The University of Electro-Communications

gocc: A Configuration Compiler for Self-adaptive Systems Using Goal-oriented Requirements Description

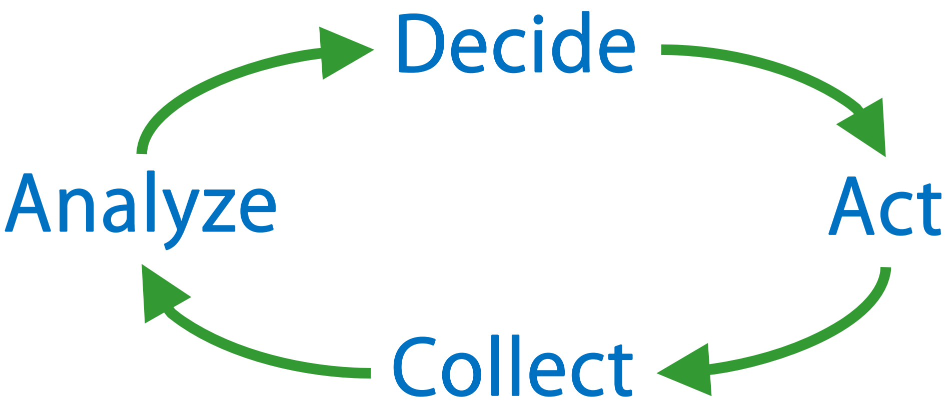
Hiroyuki Nakagawa, Akihiko Ohsuga
The University of Electro-Communications
Shinichi Honiden
National Institute of Informatics

SEAMS 2011, May 23-24, Honolulu, Hawaii, USA

Background: self-adaptive systems and control loop

- **Self-adaptive Systems:**
  – able to dynamically change behaviors
  – usually form a control loop

- **Control loop [Dobson ’06, Shaw ’95]**
  – summarized as *collect*, *analyze*, *decide*, and *act*

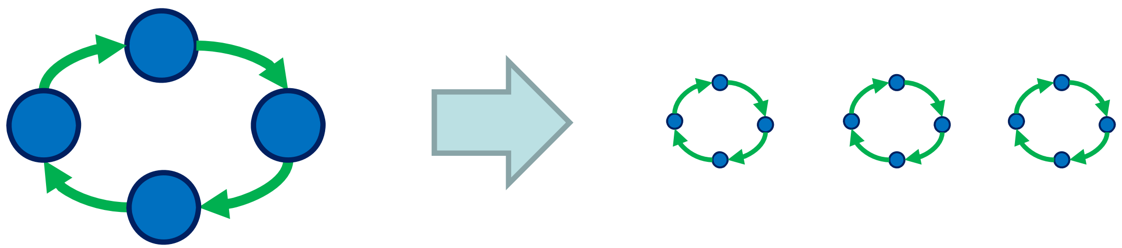


[Shaw ’95], Mary Shaw, "Beyond Objects: A Software Design Paradigm Based on Process Control", ACM SIGSOFT Software Engineering Notes Homepage archive Volume 20 Issue 1, Jan. 1995
[Dobson ’06] Dobson, et al., “A survey of autonomic communications”, ACM Transaction on Autonomous and Adaptive Systems

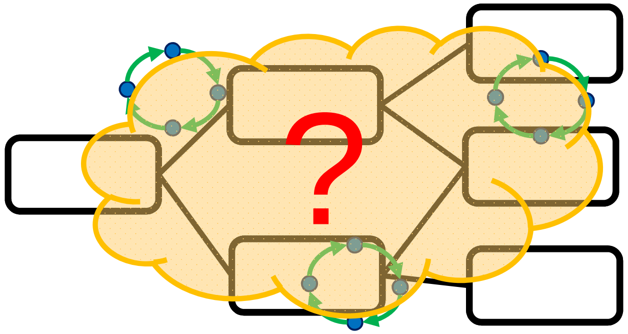
Background: multiple control loops

- **Issues of single control loop systems**
  - Difficulty of keeping consistency among activities
  - Hard to evolve

→ Divide into multiple control loops

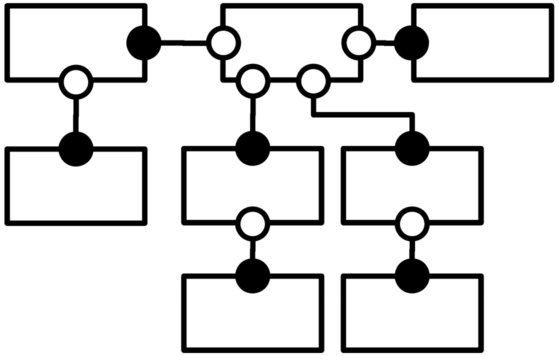


- ***But, how to implement?***
  - *How to realize multiple control loops as system architecture and implement them?*



Approach

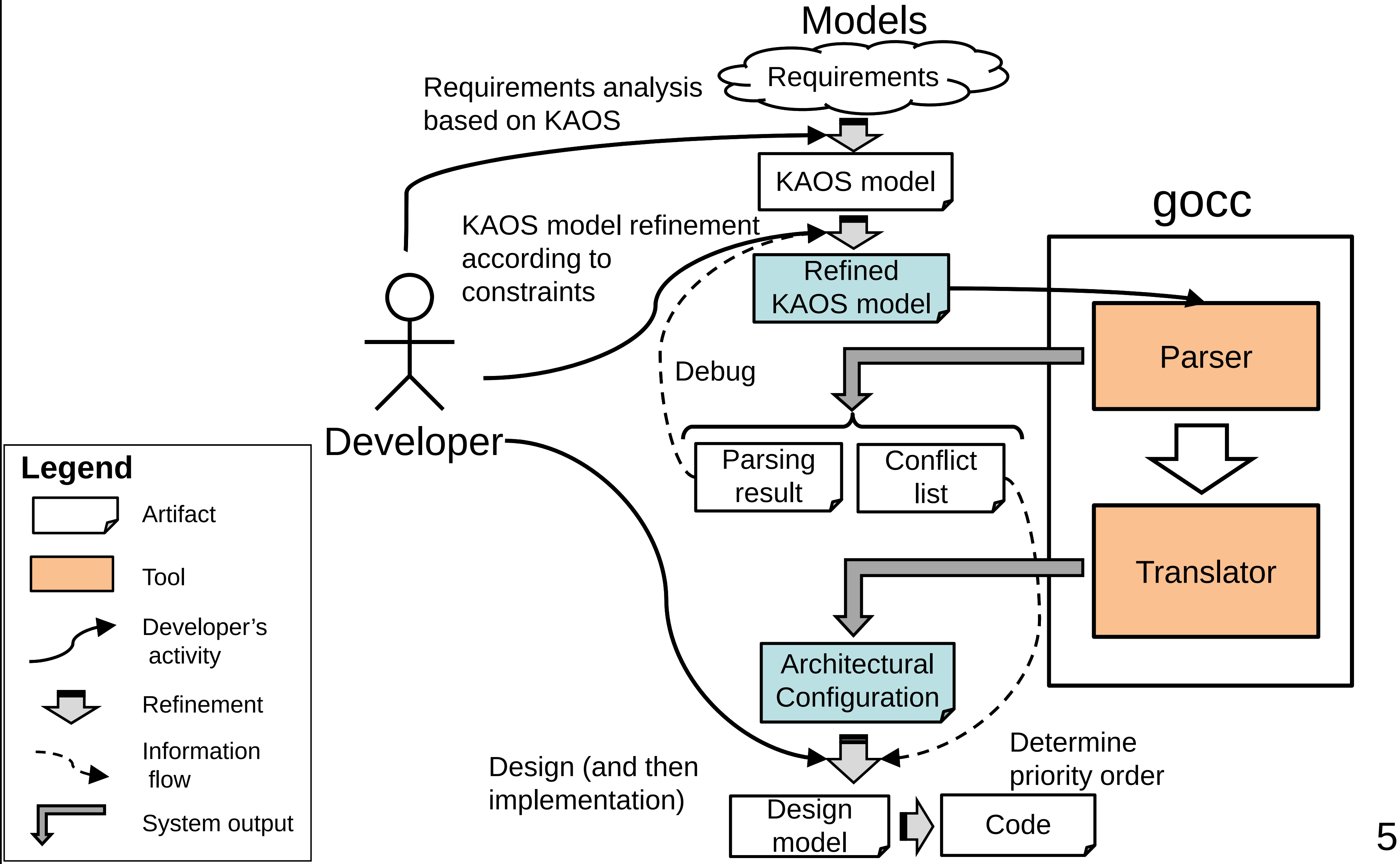
- **Goal: To determine configurations containing multiple control loops**
  - Configuration: represents system architecture by components connections such as [Oreizy '99] [Kramer, Magee '07]



