Preconditions

OC terminology

Objectives

Architectural variants for Organic Computing systems

Content

- Autonomic Computing: MAPE
- Observer/Controller Architecture
- Architectural options
- Examples of OC systems and their architectures
  - Organic Traffic Control
  - Others

Preconditions

OC terminology

Objectives

Architectural options for OC systems

Content

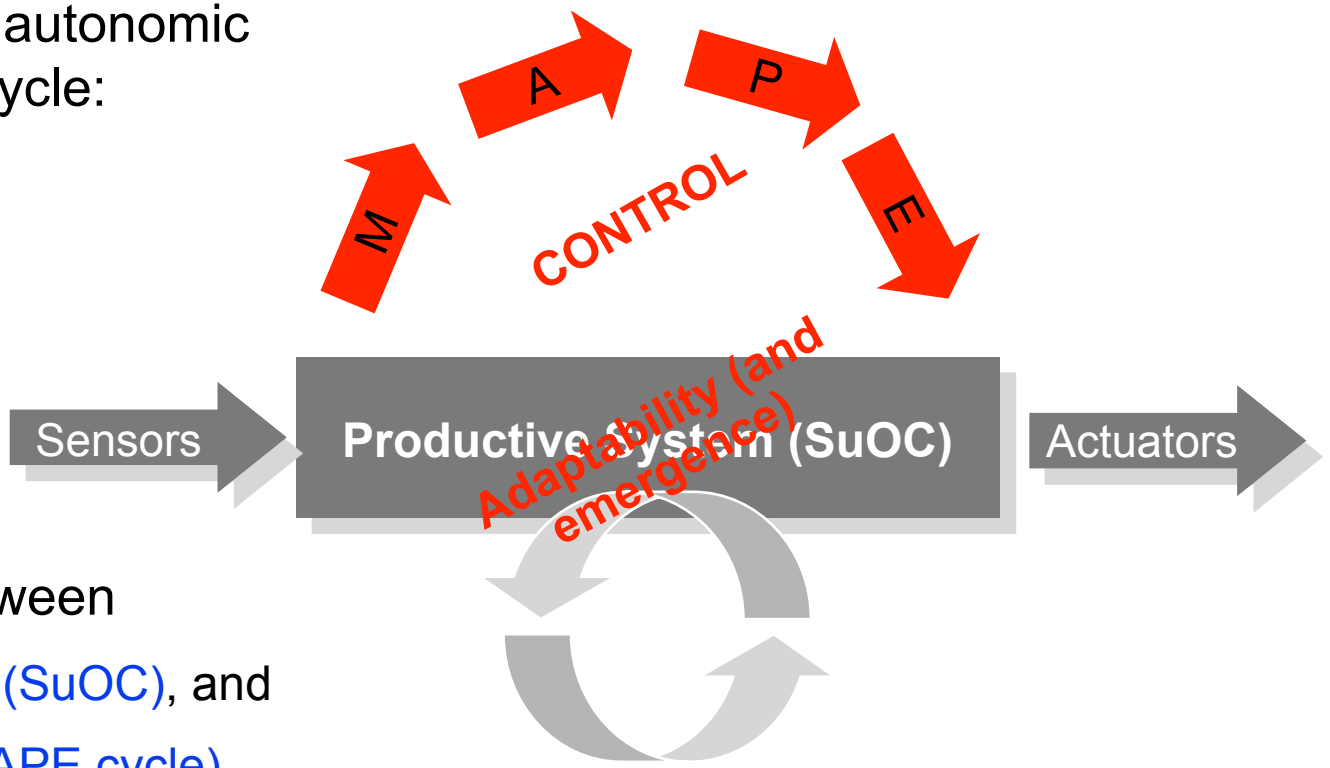
- ❑ Autonomic Computing: MAPE
- ❑ Observer/Controller Architecture
- ❑ Architectural options
- ❑ Examples for OC systems and their architectures

- Autonomic Computing (AC) is some kind of predecessor for OC:
  - In ‘The Vision of AC‘, IBM warns that the dream of interconnectivity of computing systems and devices could become the “nightmare of pervasive computing” in which architects are unable to anticipate, design and maintain the complexity of interactions. They state the essence of AC is system self-management, freeing administrators from low-level task management while delivering more optimal system behaviour.
  - The focus of AC is set on building an artificial autonomic nervous system for e.g. large server farms.

See also: http://www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf

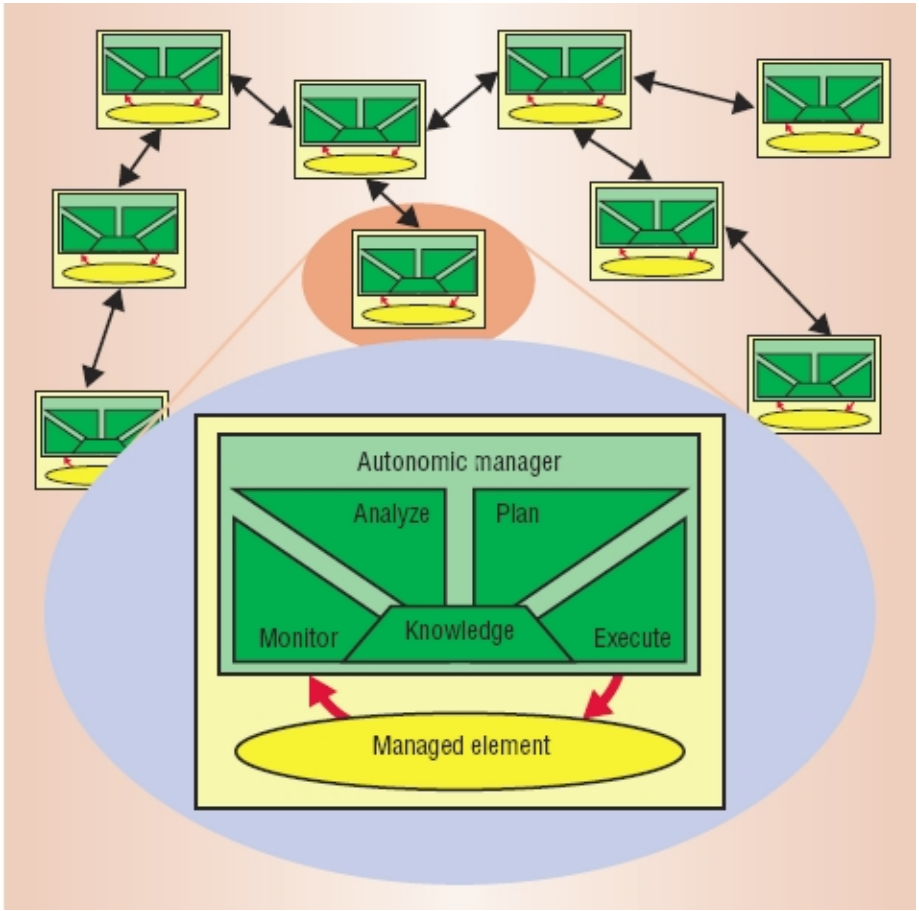
- AC tries to control complexity using principles, like e.g.:
  - Distribution of responsibilities from central entities to local ones.
  - Local goals, decentralised control, (large) populations
  - Self-* properties (for AC: Self-CHOP)
- Self-CHOP
  - Self-configuring (Automatic configuration of components).
  - Self-healing (Automatic discovery, and correction of faults).
  - Self-optimising (Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements).
  - Self-protecting (Proactive identification and protection from arbitrary attacks).
- Ever heard before?
  - Yes, indeed! Approach is similar to Organic Computing.
  - OC has a broader focus: AC can be seen as subset of OC.
  - And: IBM seems to have lost interest in AC. :-(

- Basic element of an autonomic entity is the MAPE cycle:
  - Monitor
  - Analyse
  - Plan
  - Execute



- Clear distinction between
  - Productive system (SuOC), and
  - Control system (MAPE cycle)

Multiple MAPE cycles



Preconditions

OC terminology

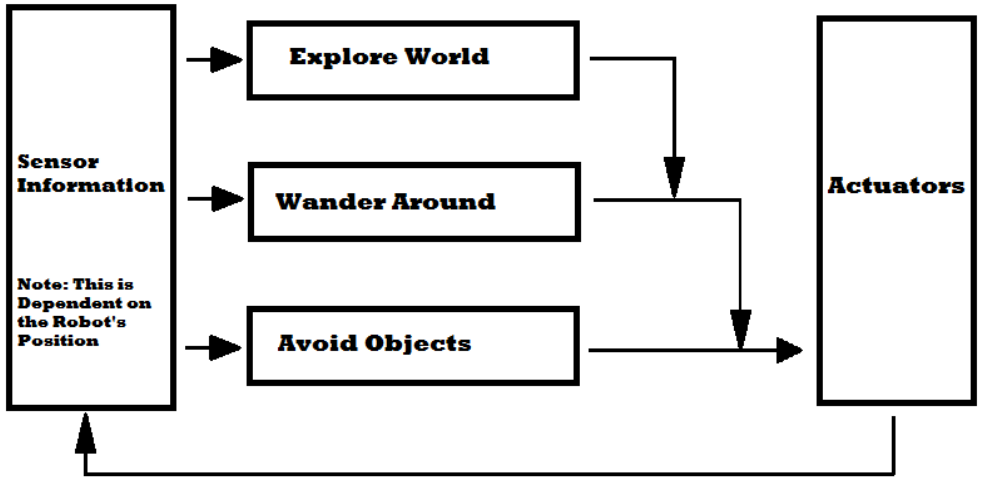
Objectives

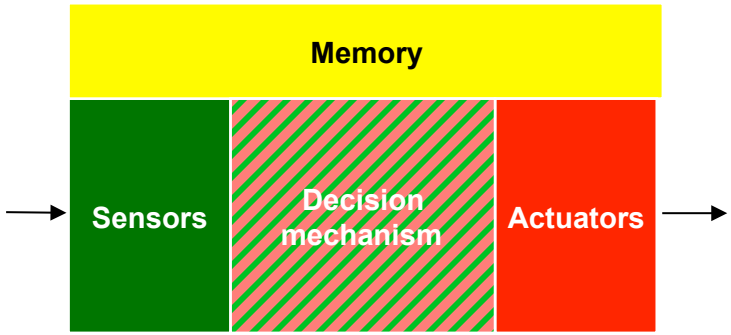
Architectural options for OC systems

Content

- Autonomic Computing: MAPE
- Observer/Controller Architecture
- Architectural options
- Examples for OC systems and their architectures

