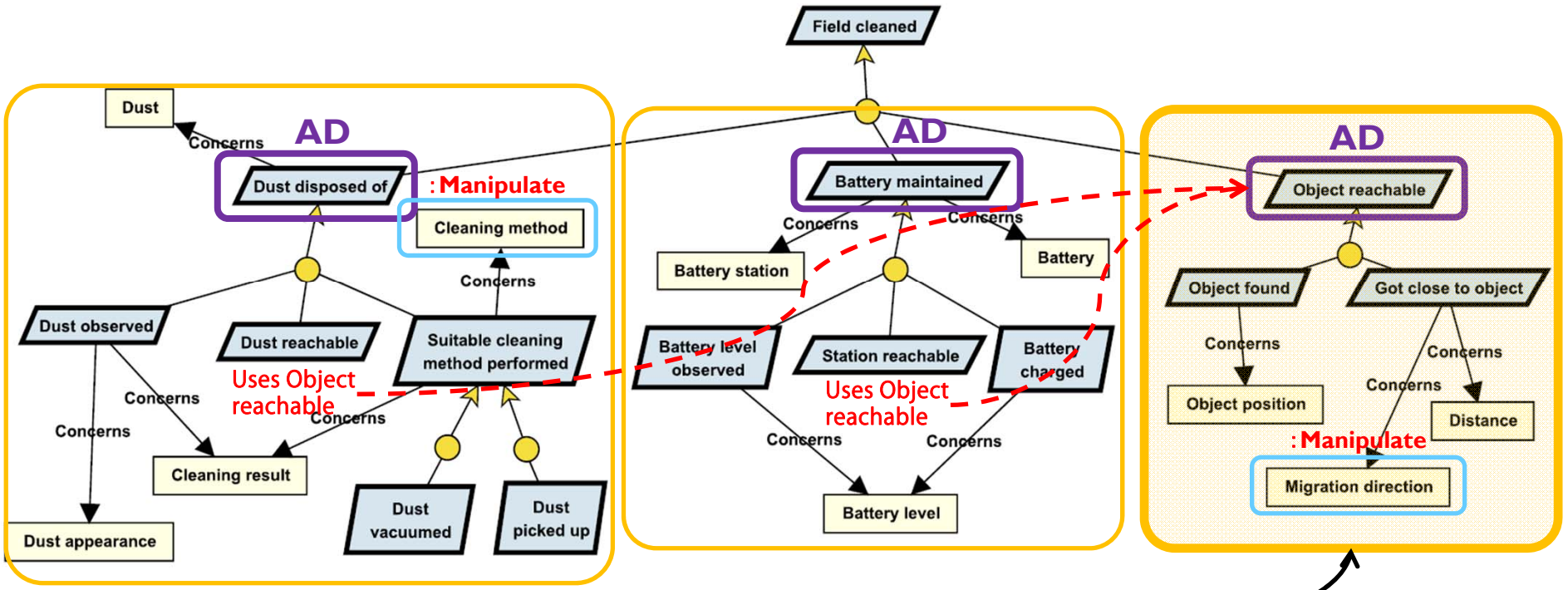
AD type goal

Control loop

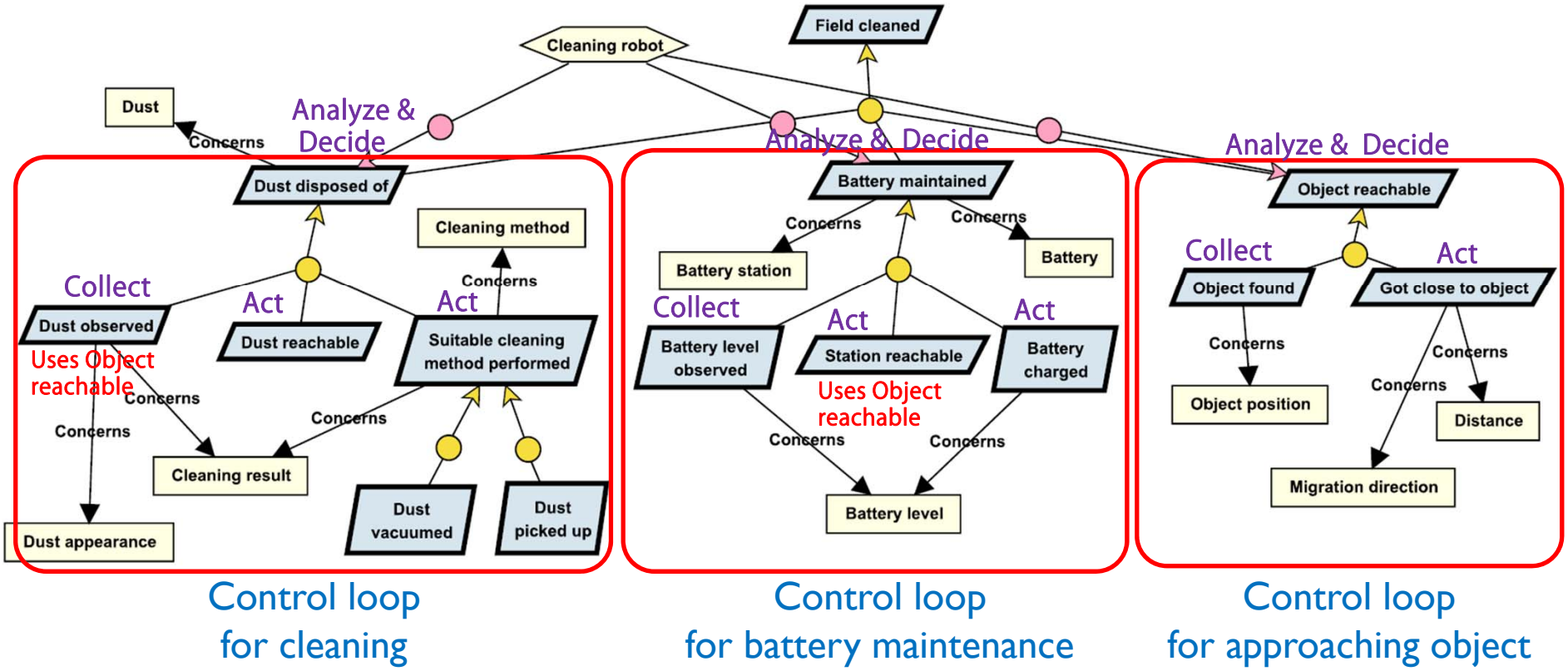
Manipulated variable

- Check conflicts
→ Detect conflict on entity "Migration direction"
→ Apply Solution 1 (Link aggregation)

Goal model for cleaning robot
4. Conflict checking (Resolve conflicts)