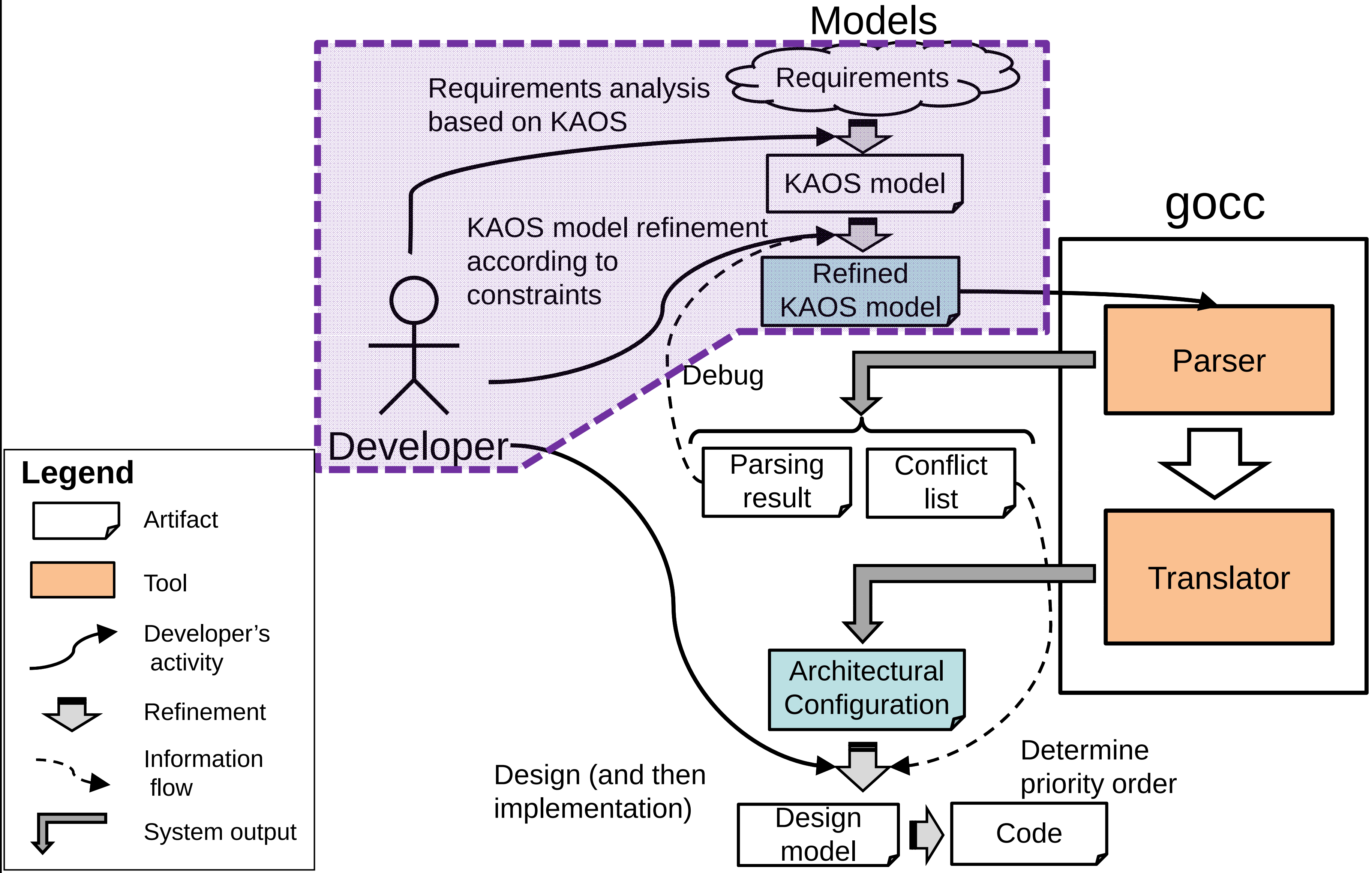
- **Approach:**
  - Make use of requirements description
    - Requirements are described in goal model
      - For separating concerns, leading to control loop separation
      - Goal model structure helps construct configurations
  - Generate Configuration from goal model
    - gocc: Goal-oriented configuration compiler

[Kramer, Magee '07] J. Kramer, J. Magee, "Self-managed systems: an architectural challenge", FOSE '07.
[Oreizy '99] P. Oreizy, M. M. Gorlick, R. N. Taylor, et al., "An architecture-based approach to self-adaptive software", IEEE Intelligent Systems, 1999.

