ascens

ASCENS: Towards Systematically Engineering Ensembles

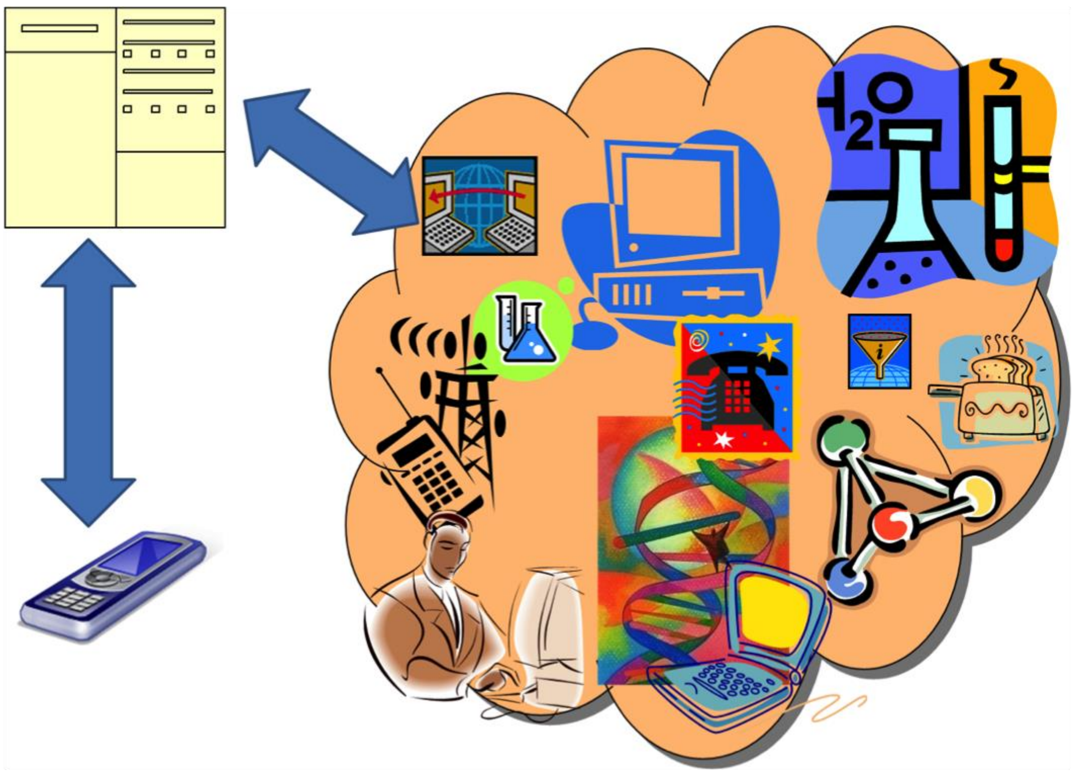
Martin Wirsing in cooperation with
Matthias Hölzl, Annabelle Klarl, Nora Koch, Mirco Tribastone,
Franco Zambonelli

Dynamische und Adaptive Systeme,
November 2013

SEVENTH FRAMEWORK PROGRAMME

Future Emerging Technologies

- **Autonomic systems** are typically distributed computing systems whose components act autonomously and can adapt to environment changes.
- We call them **ensembles** if they have some of the following characteristics:
  - Large numbers of nodes
  - Heterogeneous
  - Operating in open and non-deterministic environments
  - Complex interactions between nodes and with humans or other systems
  - Dynamic adaptation to changes in the environment



ASCENS Project

- Goal of ASCENS:

  Develop methods, tools, and theories for

  modeling and analysing

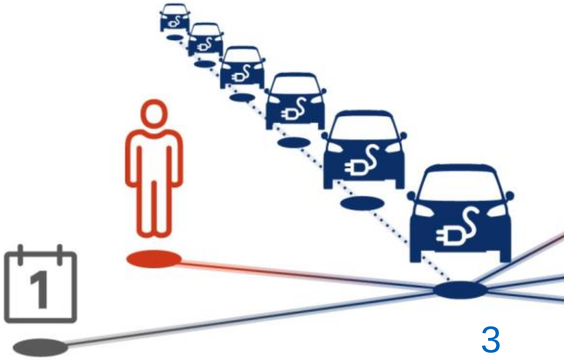
  autonomic self-aware systems

  that

  - combine traditional SE approaches based on formal methods with the flexibility of resources promised by autonomic, adaptive, and self-aware systems

- Partners:
  - LMU (Coordinator), U Pisa, U Firenze with ISTI Pisa, Fraunhofer, Verimag, U Modena e Reggio Emilia, U Libre de Bruxelles, EPFL, Volkswagen AG, Zimory GmbH, U Limerick, Charles U Prague, IMT Lucca, Mobsya