Elaborated goal model for cleaning robot

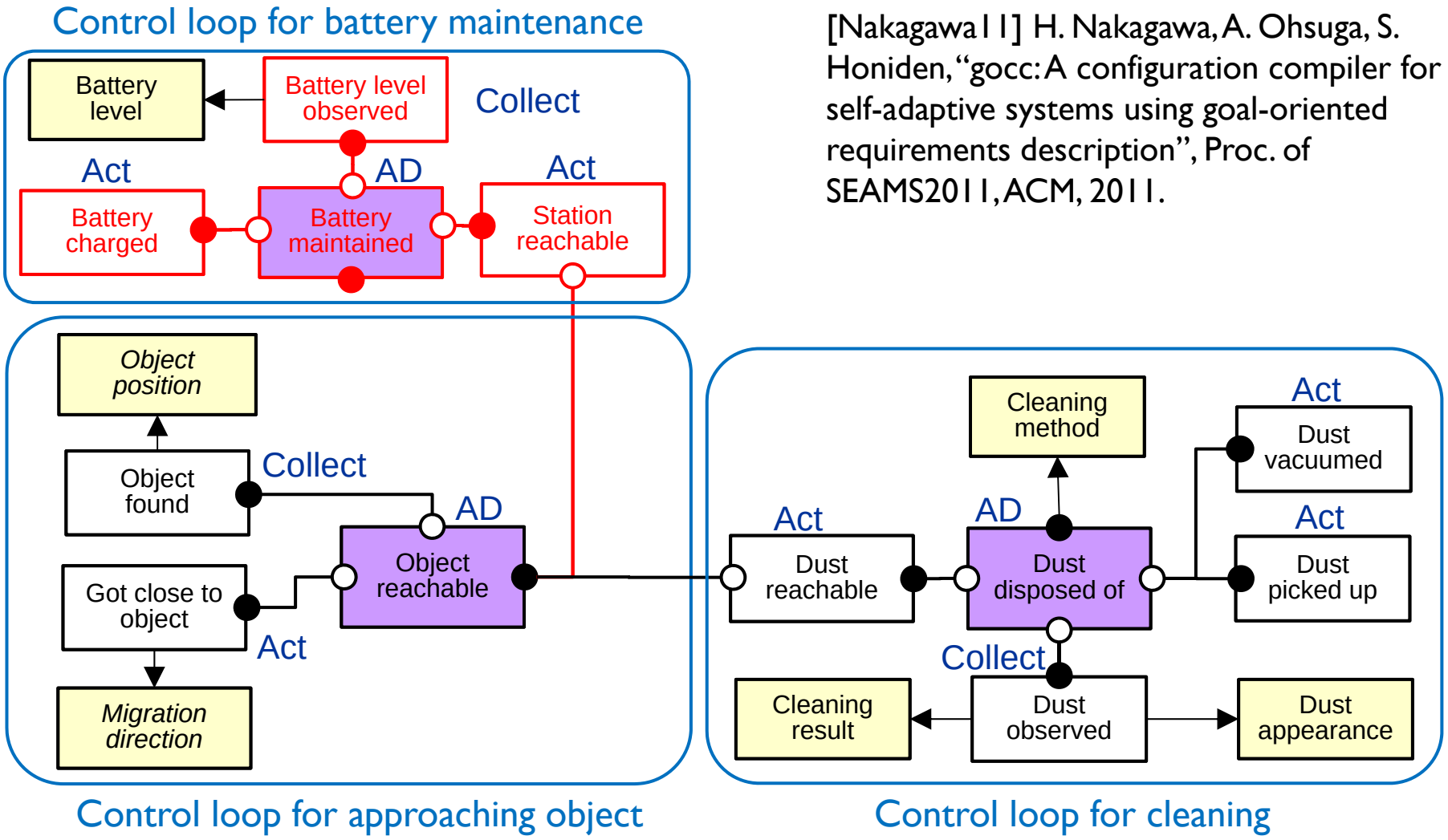


Control loop for cleaning

Control loop for battery maintenance

Control loop for approaching object

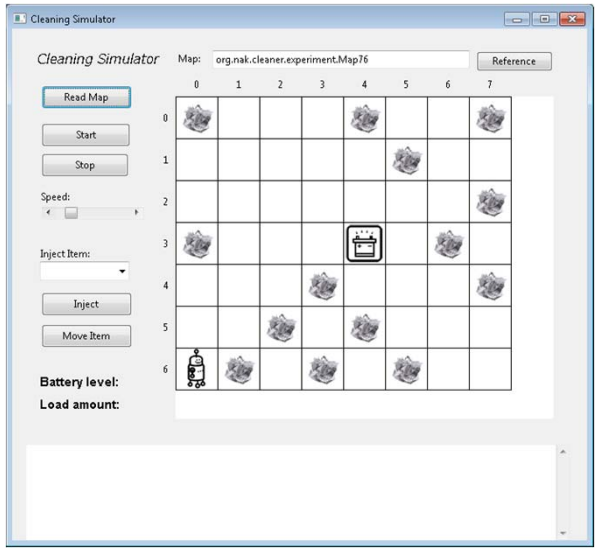
S corresponding to elaborated goal model



[Nakagawa11] H. Nakagawa, A. Ohsuga, S. Honiden, "gocc: A configuration compiler for self-adaptive systems using goal-oriented requirements description", Proc. of SEAMS2011, ACM, 2011.