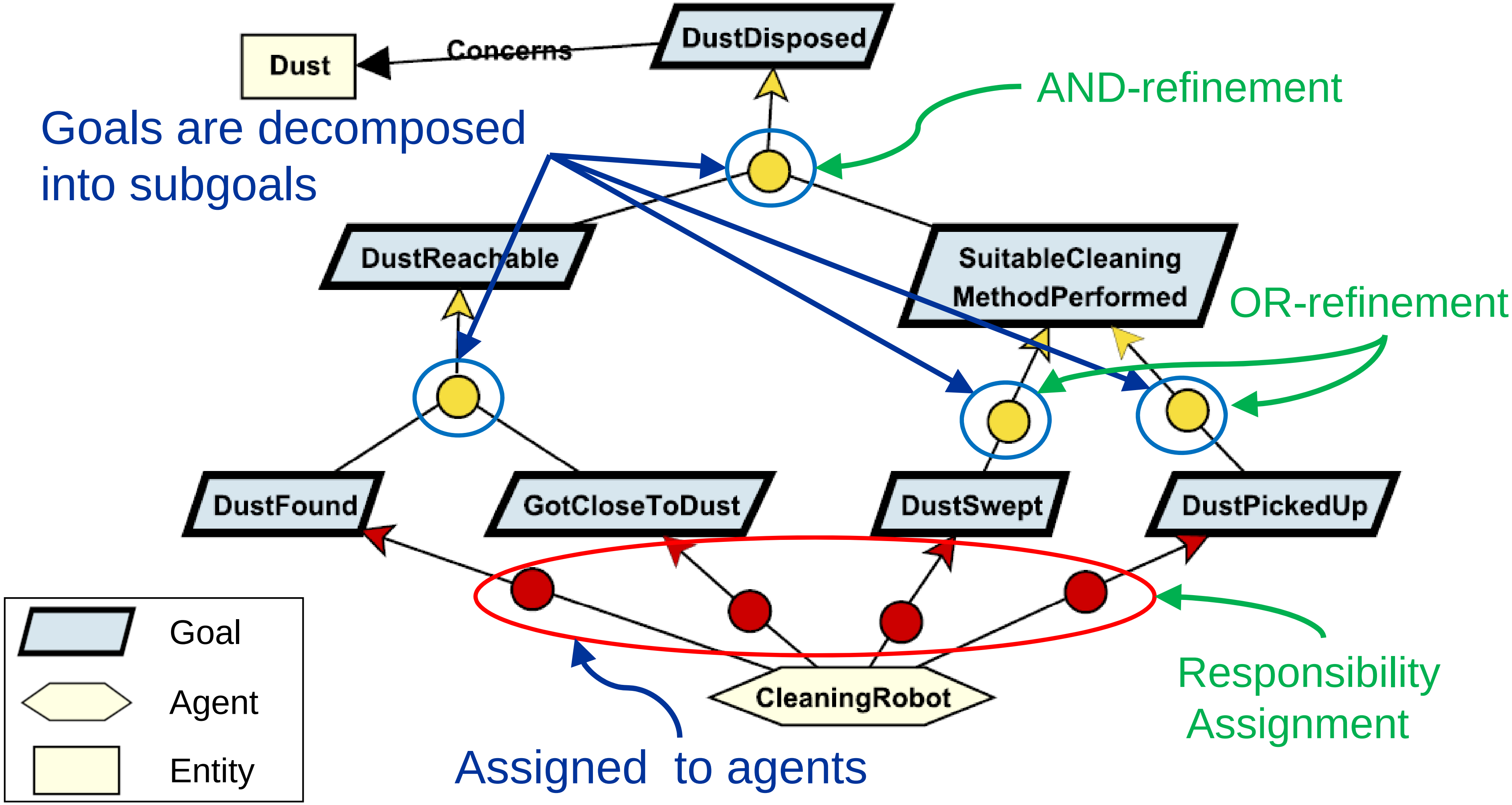
Development process using gocc



Goal modeling



KAOS goal model [Dardenne 93]

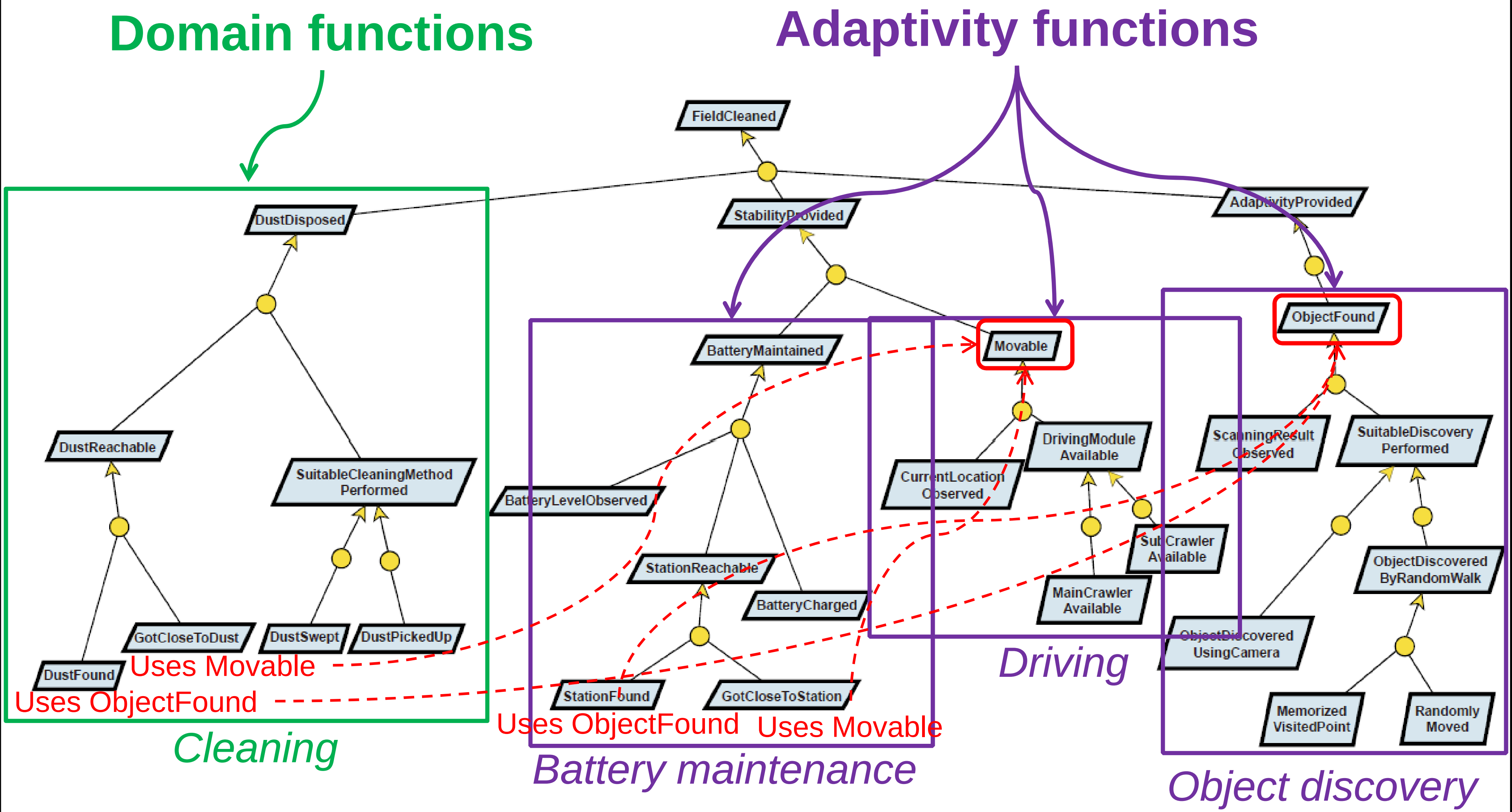


[Dardenne 93] A. Dardenne, A. van Lamsweerde, S. Fickas, “Goal-directed requirements acquisition”, *Science of Computer Programming,* 20(1-2):3–50, 1993.