- Case studies:
  - Robotics, cloud computing, and energy saving e-mobility



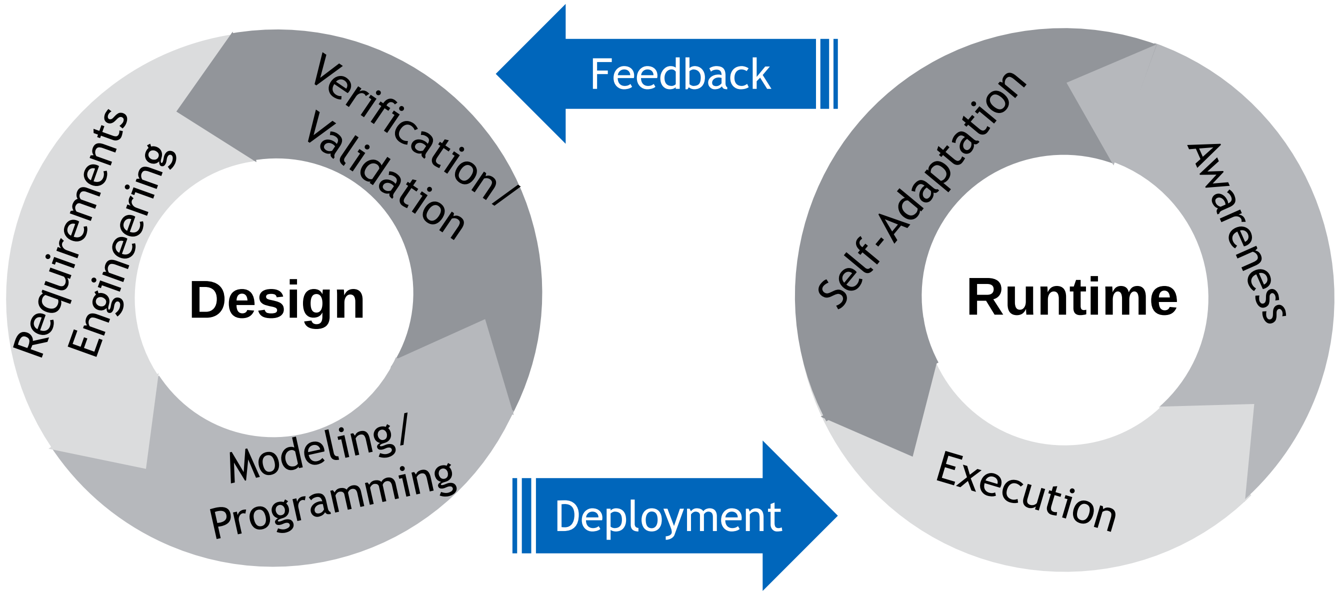
Engineering Autonomic systems

- Self-aware ensemble components are aware of their structure and their aims
  - Goals and models of ensemble components have to be available at runtime
  - Autonomous components typically have internal models and goals

- For ensuring reliability and predictability of the ensemble and its components important properties of the ensemble should be defined and established at design time and maintained during runtime
  - Analysis-driven development and execution

- Autonomic systems have to be able to adapt to dynamic changes of the environment
  - Even if the ensemble components are defined at design time, adaptation of the ensemble components will happen at runtime

Ensemble Lifecycle: Two-Wheels Approach



- Engineering an autonomic ensemble consists of an iterative agile lifecycle
  - Design time: Iteration of requirements engineering, modeling, validation
  - Runtime: Awareness, adaptation, execution loop
  - Design time and runtime loops connected by deployment and feedback
    - Feedback leads to a better understanding and improvement of the system.

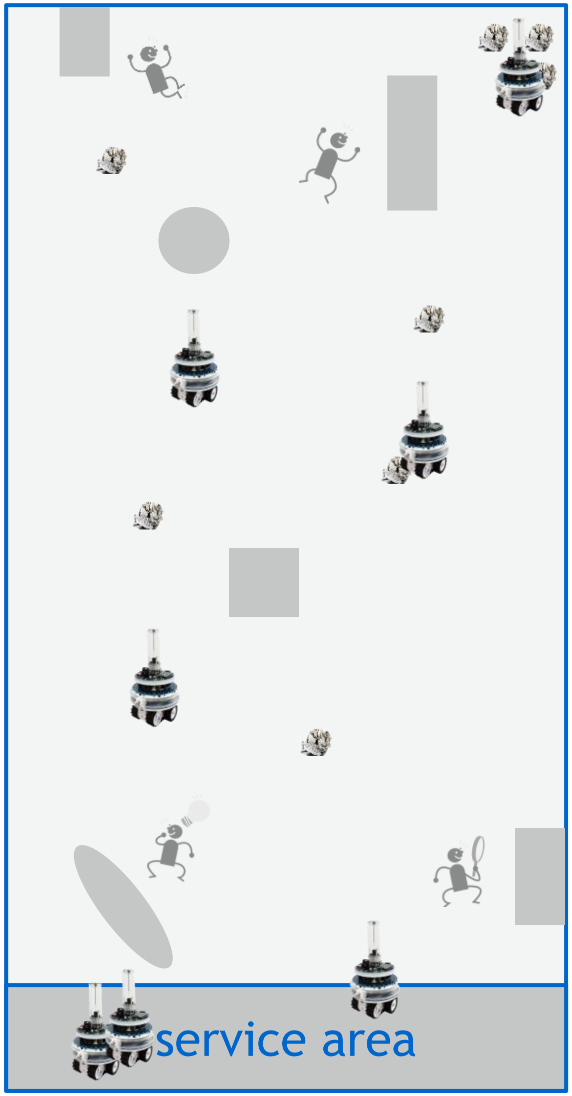
Outline

For the sake of simplicity we restrict ourselves to a simple example of autonomic robots and illustrate only the following first development steps which happen at design time.

- Requirements specification with SOTA/GEM
- Coarse modeling by adaptation pattern selection
- Fine-grained modeling in Helena
- Abstract programming in SCEL
- Quantitative analysis of autonomic system behaviour using stochastic methods

The Robot Case Study

- Swarm of garbage collecting robots
  - Acting in a rectangular exhibition hall
  - The hall is populated by visitors and exhibits
- Scenario
  - Visitors drop garbage
  - Robots move around the hall,
    pick up the garbage and
    move it to the service area
  - Robots may rest in the service area in order to not intervene too much with the visitors and to save energy



Domain and Requirements Modeling: SOTA/GEM Framework

- An adaptive system can (should?) be expressed in terms of "goals" = "states of the affairs" that an entity aims to achieve
  - Without making assumptions on the actual design of the system
  - It is a requirements engineering activity

- SOTA ("State of the Affairs")/GEM Conceptual framework
  - Goal-oriented modeling of self-adaptive systems
  - Functional requirements representing the states of affairs that the system has to achieve or maintain
  - Utilities are non-functional requirements which do not have hard boundaries and may be more or less desirable.
  - GEM is the mathematical basis of the SOTA framework

Domain modeling:

- State Of The Affairs $Q = Q_1 \times \ldots \times Q_n$
  - represents the state of all parameters that
    - may affect the ensemble's behavior and
    - are relevant to its capabilities