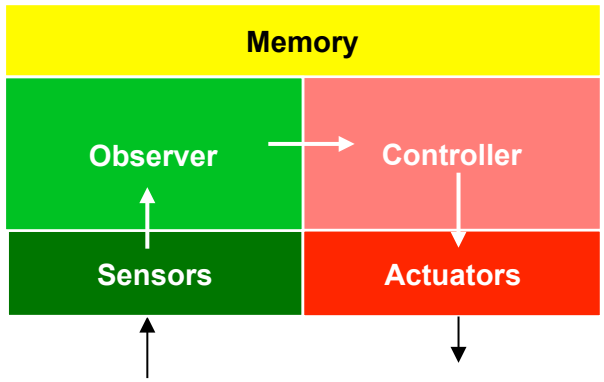
- Rodney Brooks‘ animats (subsumption architecture):



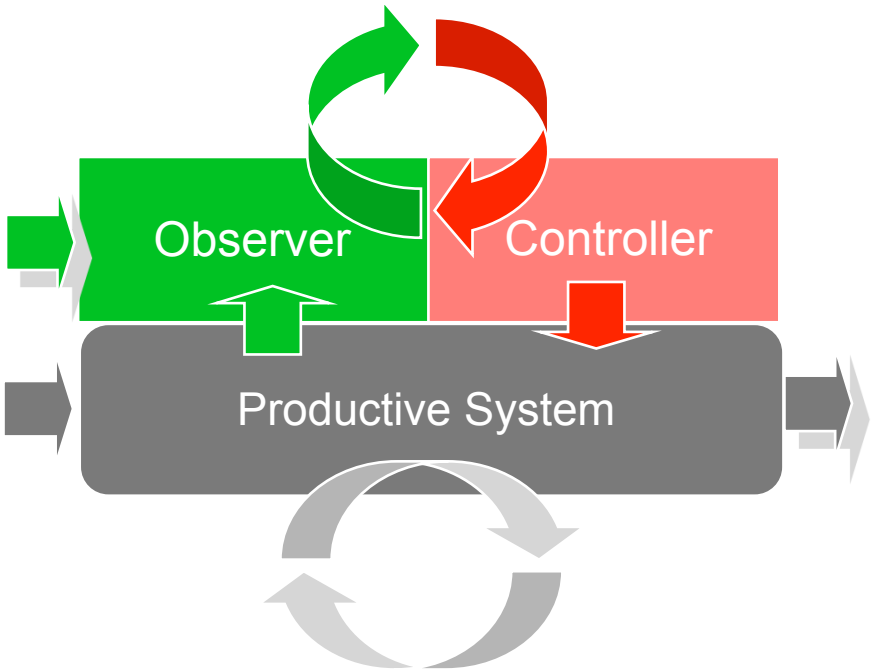


Observer/Controller architecture (1)

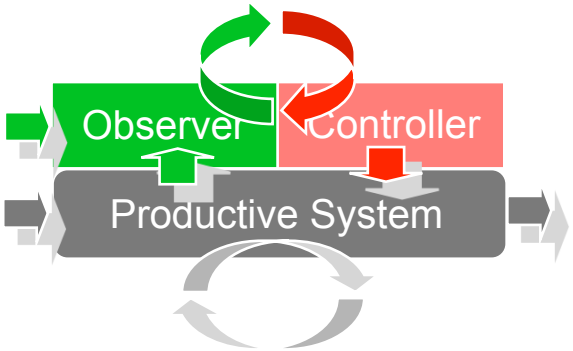
- Autonomous system:

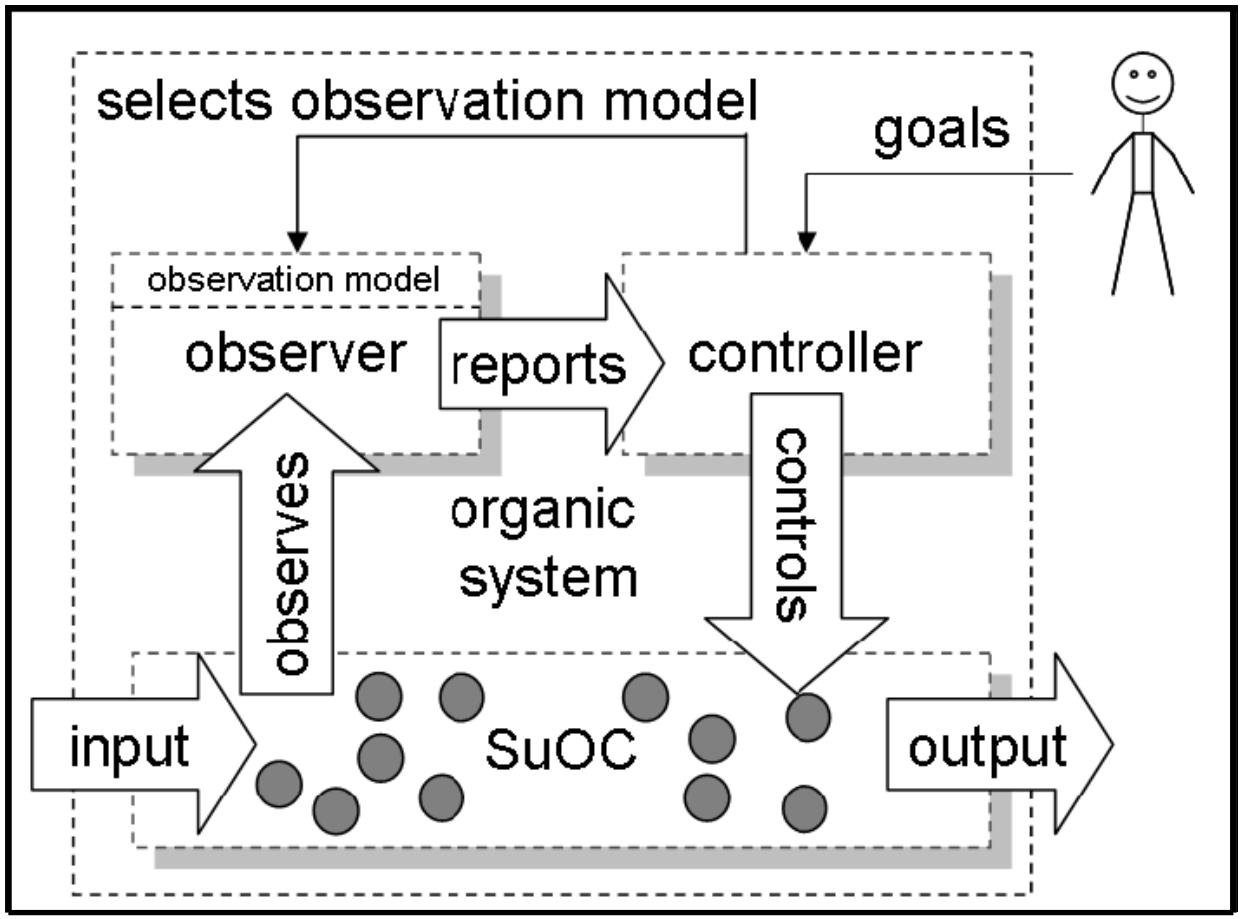


- O/C architecture:

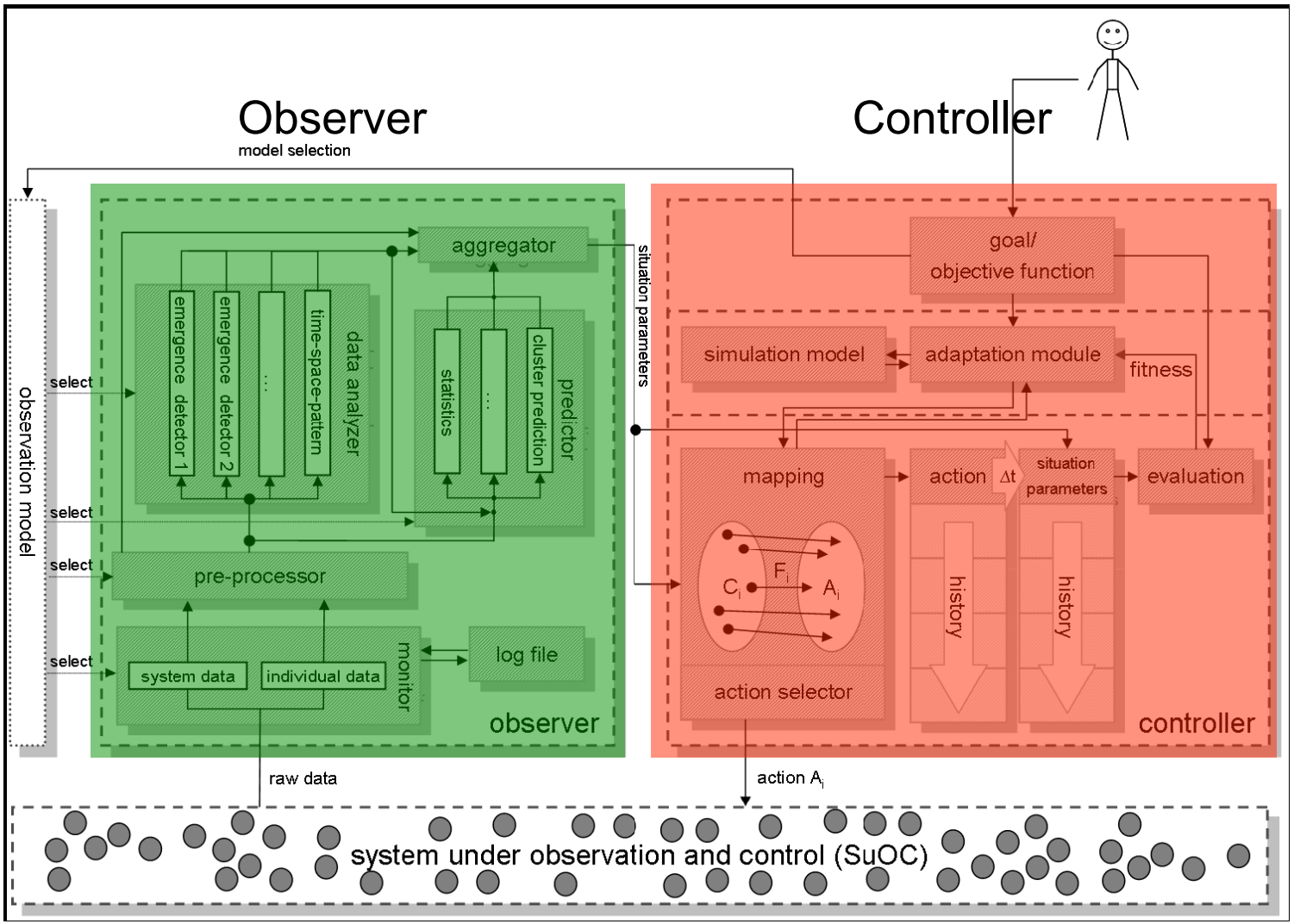


Observer/Controller architecture (2)





Observer/Controller architecture (3)



- The Observer

Observer

  - has various detectors to monitor the SuOC (e.g. an emergence detector),
  - uses an observation model, which determines
    - which data are monitored
    - how they are preprocessed
    - which detectors are activated.
  - The attribute observations (basic or derived) are provided by a monitor and a preprocessing stage.