Constraints on KAOS

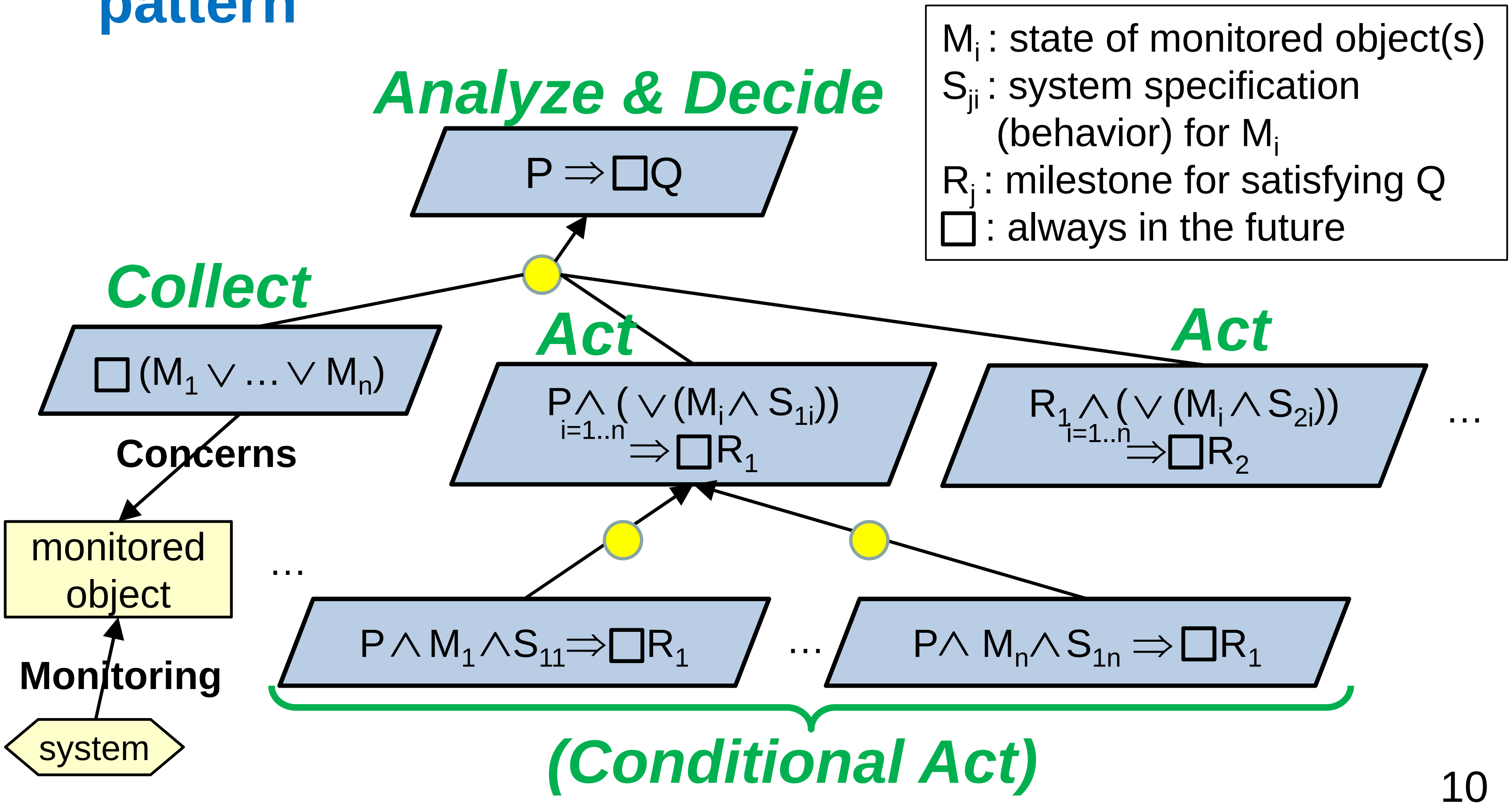
- **Add constraints on KAOS model to derive configuration with multiple control loops**
  - **Separation of concerns**
    - Explicitly separate domain functions from adaptivity functions
      - By separating branches
    - Put "Uses" label for representing goal dependencies
      - Goal A needs goal B for its satisfaction → "Uses B" on goal A
  - **Control loop embedding**
    - Elaborate branches according to the control loop pattern
    - Assign Analyze & Decide goals to the system as responsibilities
      - not individual goals but subtrees
    - Identify objects
      - Monitored objects: system can detect changes of their situation
      - Resources: environmental or system resources (e.g. H/W units)

KAOS model: Cleaning Robot



Control loop pattern

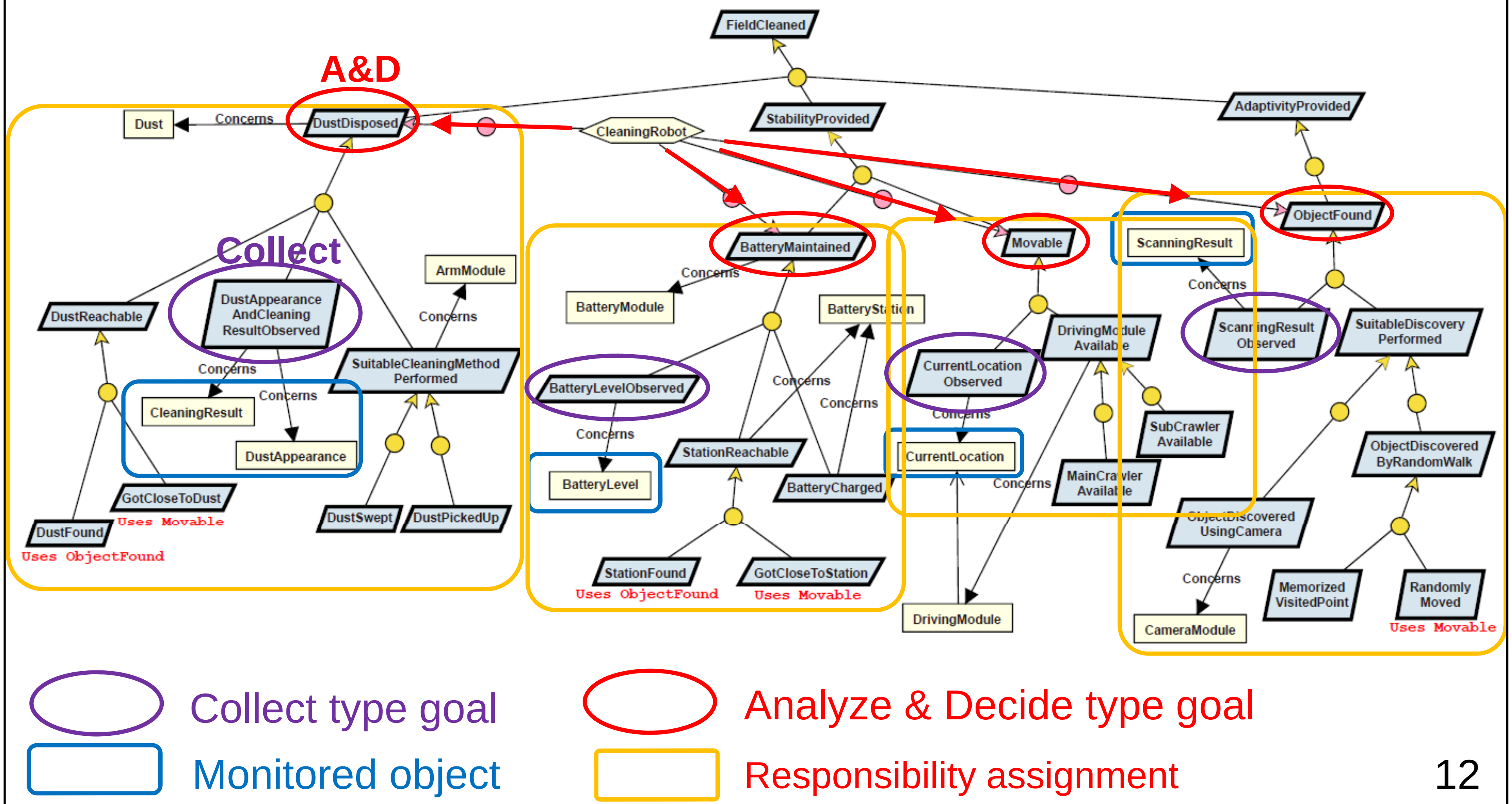
- **Elaborate branches according to Control loop pattern**