- Example: Robot Swarm State Of The Affairs

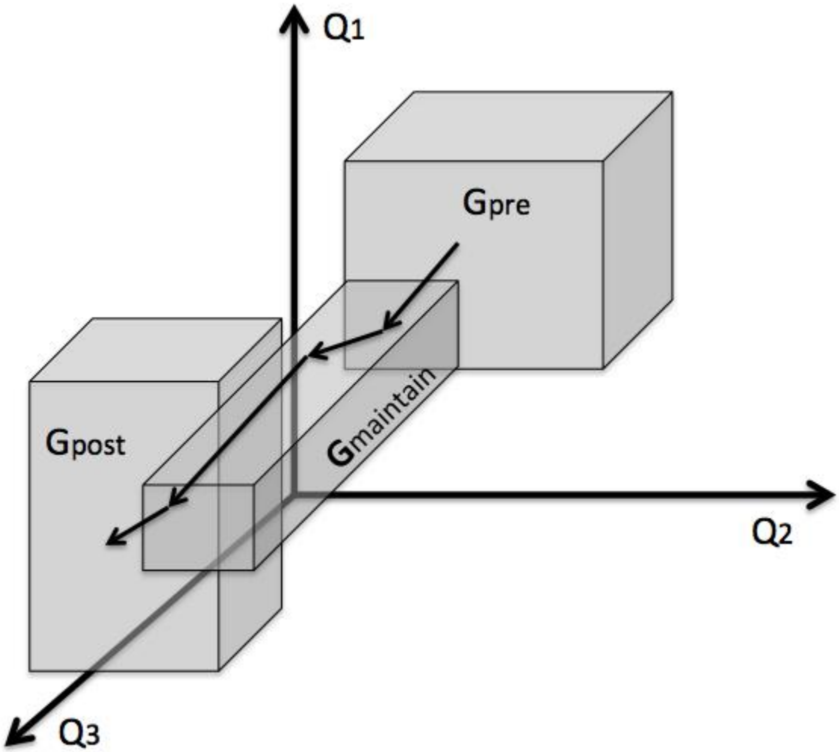
$p_i = \langle x_i, y_i \rangle \in \mathbb{R} \times \mathbb{R}$	Position of robot i
$\mathsf{Area} \subseteq \mathbb{R} \times \mathbb{R}$	Exhibition Area
$s_i \in \{\mathsf{Searching}, \mathsf{Resting}, \mathsf{Carrying}\}$	State of robot i
$g \in \left\{ \langle \gamma_1, \ldots, \gamma_K \rangle \mid \gamma_i \in \mathsf{Area}, K \in \mathbb{N} \right\}$	List of garbage item positions
$o^\flat \in \mathbb{B}$	Exhibition open for public?
$\mathbf{Q} = \{\langle p_1, s_1, \ldots, p_N, s_N, g, o^\flat \rangle \mid p_i \in \mathsf{Area}\}$	State space

Ensemble and its Environment

- Environment
  - For mathematical analysis we distinguish often between the ensemble and its **environment** such that the whole system is a combination of both
- Adaptation Space
  - The ensemble should work in a number of different environments
  - The characteristics of all environments are described by the **adaptation space**

- Example Robot Swarm
  - The **state space of the robot ensemble** is given by the state spaces all robots where Q^{Robot} is given by the position and state of the robots
  - The **state space environment** is given by the exhibition area, the list of garbage items, and the value indicating whether the exhibition is open
  - The **adaptation space** of the ensemble may be given by varying the size of the arena, the dropping rate of garbage items, etc.

- **Goal-oriented requirements modelling**
- **Goal** = achievement of a given state of the affairs
  - Where the system should eventually arrive in the phase space Q^e,
  - represented as a confined area in that space (post-condition G_{post}), and
  - the goal can be activated in another area of the space (pre-condition G_{pre})
- **Utility** = how to reach a given state of the affairs
  - “maintain goal”: constraints on the trajectory to follow in the phase space Q^e
  - expressed as a subspace $G_{maintain}$ in Q^e

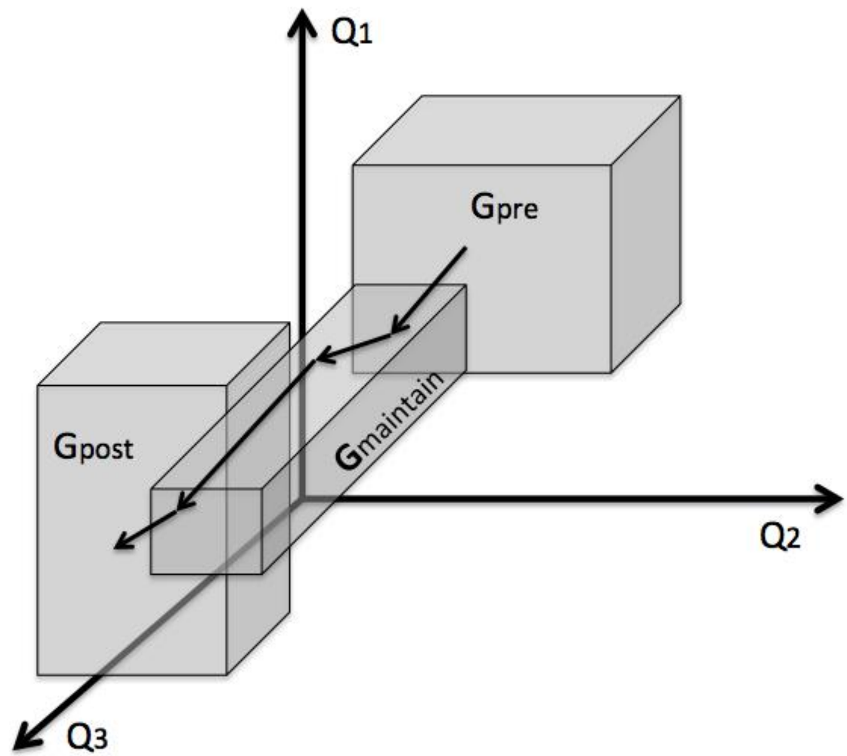


Robot Ensemble Goals and Utilities

- Example requirements:
  - Goal G[1]
    - Maintains < 300 garbage items as long as the exhibition is open

$$G^1_{pre} \equiv o^b = true$$
$$G^1_{maintain} \equiv g^\# < 300$$
$$G^1_{post} \equiv o^b = false$$