- The Controller

Controller

  - has objectives/goals given by the user,
  - classifies the system descriptor according to these objectives,
  - evaluates the SuOC's and consequently its own performance (→ learning),
  - can influence the SuOC by control actions,
  - provides a history / memory function (for learning purposes).

Preconditions

OC terminology

Objectives

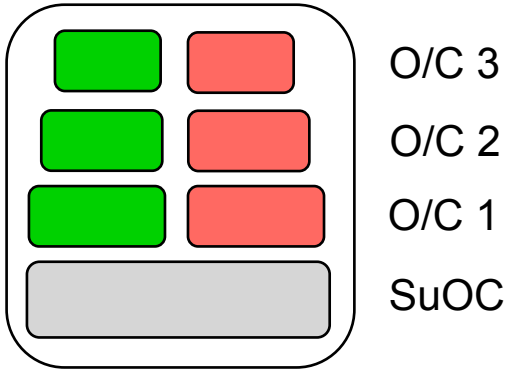
Architectural options for OC systems

Content

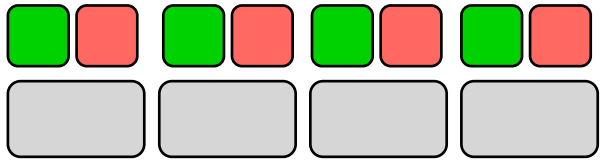
- Autonomic Computing: MAPE
- Observer/Controller Architecture
- Architectural options
- Examples for OC systems and their architectures
- Extensions: Social OC
- Design time to runtime

Observer/Controller architectures

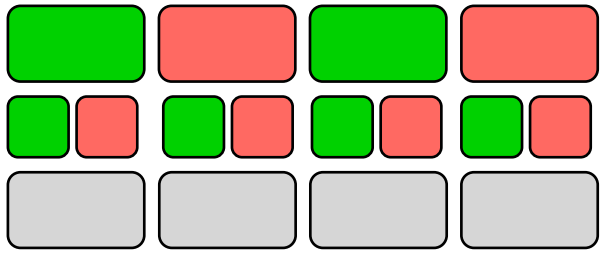
- Architectural options
  1) Hierarchy
     - Multiple layers of O/Cs
     - Decreasing level of detail



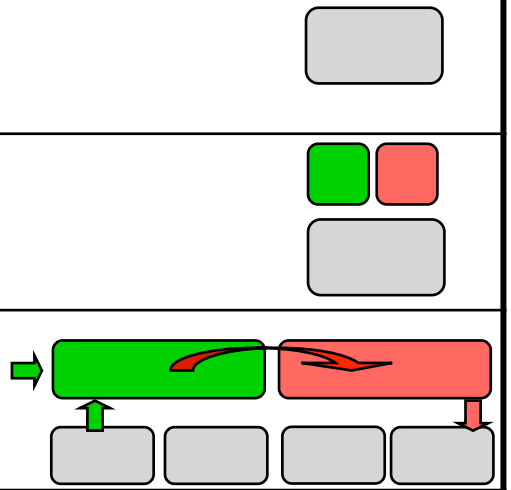
  2) Span: Multiple subsystems
     - Subsystems are autonomous
     - Coordination?

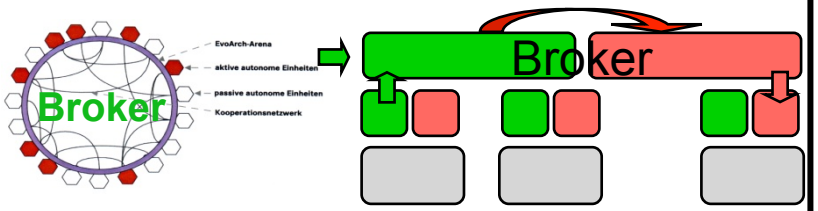


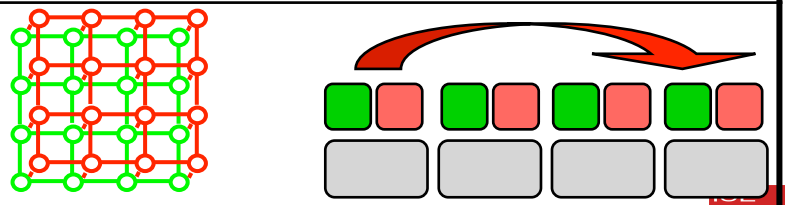
  3) Combinations

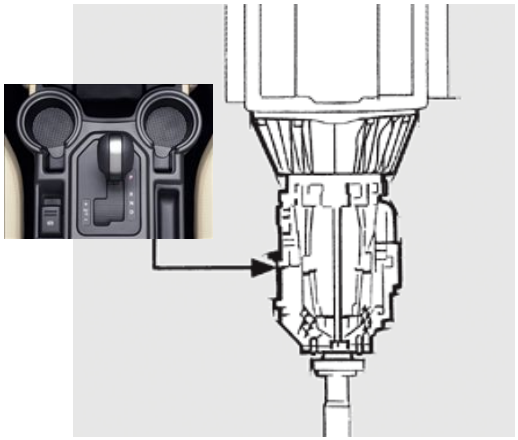


	SO type	
0	Rigid	No adaptability, no external control
1	Multi-modal	Single modes, isolated functions, small configuration space
2	Reconfigurable/ parameterizable	Large configuration space, networked components
3	Multilevel system	Active agents, mediated cooperation
4	Distributed cooperative self- organisation	Autonomous agents, w/o central control









Adaptive gear box (AGS)

The Adaptive gear box determines the optimal gear for the automatic modus by taking several factors into account. For the detection of the style of driving, position and actuation of the accelerator are continuously monitored. Slip-impulse and moment-impulse are compared and based on this the switching operation is adapted to the street conditions („normal“, „winter“, or „mountain/start“).

Freude am Fahren

Observer

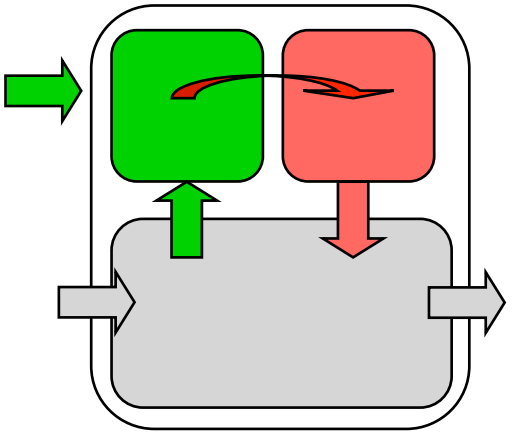
- Monitor, data collector
- Classification

Controller

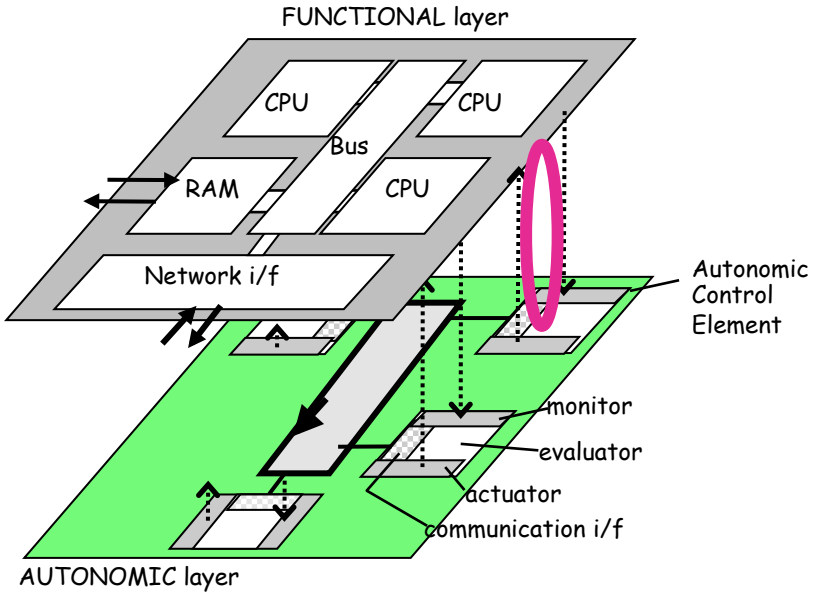
- Modus selector
- Selects economy or sport

Productive system

- Adaptable gearbox
- Sensors for driver and engine behaviour
- Modes: economy, sport, etc.