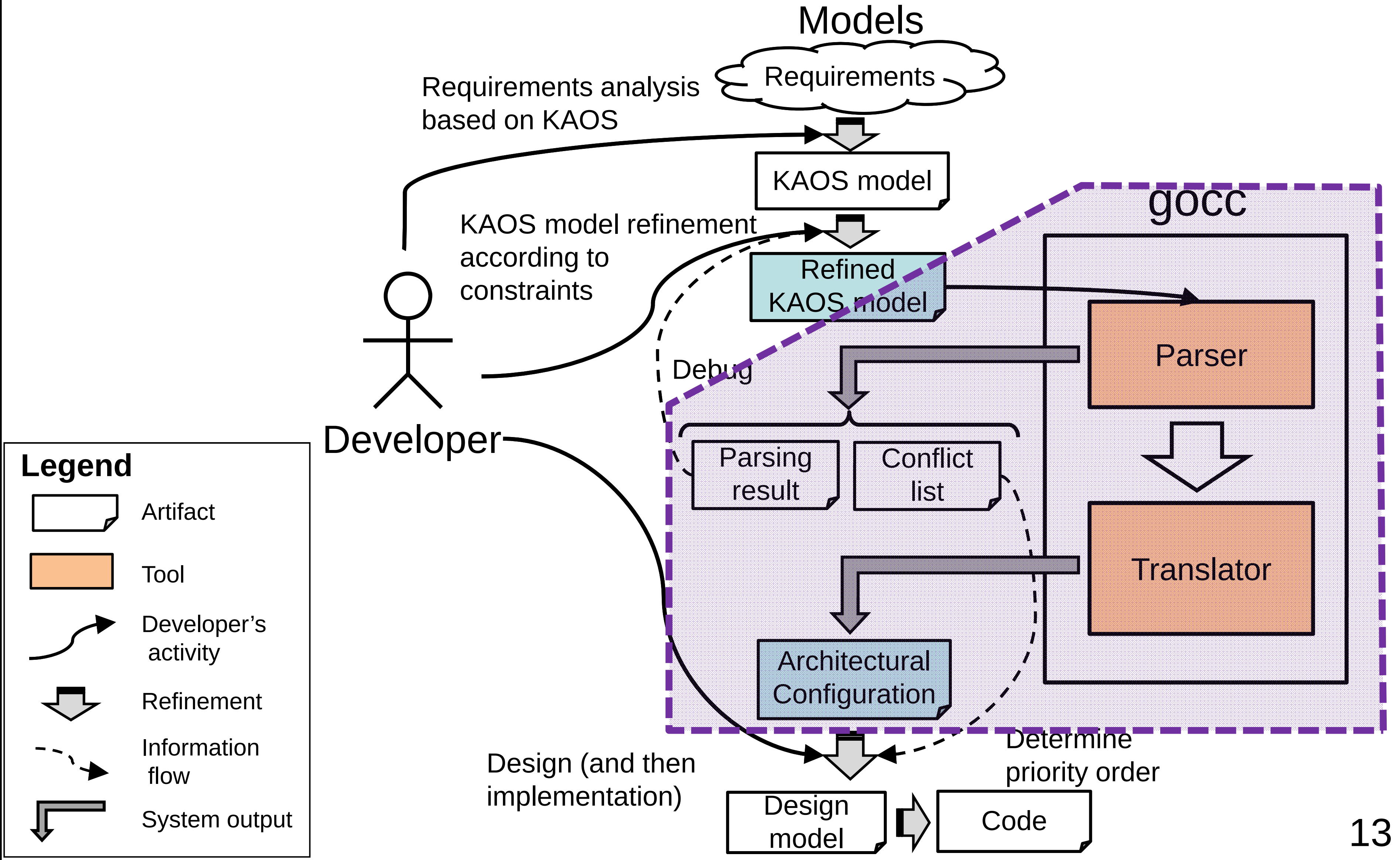
Constraints on KAOS

- **Add constraints on KAOS model to derive configuration with multiple control loops**
  - **Separation of concerns**
    - Explicitly separate domain functions from adaptivity functions
      - By separating branches
    - Put "Uses" label for representing goal dependencies
      - Goal A needs goal B for its satisfaction → "Uses B" on goal A
  - **Control loop embedding**
    - Elaborate branches according to the control loop pattern
    - Assign Analyze & Decide goals to the system as responsibilities
      - not individual goals but subtrees
    - Identify objects
      - Monitored objects: system can detect changes of their situation
      - Resources: environmental or system resources (e.g. H/W units)