    - i.e. □ (o^b => $g^\#$ < 300 until not o^b)
  - Further (adaptation) goals
    - Keep energy consumption lower than predefined threshold
    - In resting area allow sleeping time for each robot
  - Adaptation Space
    - Size of arena x garbage dropping rate

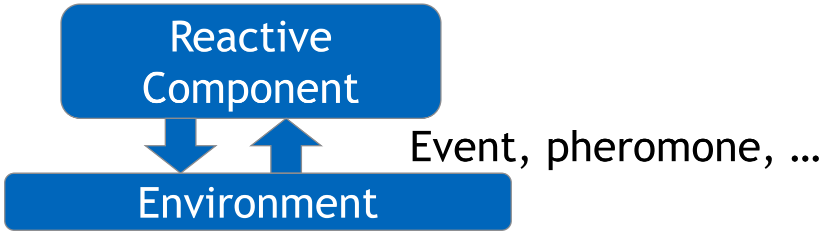


Towards Design

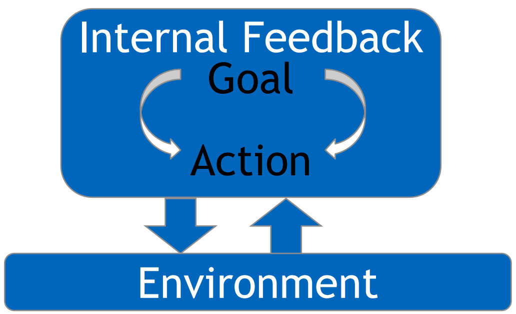
- Further requirements modelling steps
  - Check consistency of requirements
- Model the autonomic system in Helena/Poem
  - Select suitable **adaptation patterns** for ensemble design
  - Model each component and the ensemble in Agamemnon
  - (Implement each component in Poem
  - Provide abstractions for controlling adaptation
    - e.g., by learning behaviours or reasoning)
- Refine the model to a SCEL design
  - Based on the Helena model
  - Use analysis tools for predicting the behaviour and improving the design

Component Patterns

- **Reactive**



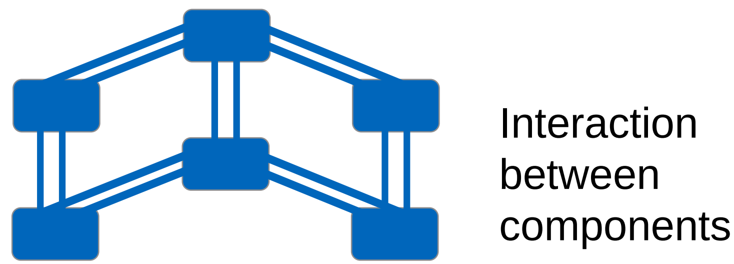
- **Internal feedback loop**



Ensemble Patterns

Environment mediated (swarm)

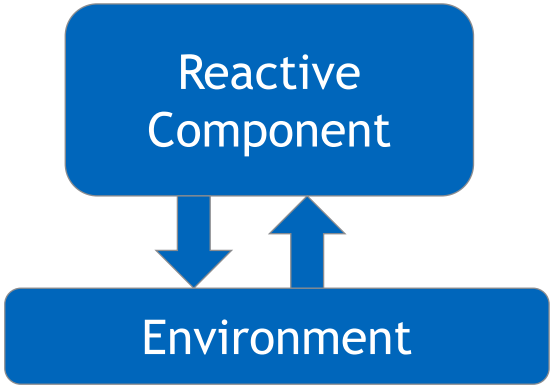
Negotiation/competition

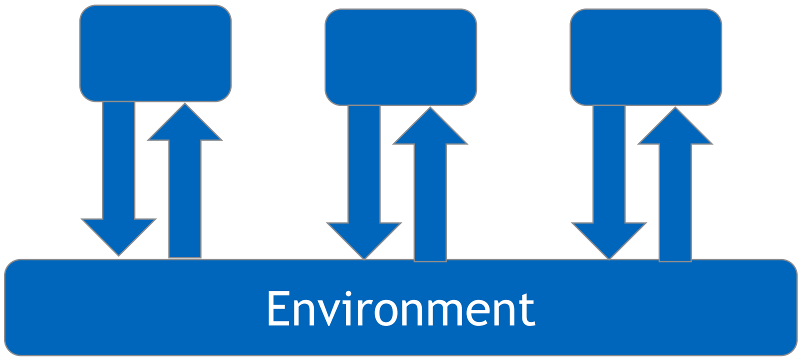


- **Further patterns:** External feedback loop, norm-based ensembles, …

Robot Ensemble Adaptation

- **Reactive component pattern** for implementing a single robot
- **Environment mediated (swarm) pattern** for the ensemble of ineracting components