Example: Reconfigurable SoC



Observer

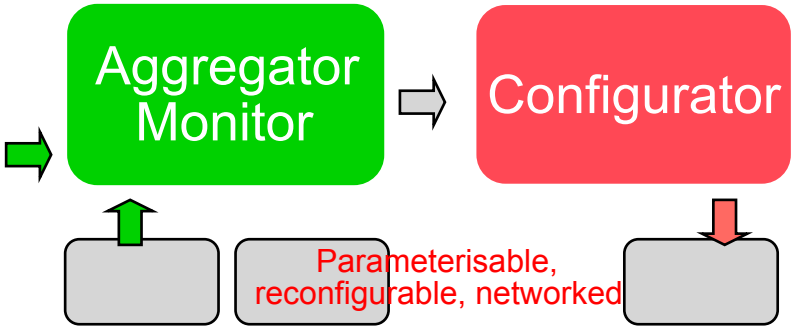
- Monitor + aggregator
- collects load data, determines load situation

Controller

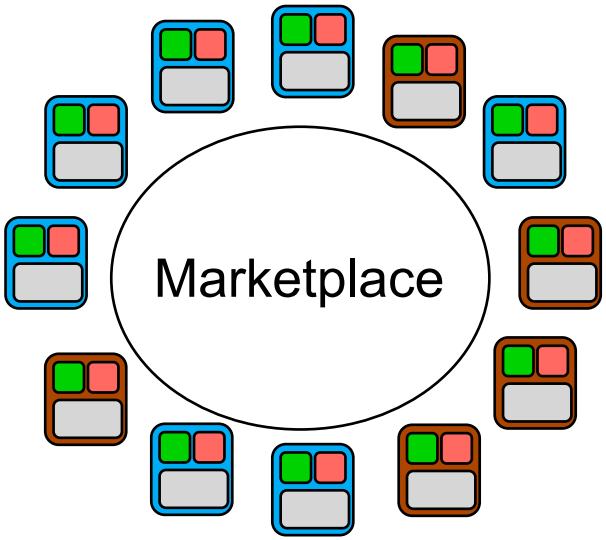
- Configurator
- Assigns additional ressources (CPU, bus bandwidth)

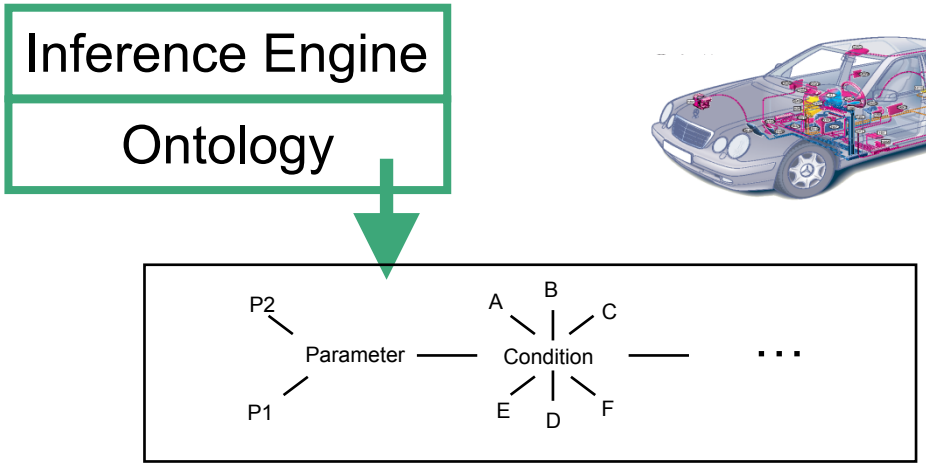
Productive system

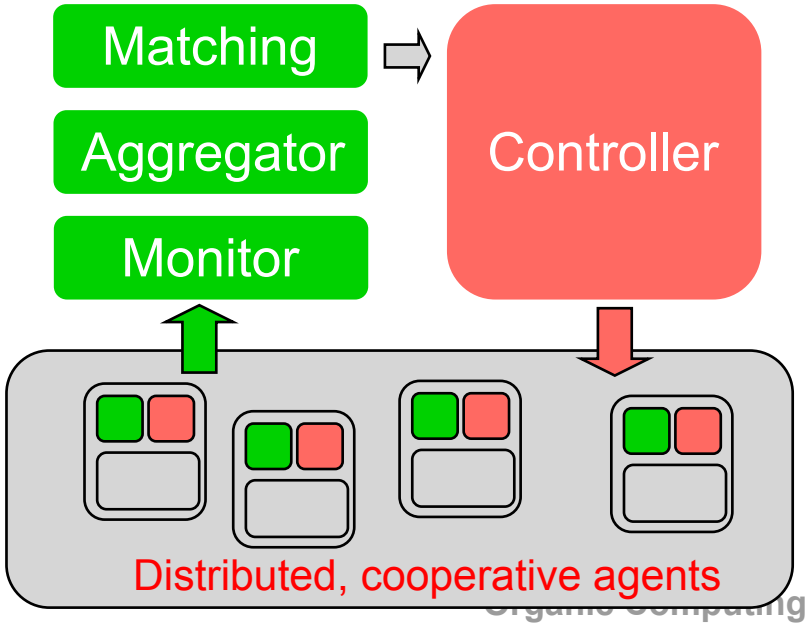
- Reconfigurable integrated circuit (System on Chip / SoC)
- Sensor equipped



Example Multilevel (Brokered) system







Observer/Controller: Broker

- Broker: central matching service
- Agents: active, passive
- Assessment of solutions by Multi-Criteria Decision Making (MCDM)

Preconditions

OC terminology

Objectives

Architectural options for OC systems

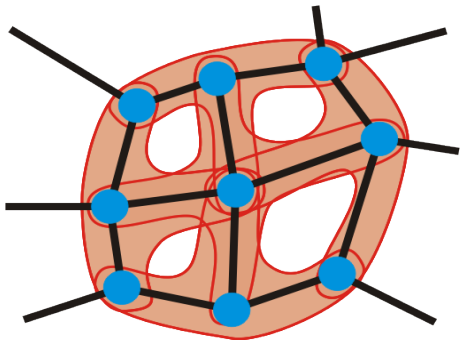
Content

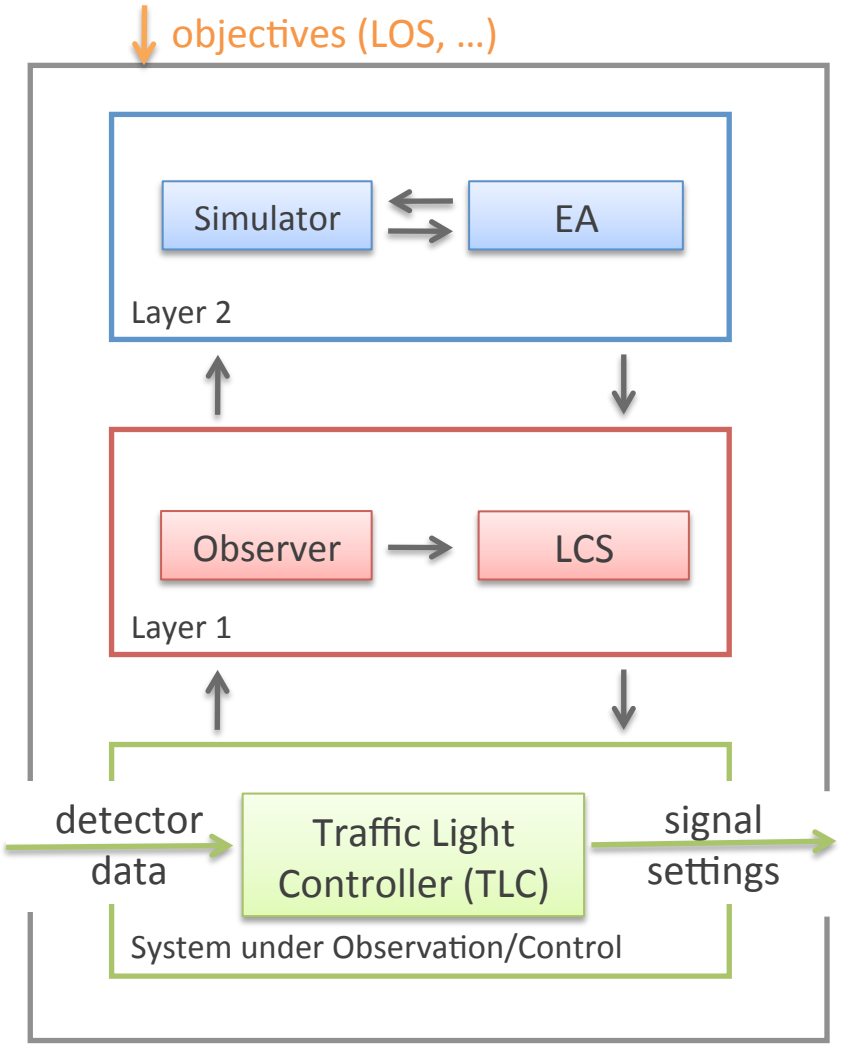
- Autonomic Computing: MAPE
- Observer/Controller Architecture
- Architectural options
- Examples for OC systems and their architectures
  - Organic Traffic Control
  - Others
  - Holonic agents

Goals

- Study a network of adaptive learning traffic light controllers (TLCs).

1. TLCs learn with some limited sensory horizon.
2. TLCs cooperate to achieve a global goal (e.g. reduced avg. travel time).
   - Self-organized Progressive Signal System (Grüne Welle)
   - Self-organized route guidance





User

Definition of system objectives

Layer 2

Off-line parameter optimisation

- Evolutionary Algorithm (EA) evolves TLC parameters
- Simulation-based evaluation

Layer 1

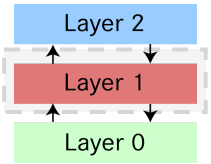
On-line parameter selection

- Observer monitors traffic
- Learning Classifier System (LCS) selects TLC parameters and learns rule quality

SuOC

Control of traffic signals

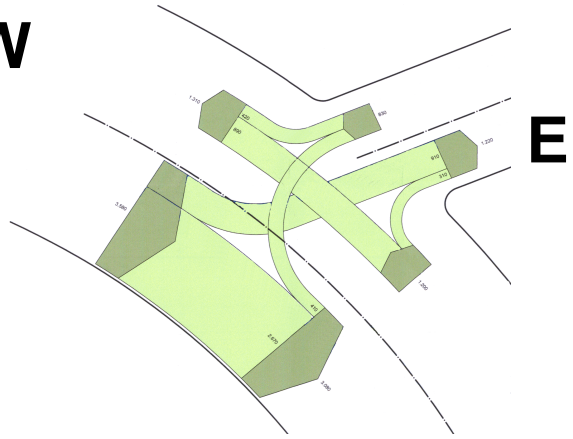
- Industry-standard TLC
  - Fixed-time
  - Traffic-responsive
  - Parameters determine performance



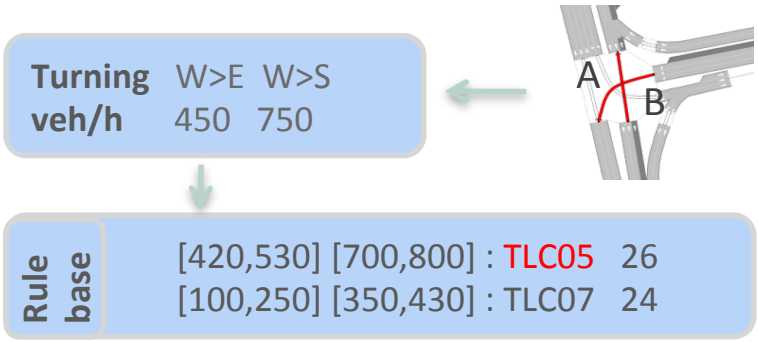
1. Observer determines traffic situation