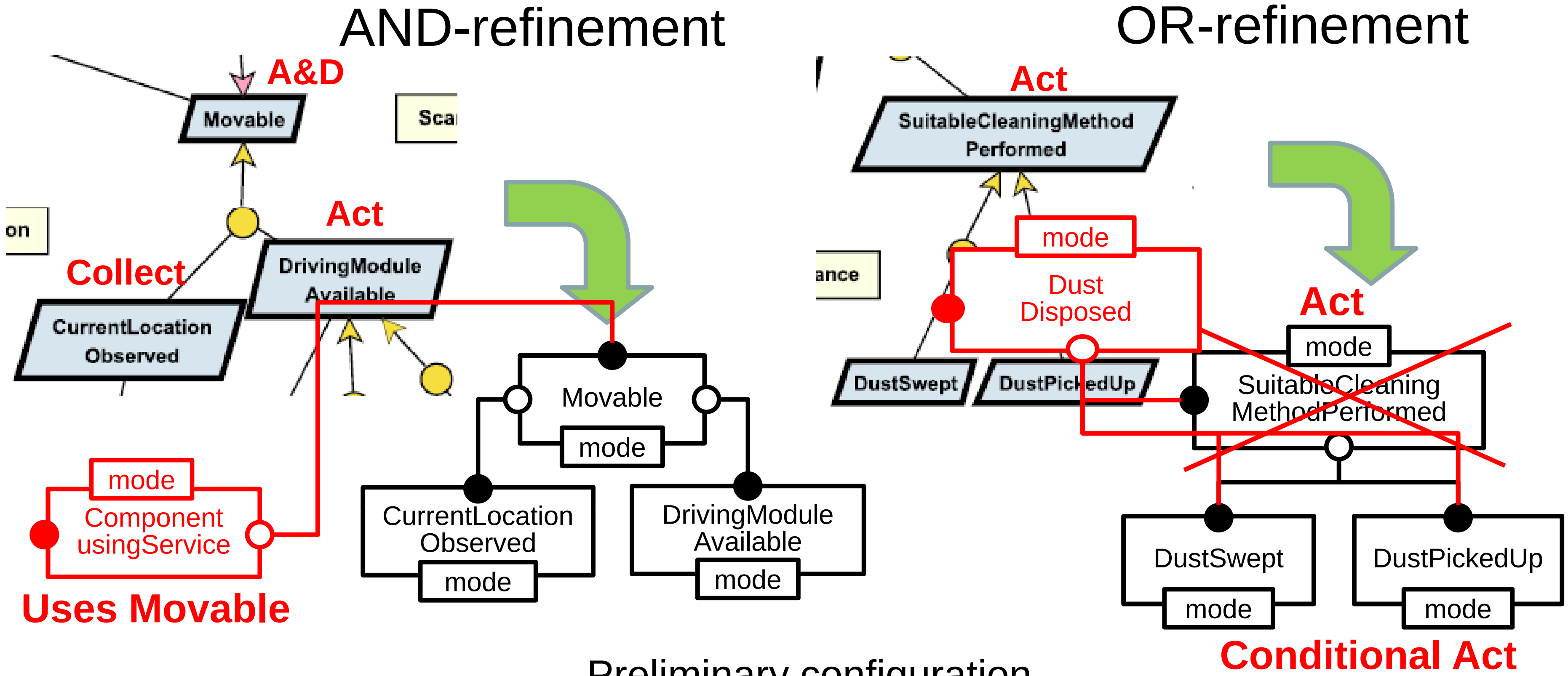
Refined KAOS model: Cleaning Robot



Configuration assembly

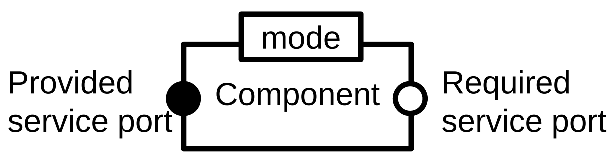


Configuration generation: Transformation from goal model

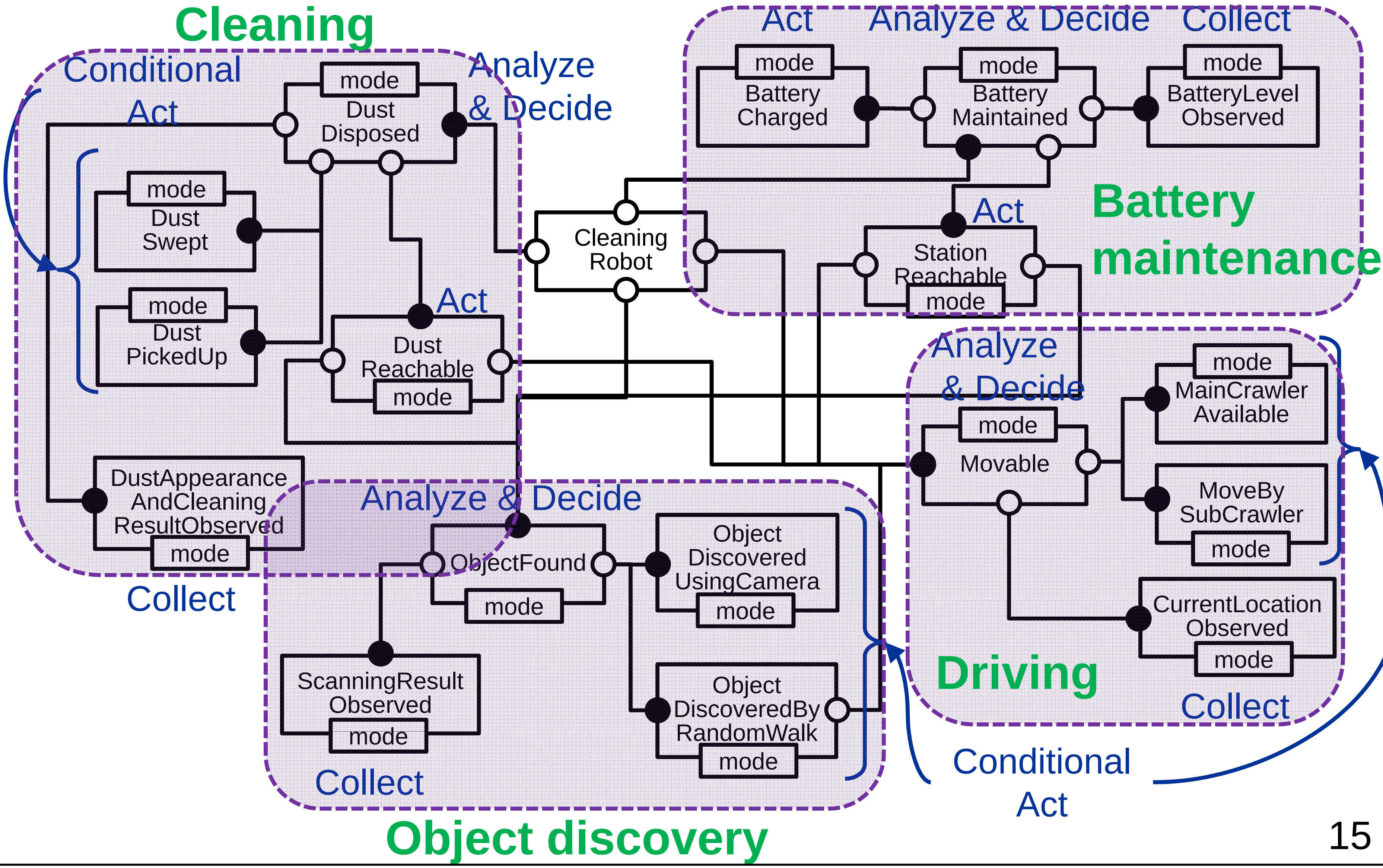


- Preliminary configuration
  - Combine goals with components
  - Connect components according to the refinement links
- Elaborate configuration
  - Join components according to Uses labels
  - Replace the direct connections of Alternative Act by eliminating intermediate Act