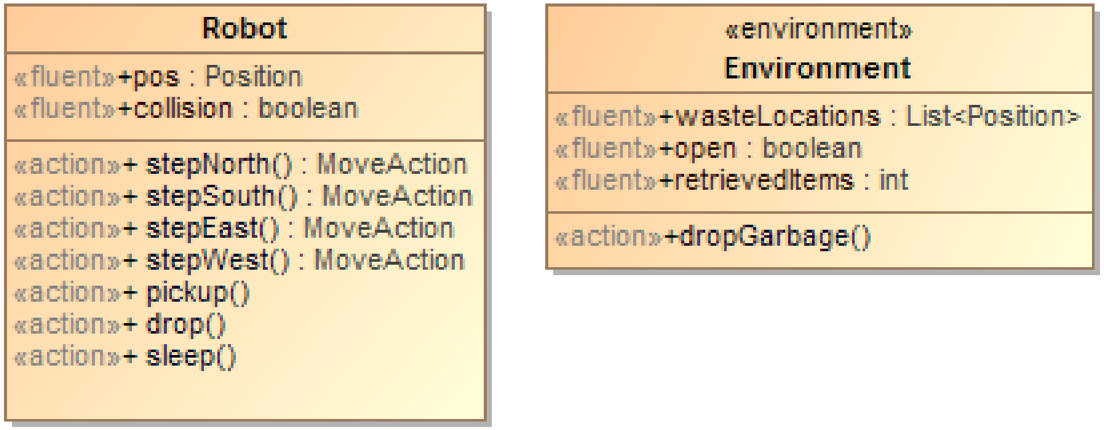
Helena

Helena is a UML-based approach for modeling ensembles of components.

- **Dynamic behaviour** of (service) components is described by a UML profile based on the situation calculus.
  - **Domain models** are UML class diagrams
    - with properties (=fluents) and actions
  - **Behaviour specification** by UML activity diagrams
    - stereotypes for the specification of partial programs and their computation via learning or planning

Helena Model: Domain Model

Model of components together with their properties (=fluents) and actions

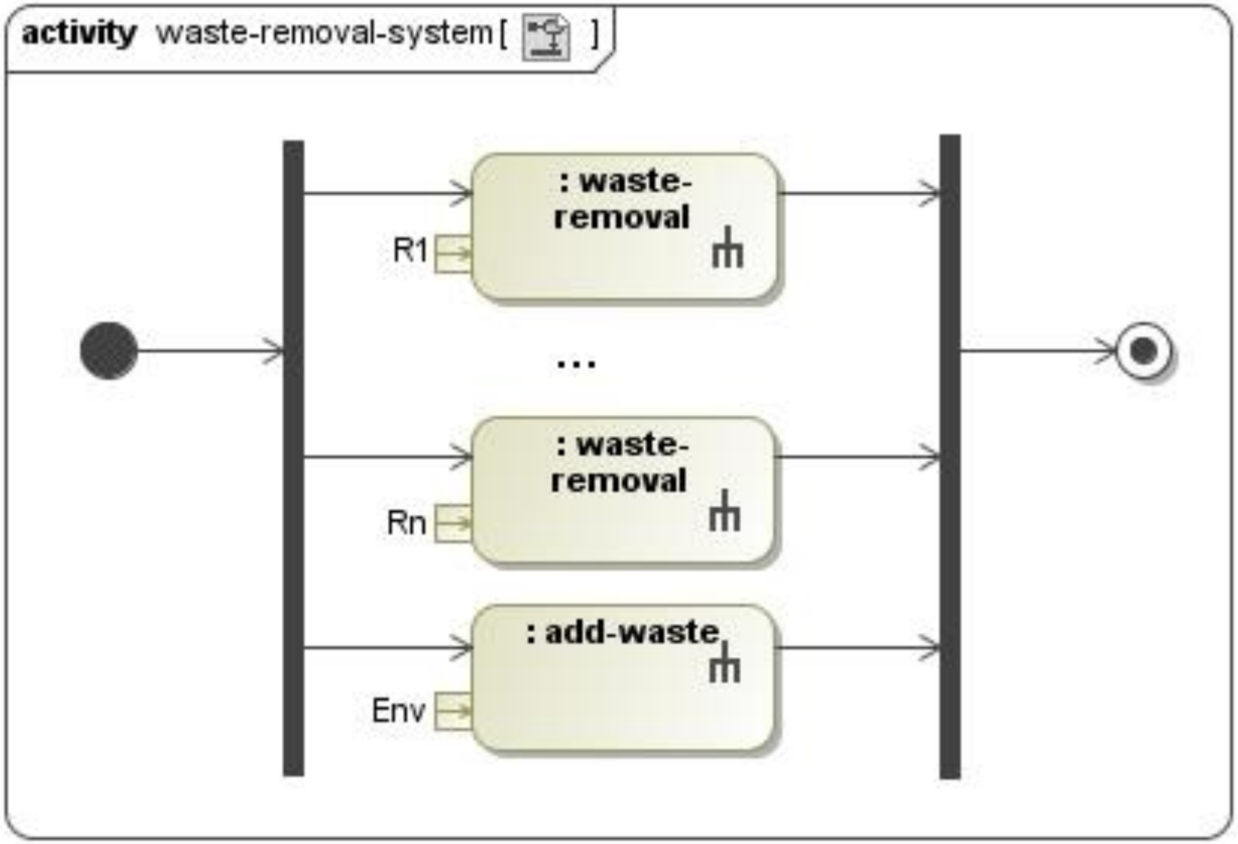


Deterministic axiomatization of effects of actions e.g.