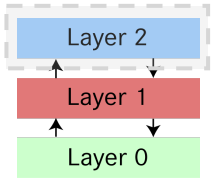
- Available data: Demand on approaches, average speed, waiting times, up- and downstream information
- Numbers combined: “Traffic situation” as input for controller

W→E	W→S	E→W	
910	2670	420	…



2. Controller selects action (parameter set for TLC)





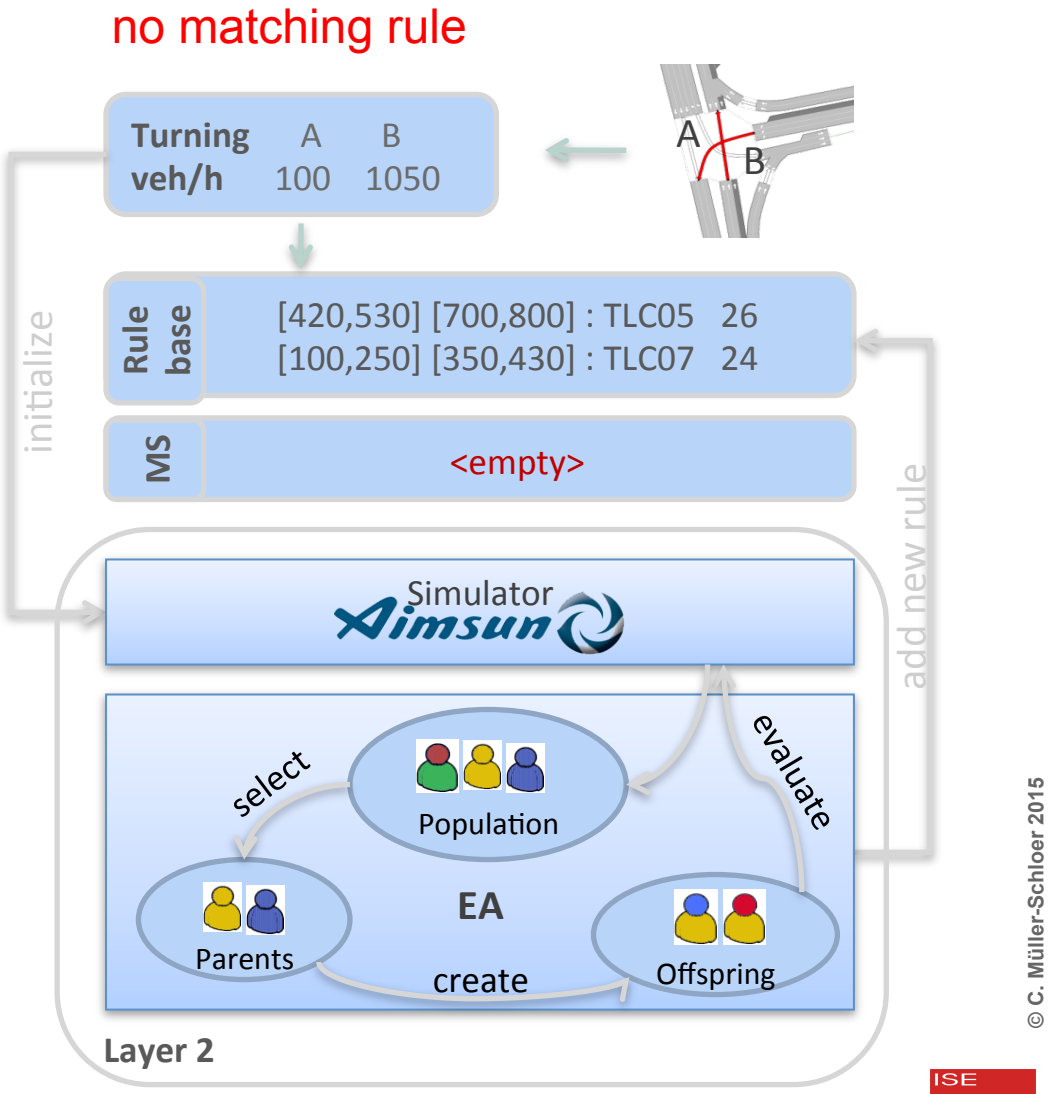
Layer 2: Off-line generation of new parameters

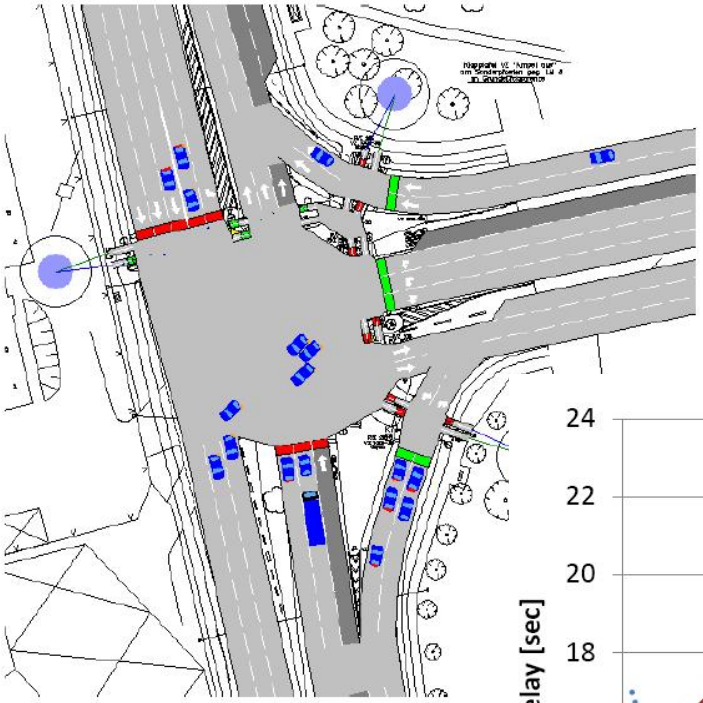
Rule creation in “classical” LCS

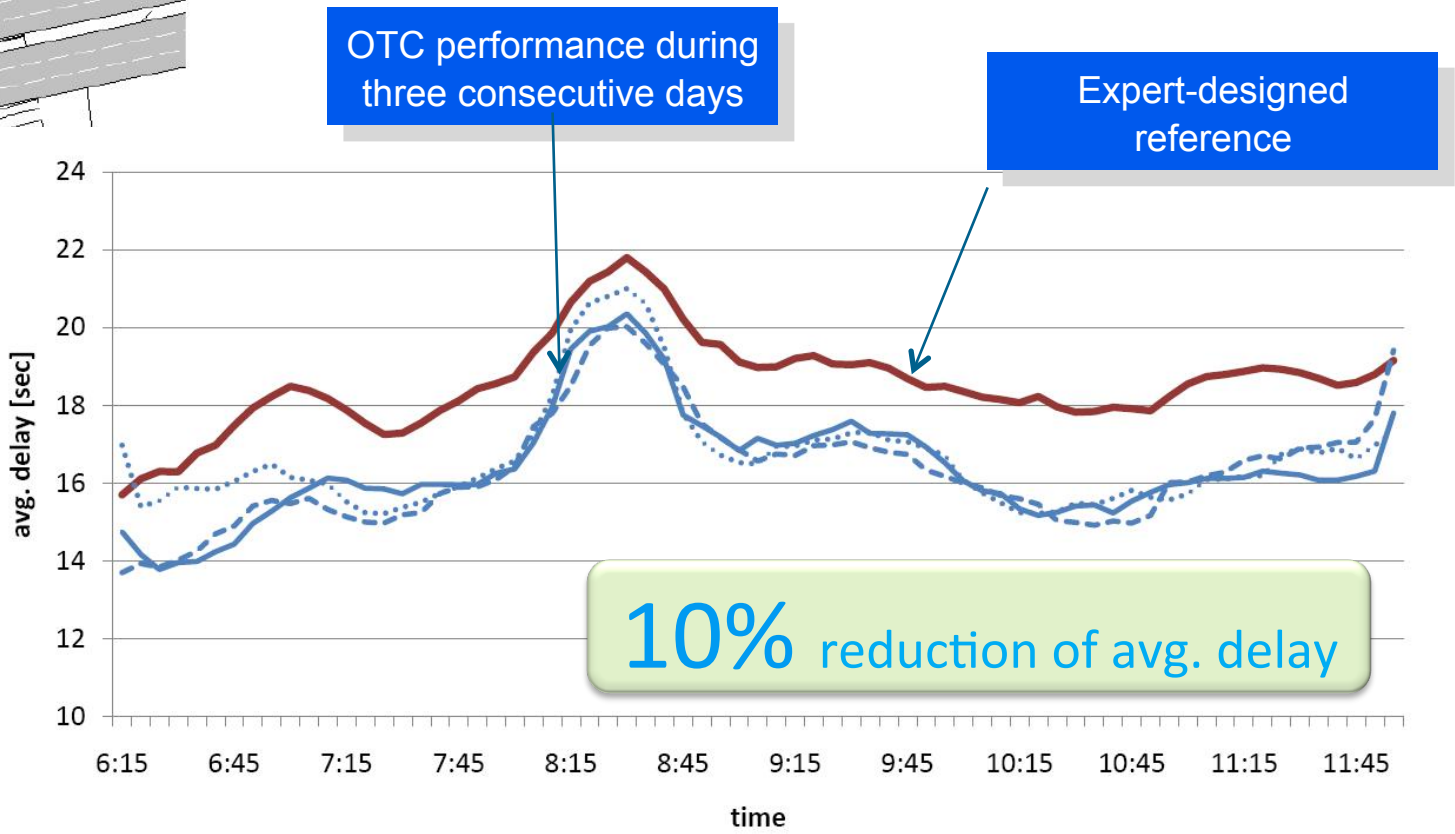
- ❑ Covering
  (Create matching rules randomly)
- ❑ Random crossover / mutation
  of existing rules
- ❑ Environment used for evaluation

… in OTC

- ❑ EA optimizes TLC parameters
  for observed situation
- ❑ Simulation-based evaluation