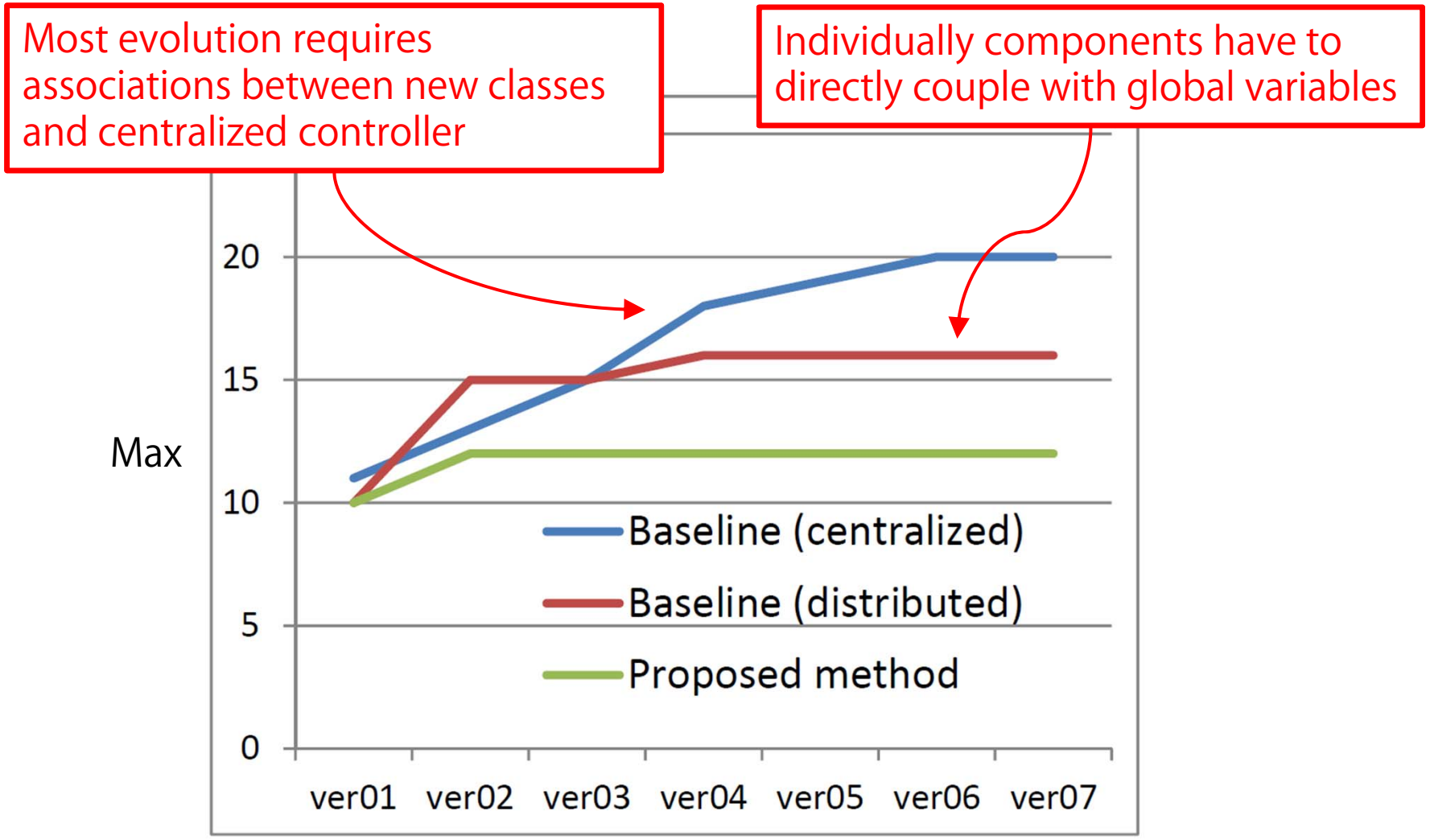
Experiment: Evolution of cleaning robot [Outline]

- Continuously evolve cleaning robots in a simulator
  - Six functions are incrementally added
    - Battery maintenance, additional cleaning methods, load management, ...
  - Three robots
    - **Baseline (centralized)**: robot with a centralized control
    - **Baseline (distributed)**: robot composed of distributed components
    - **Proposed style**: robot according to extracted control loops
- Compare three robots by measuring the increase in code complexity
  - CBO (Coupling Between Objects)
    - One of CK metrics [Chidamber94]
    - Represents coupling complexity
  - Cyclomatic complexity [McCabe76]
    - Represents complexity of control flow