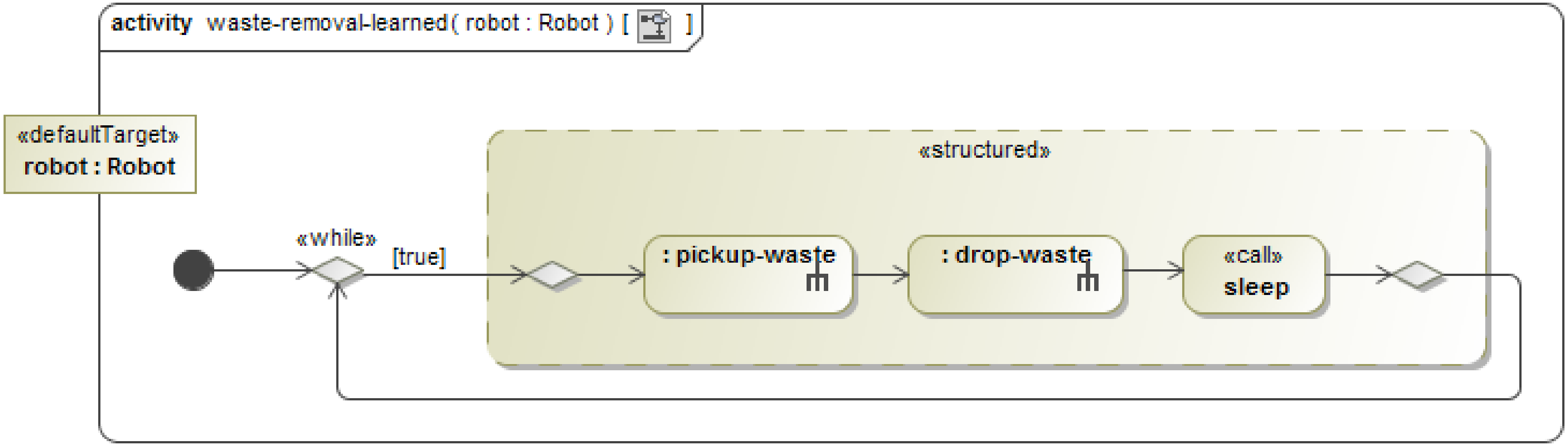
```
action Robot::stepNorth {
   pre: true;
   effects {
      self.position.y := self.position.y@pre + 1;
      }
}
```

precondition

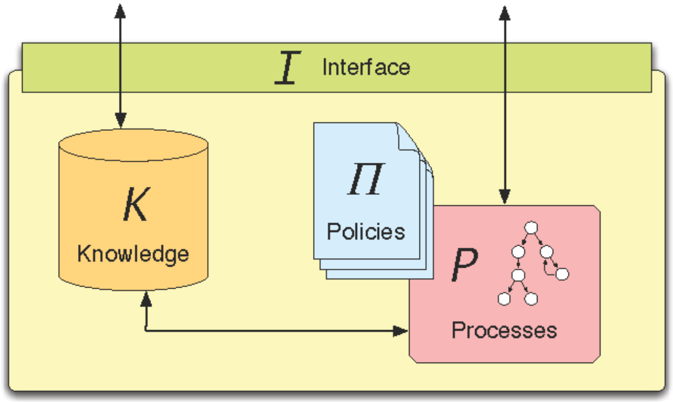
effect:
move one cell to north


activity waste-removal-system [ ]
: waste-removal
R1
...
: waste-removal
Rn
: add-waste
Env

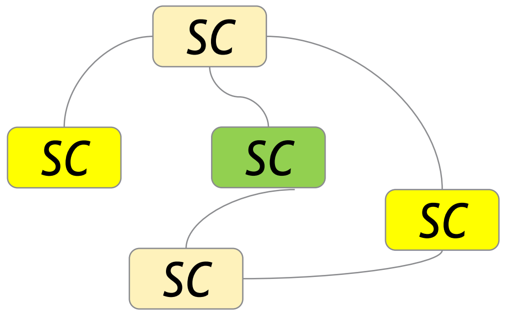
Helena Model: Robot Behaviour



- The **Service Component Ensemble Language** (SCEL) provides an abstract ensemble programming framework by offering primitives and constructs for the following programming abstractions
  - **Knowledge**: describe how data, information and knowledge is manipulated and shared (“tuple space”; put, get)
  - **Processes**: describe how systems of components progress
  - **Policies**: deal with the way properties of computations are represented and enforced
  - **Systems**: describe how different entities are brought together to form components, systems and, possibly, ensembles



Service component



Service component ensemble

The SCEL Syntax (in one slide)

- SCEL
  - Parametrized by the (distributed) knowledge tuple space and policies
  - Predicate-based communication
  - Processes interact with the tuple space by query and put actions

SYSTEMS:

$$S \quad ::= \quad C \quad | \quad S_1 \parallel S_2 \quad | \quad (\nu n)S$$

COMPONENTS:

$$C \quad ::= \quad \mathcal{I}[\mathcal{K}, \Pi, P]$$

PROCESSES:

$$P \quad ::= \quad \mathbf{nil} \quad | \quad a.P \quad | \quad P_1 + P_2 \quad | \quad P_1[\, P_2\,] \quad | \quad X \quad | \quad A(\bar{p})$$

ACTIONS:

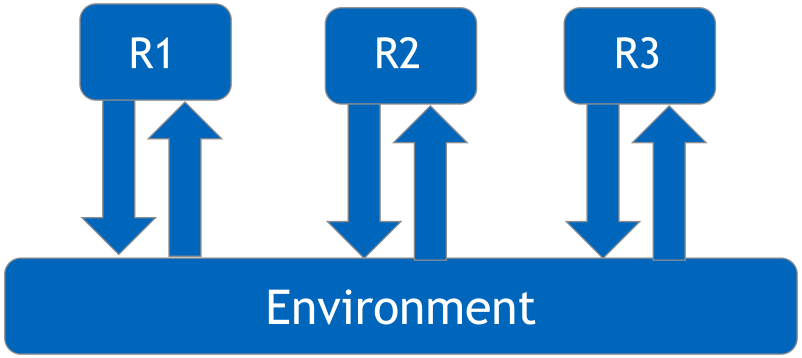
$$a \quad ::= \quad \mathbf{get}(T)@c \quad | \quad \mathbf{qry}(T)@c \quad | \quad \mathbf{put}(t)@c \quad | \quad \mathbf{fresh}(n) \quad | \quad \mathbf{new}(\mathcal{I}, \mathcal{K}, \Pi, P)$$

TARGETS:

$$c \quad ::= \quad n \quad | \quad x \quad | \quad \mathsf{self} \quad | \quad \mathsf{P} \quad | \quad \mathcal{I}.\mathsf{p}$$