Component model
[Hirsch '06, Kramer, Magee '07]

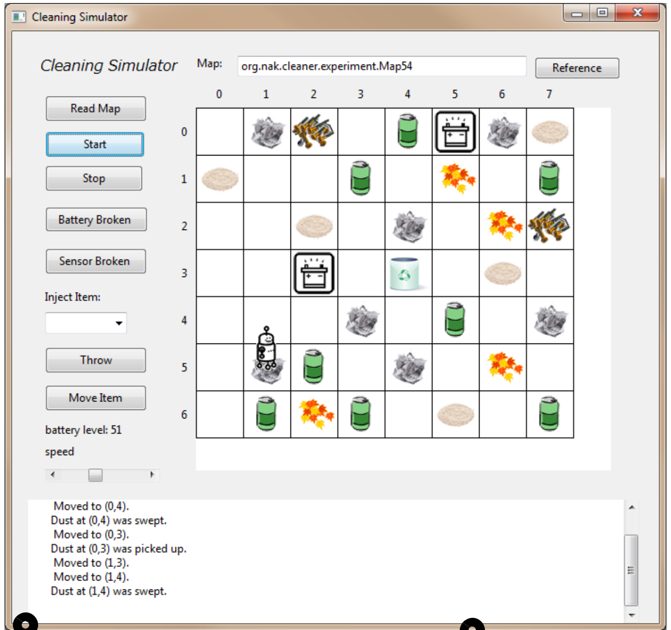


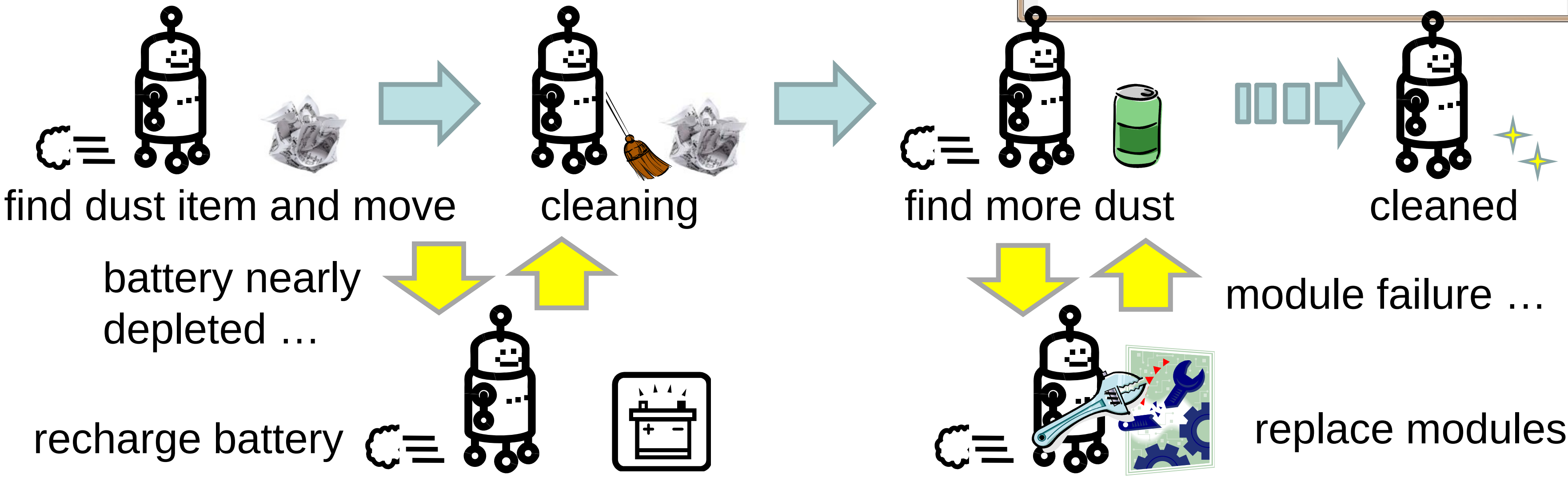
Generated configuration



Case study

- **Objective: evaluate feasibility of our development process**
  - Implement cleaning robot on a simulator according to the generated configuration
  - Add new function after development





Applicability

- **We observed robot's reactions when various unexpected events occurred**
  - Switching active process:
    - Pushed "Battery Broken" button
      → Robot changed target to battery station

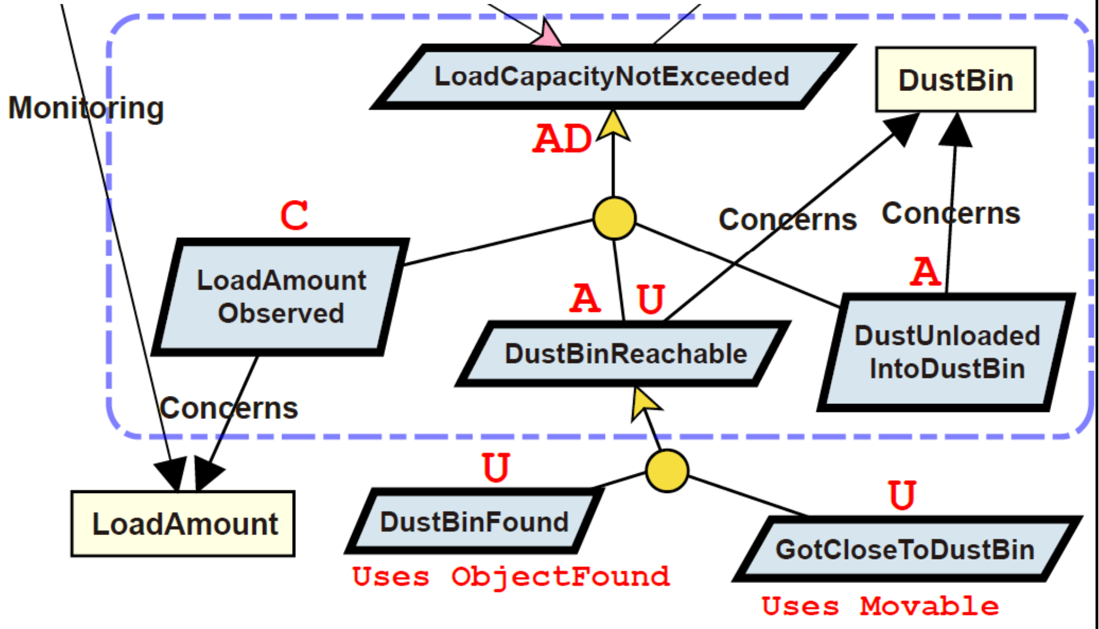
    → Multiple control loops provide continuous monitor

  - Switching components:
    - Pushed "Sensor Broken" button
      → Robot changed object discovery method from using camera to random walk

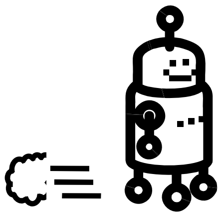
    → AD component chose a suitable Conditional Act components by analyzing data collected by Collect component