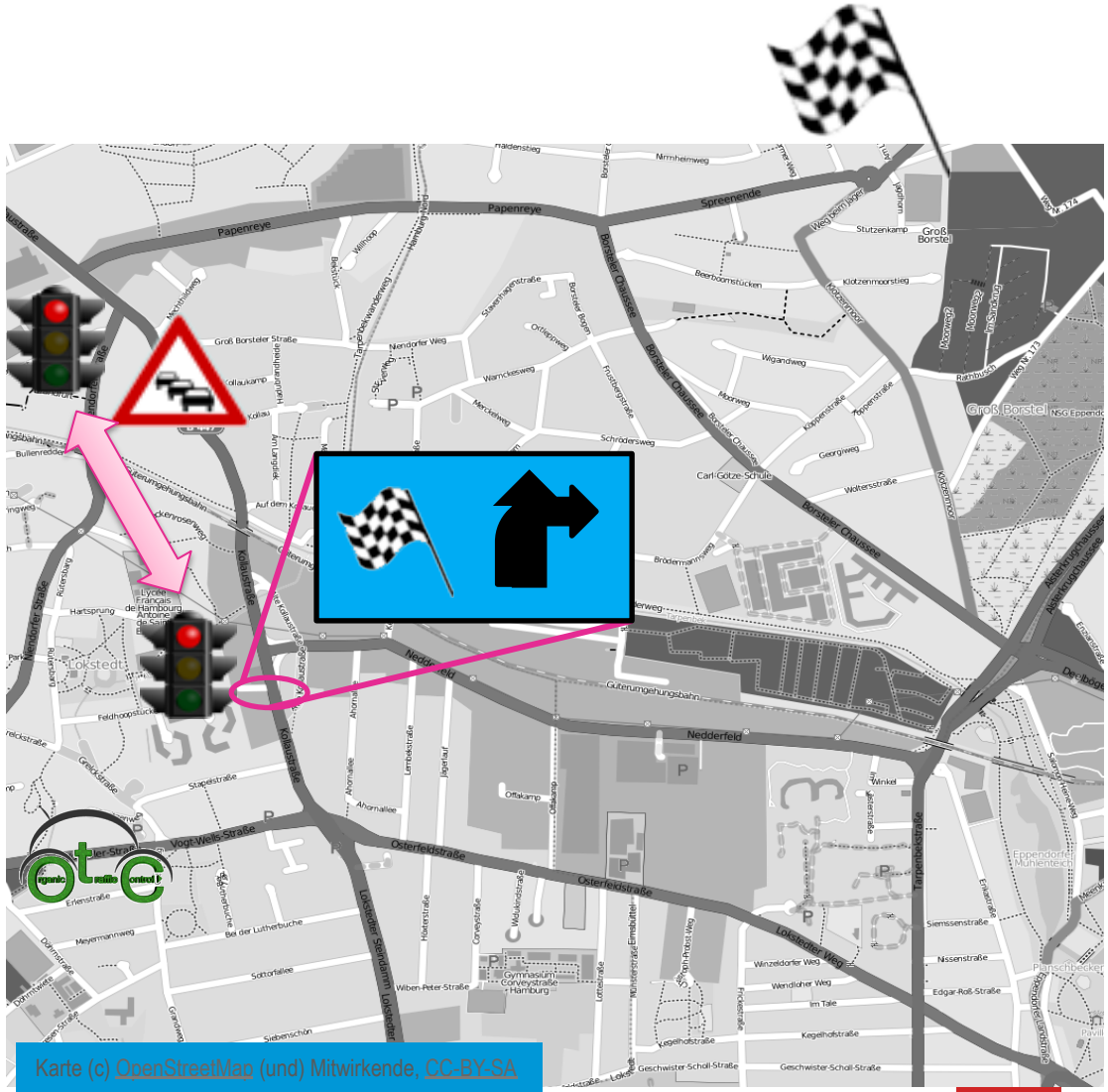

OTC performance during three consecutive days
Expert-designed reference
avg. delay [sec]
24
22
20
18
16
14
12
10
6:15
6:45
7:15
7:45
8:15
8:45
9:15
9:45
10:15
10:45
11:15
11:45
time
10% reduction of avg. delay

Route guidance and driver information

- ❑ Variable Message Signs:

  No car-to-x communication required

- ❑ Communicating traffic lights

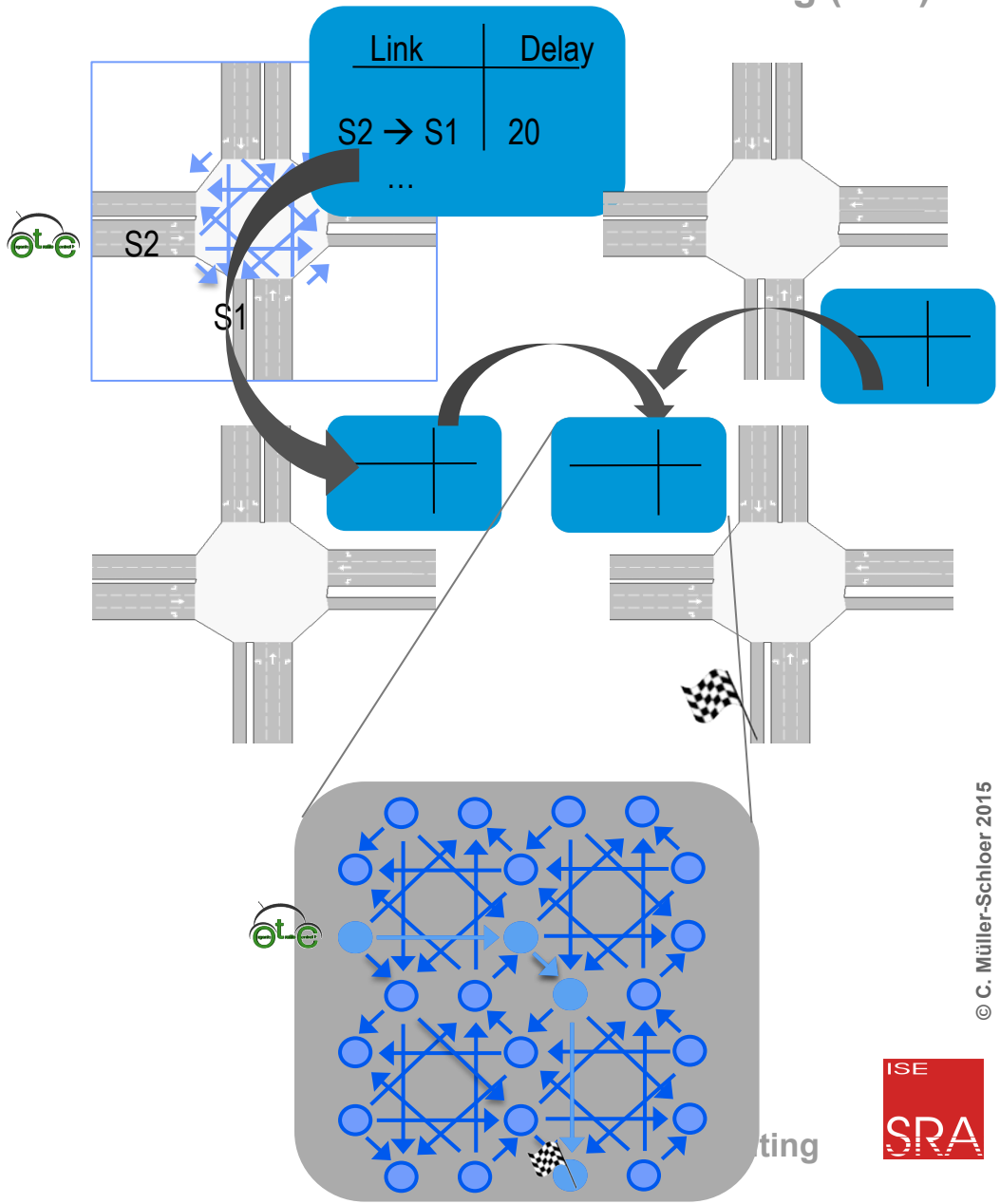


Organic Traffic Control

Link state routing (LST)

Routing components

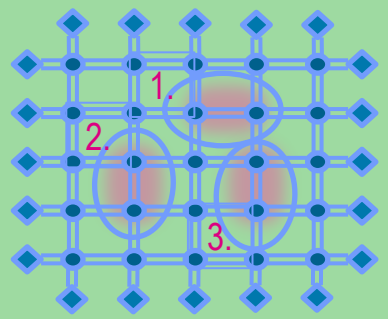
1. Determine turning delays
2. Communicate delay changes
   - Link state advertisement
   - Network flooding
3. Create weighted network graph from received link states
4. Compute shortest paths using Dijkstra's algorithm



http://de.wikipedia.org/wiki/Dijkstra-Algorithmus

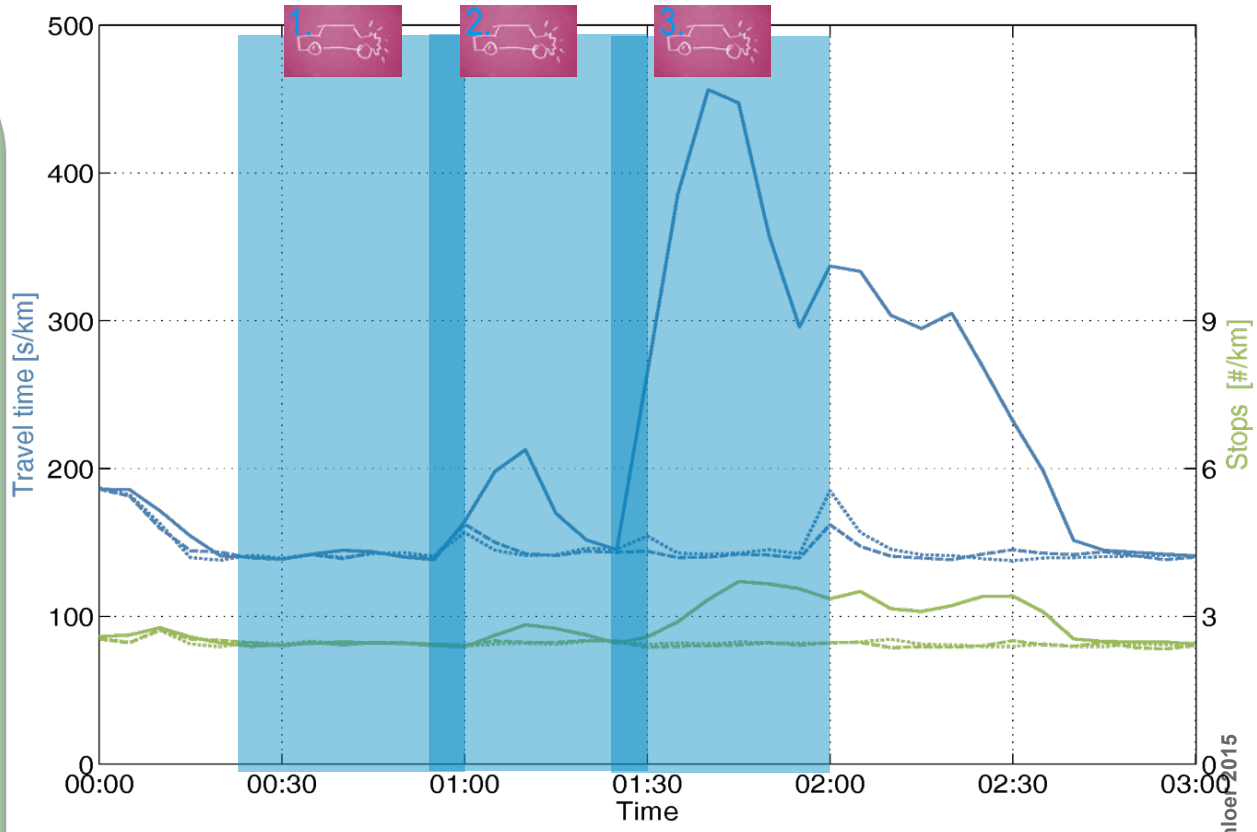
Internet-analog routing: Experimental evaluation