Robot Ensemble SCEL Design

- Environment mediated robot ensemble



- *n* robots R_i interacting with environment Env and other robots

$$R_1 \;||\; \ldots \;||\; R_n \;||\; Env$$

- Env is abstractly represented by a component

$$I_{env}[.,., m]$$

keeping track of the total number of collected items

Robot Ensemble SCEL Refinement

- Each robot R_i is of form

$$R_i = I[.,.,\ explore[\ col[\ t]]\ ]$$

where

  - `explore` monitors the reactive robot behavior (searching for waste)
  - `col` detects collisions,
  - `t` controls the sleeping time

Parallel processes

Robot Behaviour

ascens

- E.g. monitoring the reactive behavior `explore` of a robot R_i for performance analysis
  - If R_i is exploring for picking up waste then
    - if it encounters another robot or a wall, it changes direction and continues exploring ("normal" moves and direction change abstracted in SCEL)
    - if it encounters an item, the robot picks it up (abstracted in SCEL), informs the environment `env` and starts returning to the service area

```
explore = get(collision)@self.explore + get(item)@self.pick
pick = get(items,!x)@env.pick'
pick' = put(items,x+1)@env.return
```

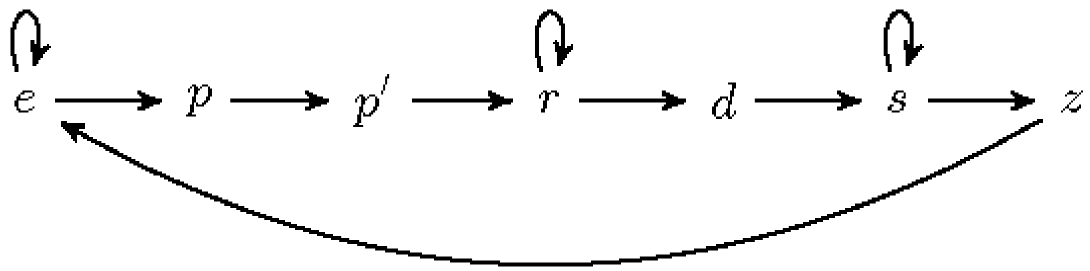
. . .

Validating Requirements: Quantitative Analysis

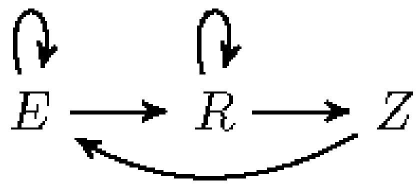
- Validating the adaptation requirements includes the following steps:
  - Ensemble simulation
    - jRESP, MISSCEL, or SCELua
  - Study timing behaviour by abstracting SCEL models to
    - Continuous-time Markov chains
    - Ordinary differential equations
    - Statistical modelchecking
  - Validate performance model by comparing to simulation and
  - Validate the adaptation requirements by sensitivity analysis

Quantitative Analysis

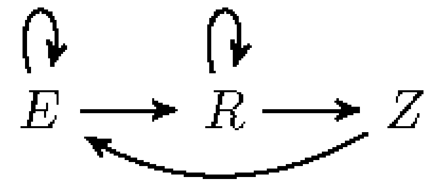
- Simplify robot behavior
  - From



  - To the (Helena) abstraction



Quantitative Analysis

- Derive continuous-time Markov chain from  :

$$
\begin{aligned}
(E,R,Z,F) &\longrightarrow (E-1,R+1,Z,F-1), && \text{with rate} \quad \mu E \frac{F}{E+R+F},\\
(E,R,Z,F) &\longrightarrow (E+1,R,Z-1,F), && \text{with rate} \quad \beta Z,\\
(E,R,Z,F) &\longrightarrow (E,R-1,Z+1,F), && \text{with rate} \quad \gamma R,\\
(E,R,Z,F) &\longrightarrow (E,R,Z,F+1), && \text{with rate} \quad \lambda.
\end{aligned}
$$

  - CTMC as infinitely many states

- Transform into ODE

$$
\begin{aligned}
\dot{E} &= -\mu E F (E+R+F)^{-1} + \beta Z\\
\dot{R} &= +\mu E F (E+R+F)^{-1} - \gamma R\\
\dot{Z} &= +\gamma R - \beta Z\\
\dot{F} &= +\lambda - \mu E F (E+R+F)^{-1}
\end{aligned}
$$