Evolvability

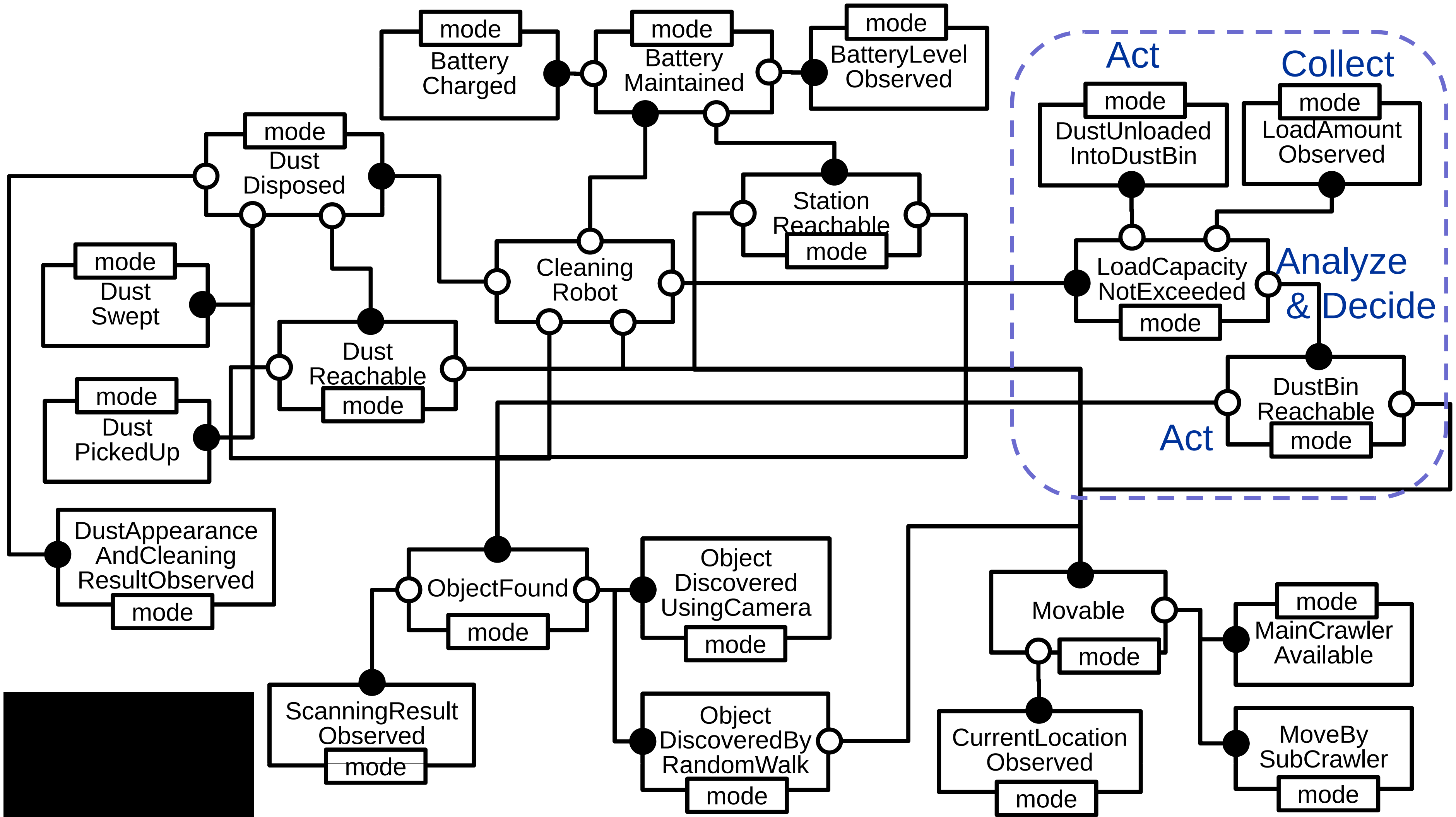
- **We evolved the robot to react to new requirement**
  - Requirements: load amount management
- **Development activities**
  - Added goals
  - Applied control loop pattern
  - Detected conflicts
  - Implemented



Embedded subtree



New configuration

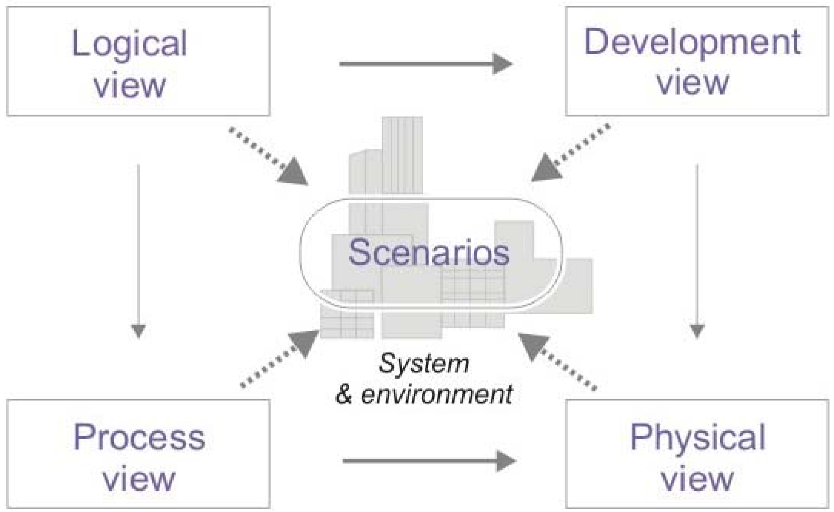


[redacted]

Discussion (1/2)

Based on 4+1 architectural views [Kruchten '95]:

- **Logical view:**
  - Our configuration
    - Contains multiple control loops
    - Tends to consist of a large number of components
- **Process view:**
  - Concurrent control loops execution causes overhead
- **Physical view:**
  - Requires a platform that executes implemented components concurrently



[Kruchten '95] P. Kruchten, "The 4+1 view model of architecture", IEEE Software, 12:42–50, 1995.

Discussion (2/2)

- **Development view:**
  - Configuration generation from requirements model
    - Helps developers deal with requirements changes or software evolution
    - Separation of control loops eases the individual control loop implementation
    - Aggregation of control loops is not easy when they interfere with one another
- **Scenarios:**
  - Milestone-driven refinement [Darimont '96] enables scenario injection
    - Act components: responsible to achieve individual milestones
    - A&D components: responsible to control Act components to go forward the scenarios

[Darimont '96] R. Darimont and A. van Lamsweerde, "Formal refinement patterns for goal-driven requirements elaboration", 4th ACM SIGSOFT/FSE, 1996.

Conclusion

Development process for self-adaptive systems with multiple control loops

- **Introduce configuration generating compiler**
  - Make use of goal-oriented requirements description
    - Add constraints on the goal model
    - Introduce configuration compiler
    - Detect conflicts on the goal model
- **Evaluate feasibility through a case study**
  - Implementing a cleaning-robot on simulator
- **Remained and future work**
  - Further connection between configuration and code

→ Realizing self-adaptive systems in accordance with requirements