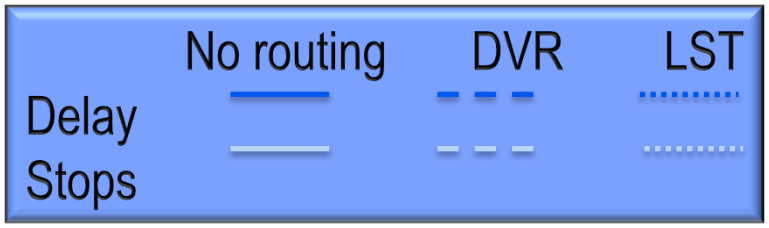
- 75% compliance rate
- Undersaturated demand
- 3 blocked roads



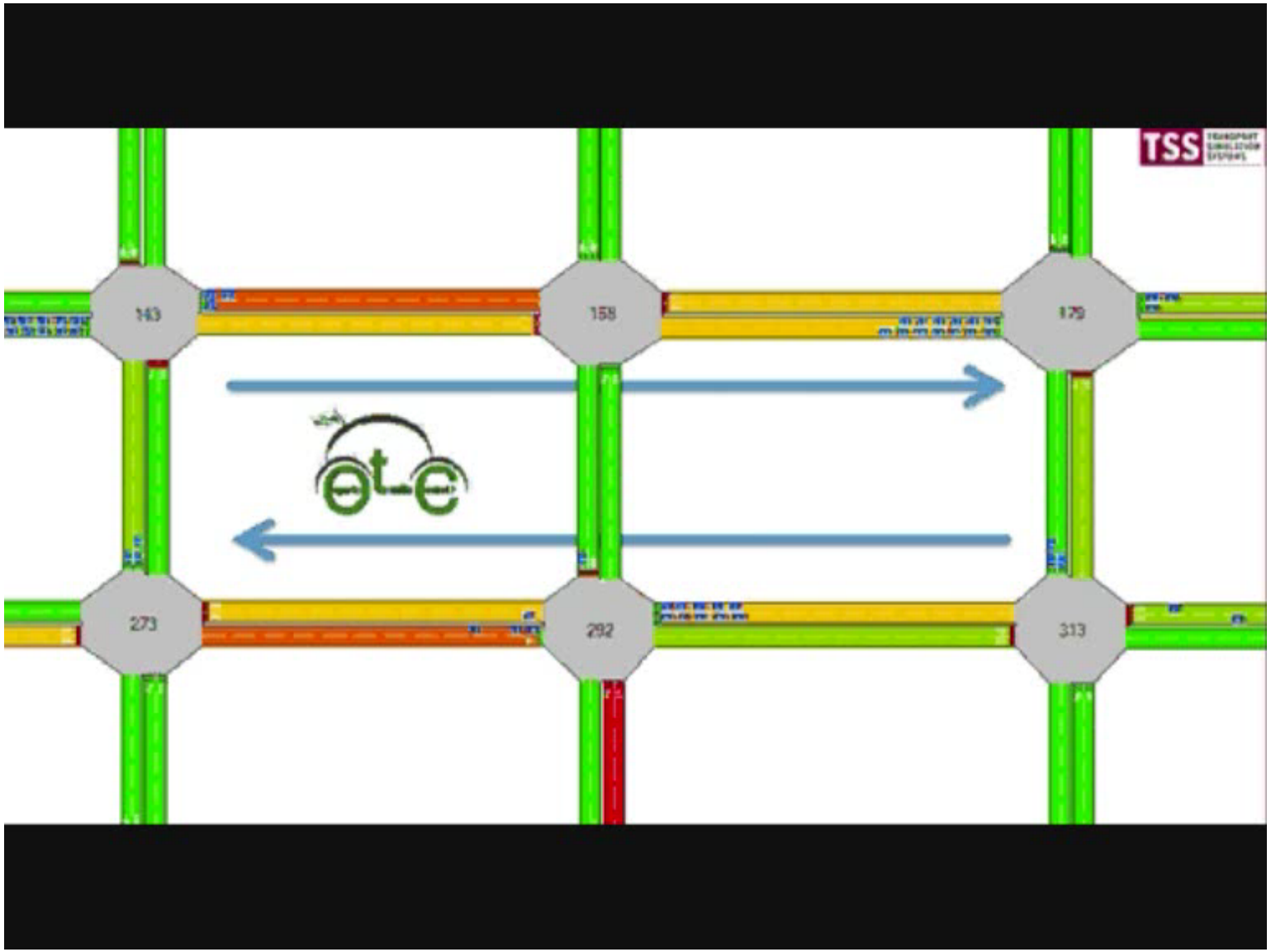
Reductions

	DVR	LST
Travel time	32%	
Stops	12%	
Fuel/CO_2	19%	



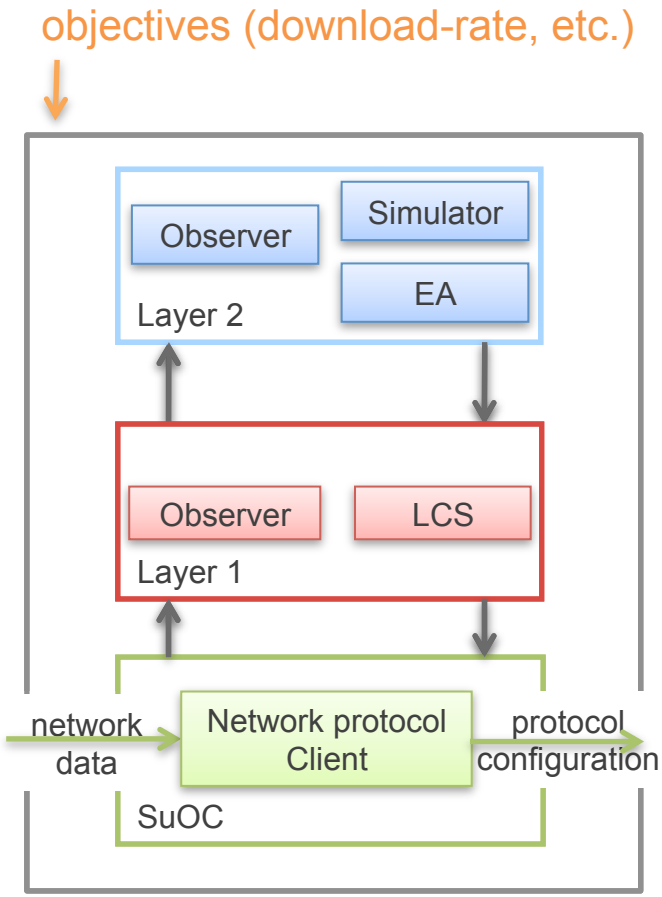


Organic Traffic Control
Progressive signal system

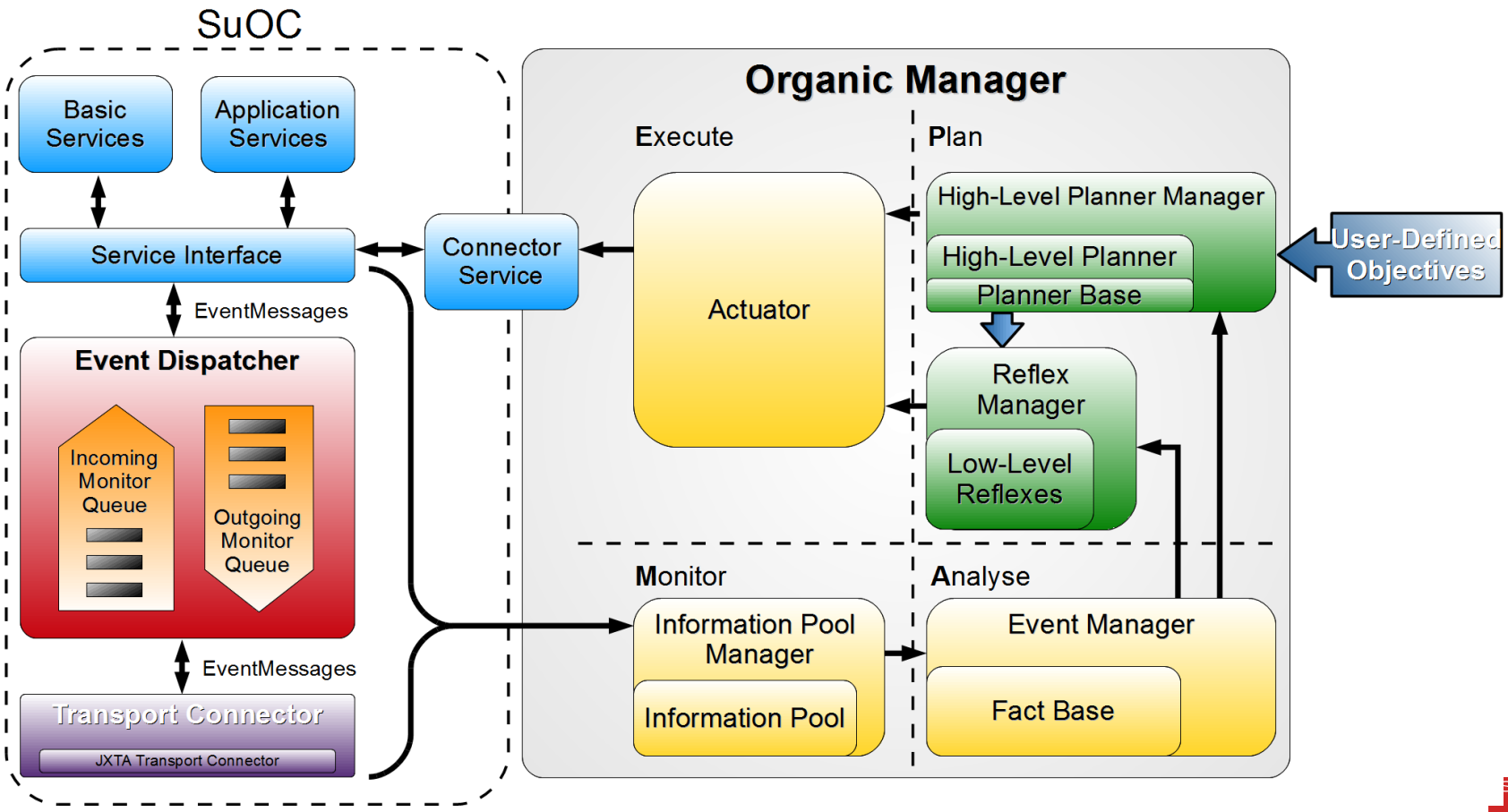


Example (2): Organic Network Control

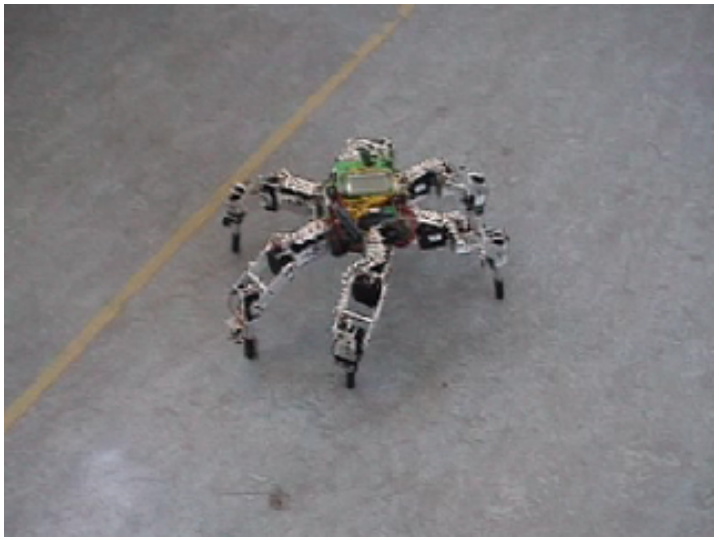
- Organic Network Control (ONC)
  - Goal: Adapt network protocols to dynamically changing environments in order to increase the system performance.
  - Autonomous network nodes observe the current status of the environment and adapt their behaviour accordingly.
  - Each node:
    - is self-organized,
    - learns and optimizes itself,
    - incorporates external goals,
    - but acts completely autonomously.
  - Additional collaboration mechanisms can be found: e.g. exchange of knowledge.

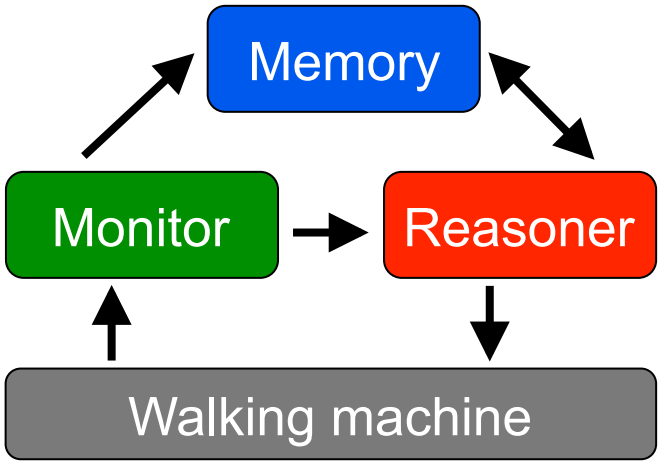


- Service-oriented middleware for self-x properties
  - Monitoring – Self-healing – Self-configuration – Self-optimization



- Organic Fault-tolerant Robot Control Architecture (ORCA)
  - Robots (here: walking machines) have to survive and move in unstructured, dynamically changing environments.