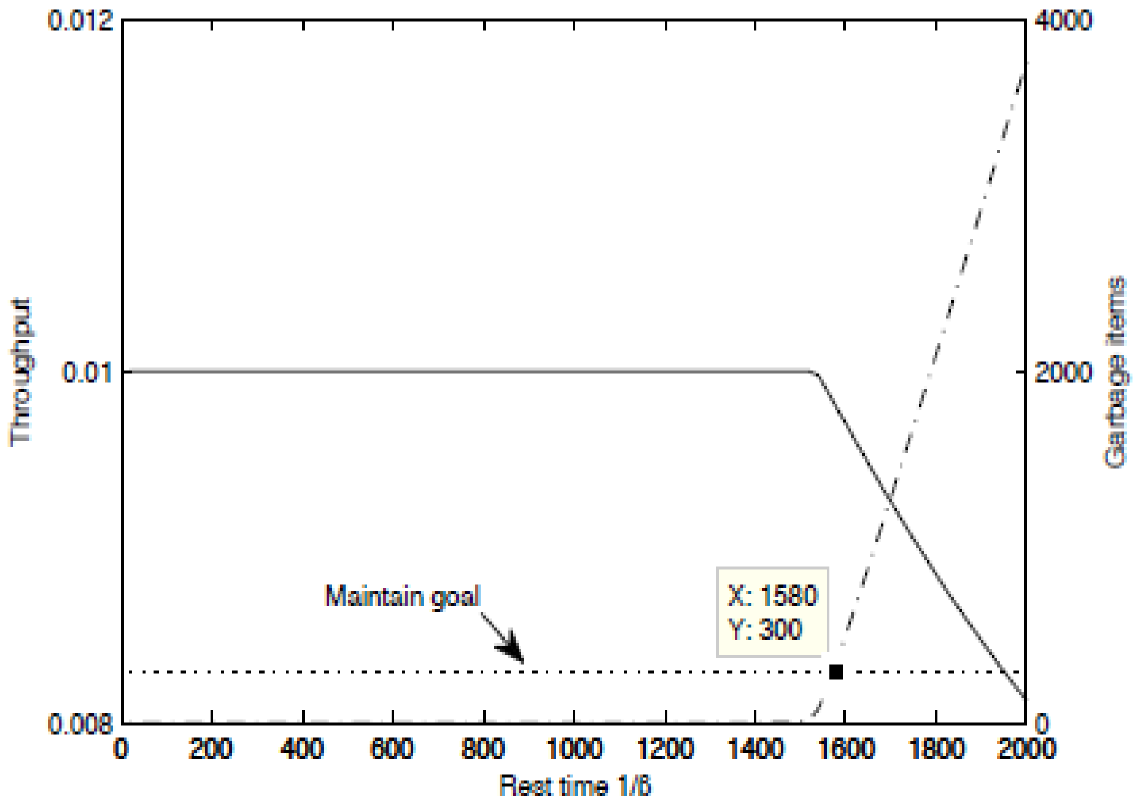
Quantitative Analysis

- SCELua simulation
  - SCELua is an experimental SCEL implementation in Lua/ARGOS [Hölzl 2012]
  - Simulate robot example
    - 20 robots, arena 16 m², 150 independent runs of 10 h simulated time
    - Instrument code to record timestamps of transitions and calculate μ and γ
- Compare
    - Steady state ODE estimates of robot subpopulations and
    - discrete-event LuaSCEL simulation
- Results

	E	R	S
Simulation	15.372	3.917	0.068
Model	16.070	3.730	0.200

    - Maximum error < 3.5%

- Adaptation requirements
  - Keep area clean (< 300 garbage items) while allowing sleeping time t (e.g. <= 1000) for each robot
  - Energy consumption lower than predefined threshold
- Sensitivity analysis of throughput
  - where throughput = frequency of returning garbage items to service area

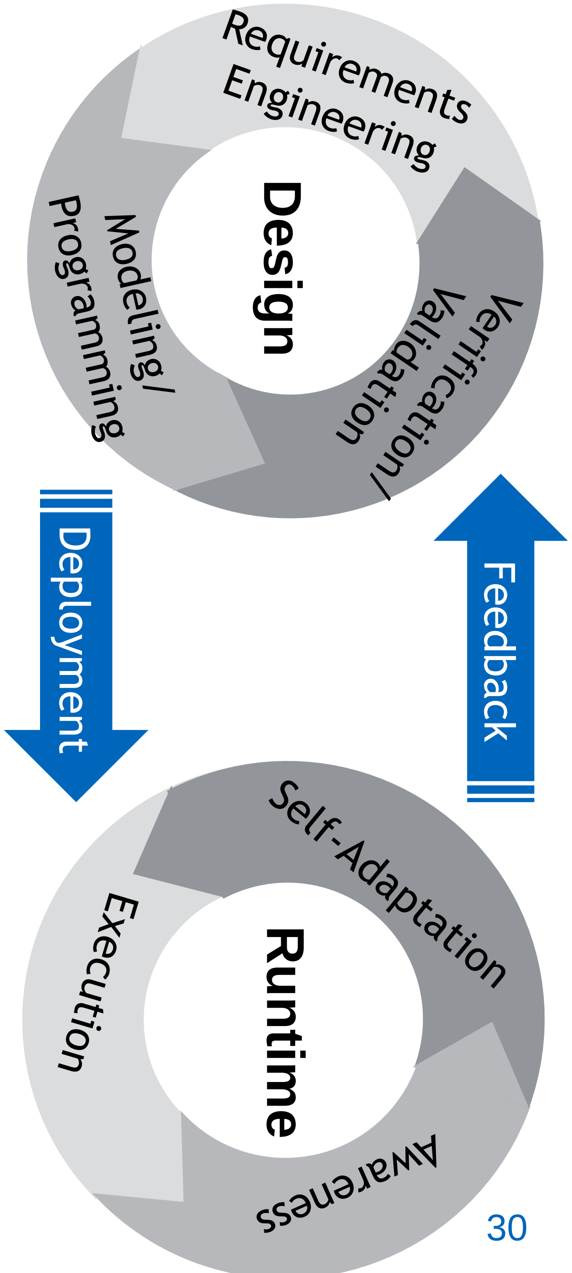


Model prediction:

- Adaptation requirement is satisfied
- Maximum allowed rest time (whilst achieving the maintain goal): 1580

Summary

- ASCENS is developing a systematic approach for constructing Autonomic Service-Component Ensembles
- A few development steps for a simple example
- More research needed for all development phases, in particular on
  - Modeling and formalising ensembles
  - Knowledge representation and self-awareness
  - Adaptation and dynamic self-expression patterns and mechanisms.
  - Correctness, verification, and security of ensembles
  - Tools and methodologies for designing and developing correct ensembles
  - Experimentations with case studies



References

- Rolf Hennicker and Annabelle Klarl. Foundations for Ensemble Modeling - The Helena Approach. Specification, Algebra, and Software: A Festschrift Symposium in Honor of Kokichi Futatsugi (SAS 2014) To Appear, April 2014.
- Martin Wirsing, Matthias Hölzl, Mirco Tribastone, and Franco Zambonelli. ASCENS: Engineering Autonomic Service-Component Ensembles. In Bernhard Beckert, Ferruccio Damiani, Marcello Bonsangue, and Frank de Boer, editors, *Formal Methods for Components and Objects, 10th International Symposium, FMCO 2011*, LNCS. Springer, 2012.
- Giacomo Cabri, Mariachiara Puviani, and Franco Zambonelli. Towards a Taxonomy of Adaptive Agent-based Collaboration Patterns for Autonomic Service Ensembles. In *2011 International Conference on Collaboration Technologies and Systems*, pages 508–515, Philadelphia (PA), May 2011. IEEE Press.