Experiment: Evolution of cleaning robot [Results] CBO

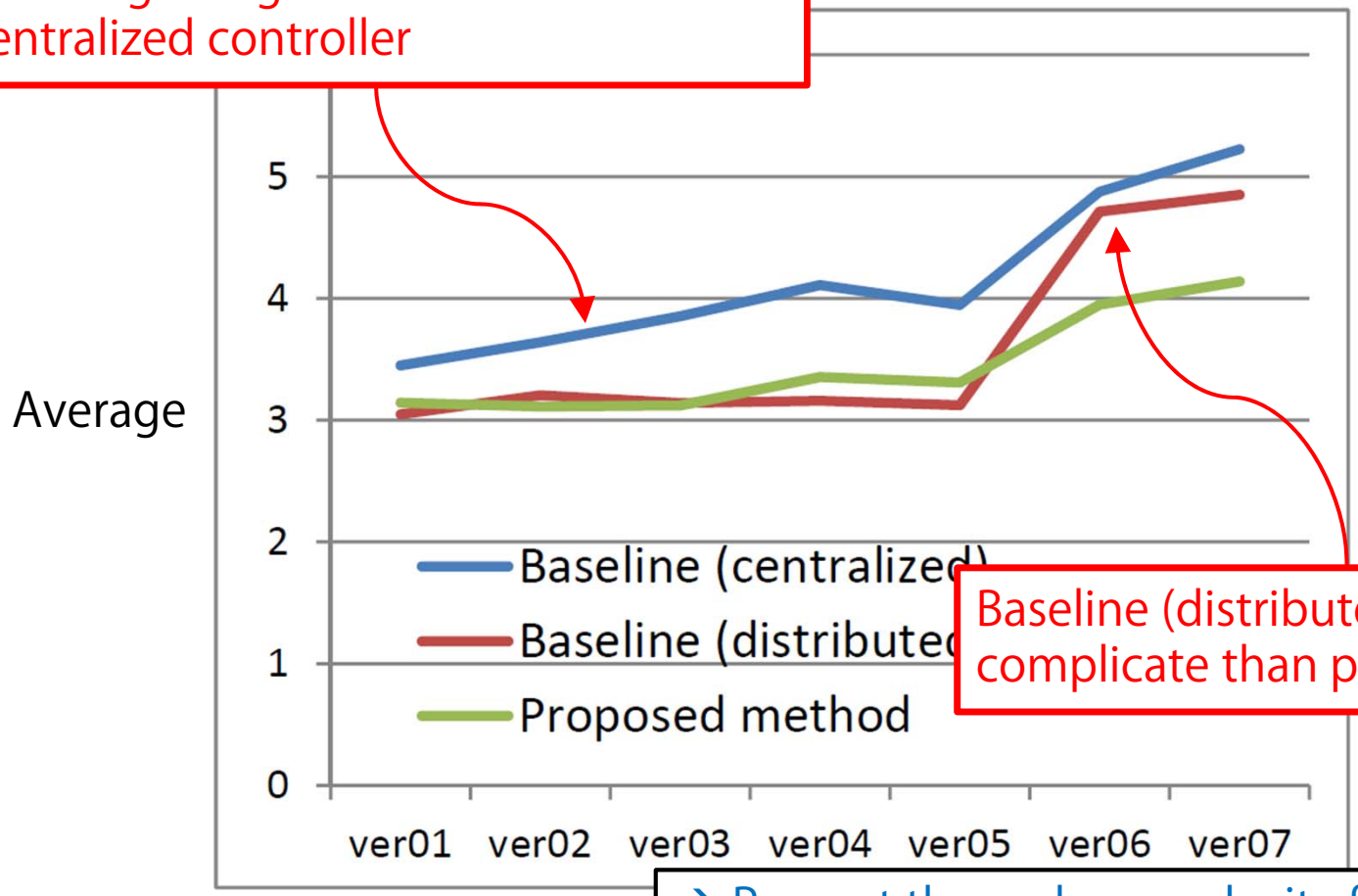
CBO: represents coupling complexity



Experiment: Evolution of cleaning robot [Results] Cyclomatic complexity

Evolution adds condition statements for recognizing new world into the centralized controller

Cyclomatic complexity: represents complexity of control flow



Baseline (distributed) rapidly complicate than proposed method

→ Prevent the code complexity from increasing by localizing the impact of changes

Discussion: Localizing changes

- Impact analysis:
  - Elaboration makes separation of Sc and Sd by control loop block
  - localizing the impact of changes into relevant control loops
    - New prime goals: control loops addition
    - Changes of descendant goals of prime goals: behavioral changes of their control loops

  → Help to analyze the impact of changes

- Change implementation:
  - Control loop modeling prevents code complexity from increasing
    - Modules with higher modularity have lower associated maintenance effort [Bhattacharya12]
  - Adding control loops instead of modifying existing code

  → Prevent evolution cost from increasing

[Bhattacharya12] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution", ICSE 2012, IEEE, 2012.

Conclusions

- To deal with continuous software evolution ...
  - Introduce a goal model elaboration process
  - Extract control loops from goal model as highly independent modules

    → Separate Sc and Sd by control loop block

  → Clear separation makes evolution cost lower