  - Complex control task, no explicit fault/world model available.
  - Organic approach to observe and control the walking application



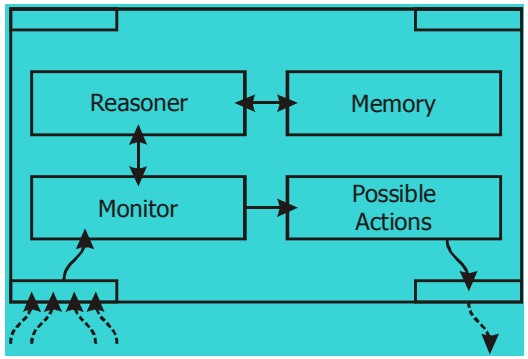


Variant of the Observer/Controller Architecture!

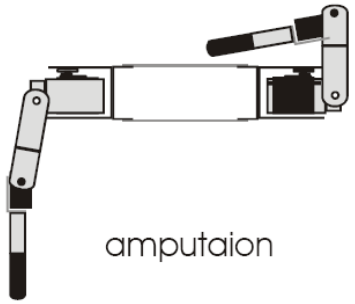
More details: http://www.iti.uni-luebeck.de/index.php?id=orca&L=1

Erik Maehle, Lübeck
Werner Brockmann, Osnabrück

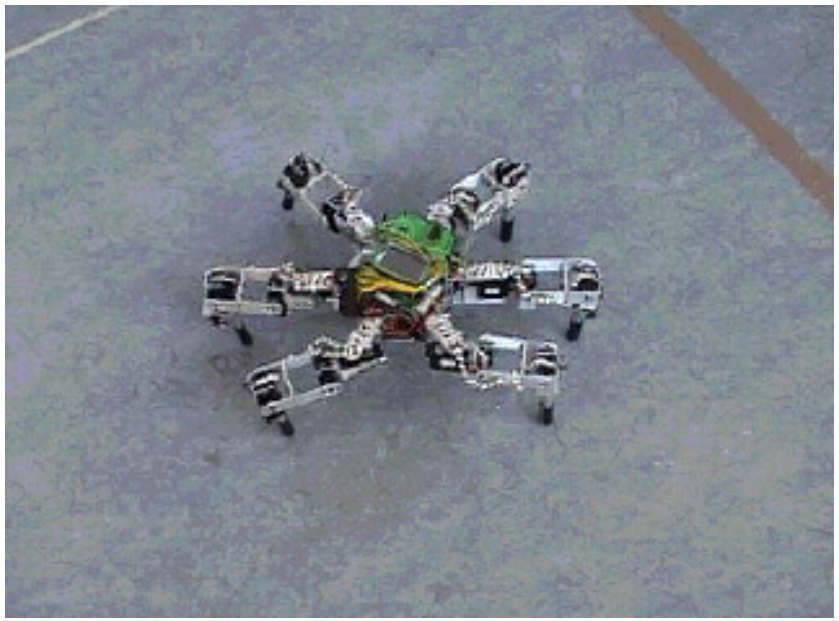
- 6-legged robot with 1 leg defect



Organic Control Unit



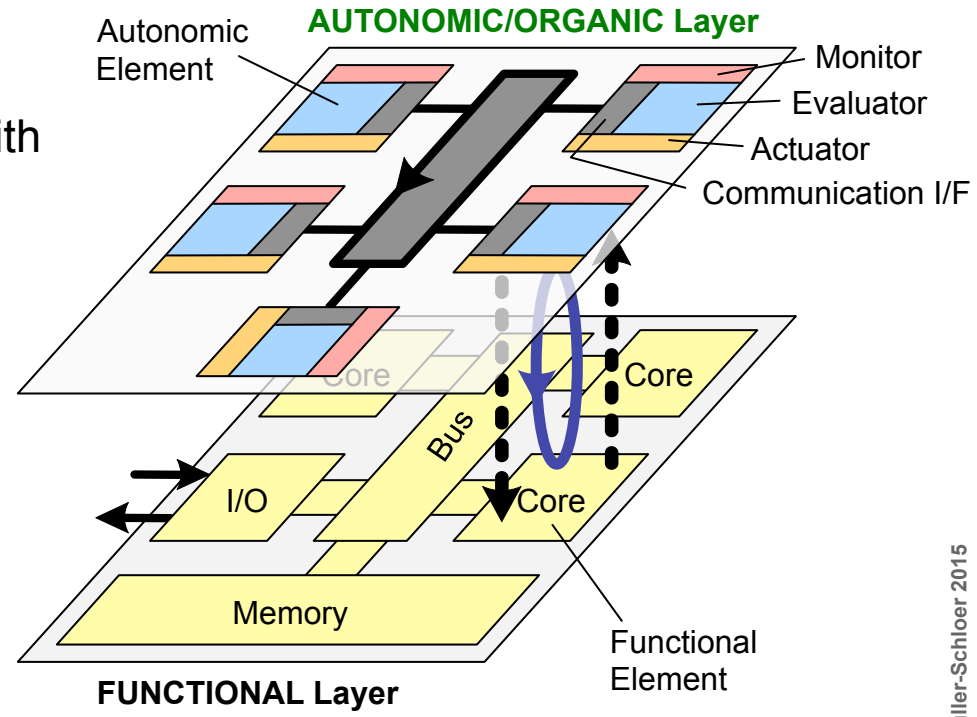
Simulated amputation



- Same *local* rules as with 6 functional legs (broken leg is skipped)
- New gait emerges through self-organization.

Example (5): Autonomic SoC Architecture

- Goal: Dynamic reliability, power, performance optimization at run time
- MPSoC system is logically split into two layers
  - Functional layer contains traditional SoC building blocks / IPs
  - Autonomic/Organic layer extends IPs with self-organization properties
    - Distributed LCS-based evaluator to adjust frequency, supply voltage, task assignment at run-time
  - LCS-based learning of optimized evaluator rules at design-time



More details: http://www-ti.informatik.uni-tuebingen.de/~bernauer/asoc/