Christian-Albrechts-Universität zu Kiel

Intelligent Systems

Chapter 3: Design of Intelligent Systems

Winter Term 2019 / 2020

Prof. Dr.-Ing. habil. Sven Tomforde
Institute of Computer Science / Intelligent Systems group

About this Chapter

Christian-Albrechts-Universität zu Kiel

Content

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings

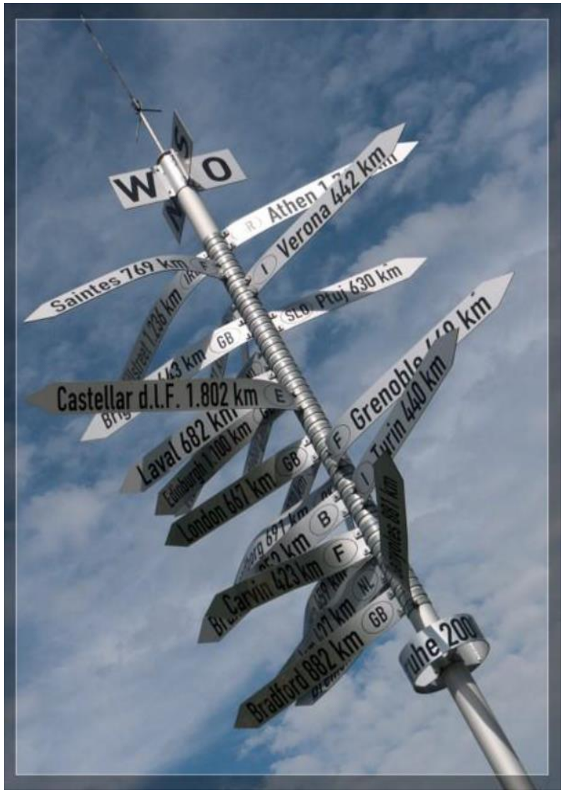
Goals

Students should be able to:

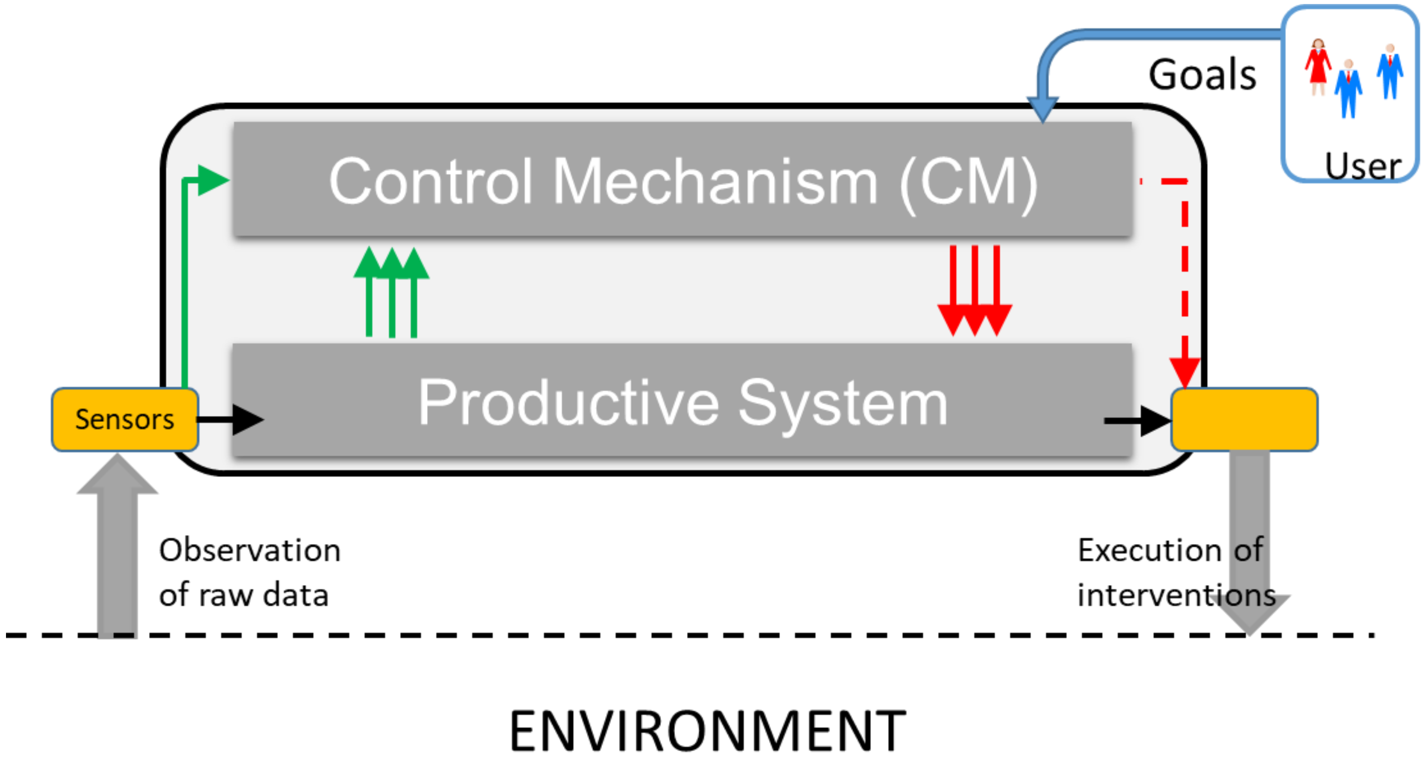
- Outline the basic design concept used in intelligent systems.
- Sketch the Observer/Controller approach and the multi-level extension.
- Compare distribution variants.
- Explain the responsibilities of the contained components.
- Describe how a social-inspired system structure looks like.
- Summarise what goal-oriented holonics means.

Agenda

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



General concept of an intelligent system



Green: flow of observed data
Red: flow of control interventions

General concept of an intelligent system (2)

System boundaries

- Everything not directly related to the system itself is modelled as being part of the external environment.
- Environment is accessed through sensors and manipulated by actuators.

Take the human operator "out of the loop"

- Productive system is only influenced by internal control mechanism.
- Theoretical possibility for the user to intervene directly with the output signal.
- User provides just a goal (or a set of goals) defining good or bad behaviour.
- CM figures out what to do in which situation without direct human intervention.
- Classic approach: user specifies directly what the system has to do.

General concept of an intelligent system (3)

Requirements

- Access to sensor readings covering all important aspects.
- Access to internal status variables of productive system.
- Utility function specifying good/bad behaviour.
- Access to control interfaces of productive system.
- Control interface has to be directly related to performance.

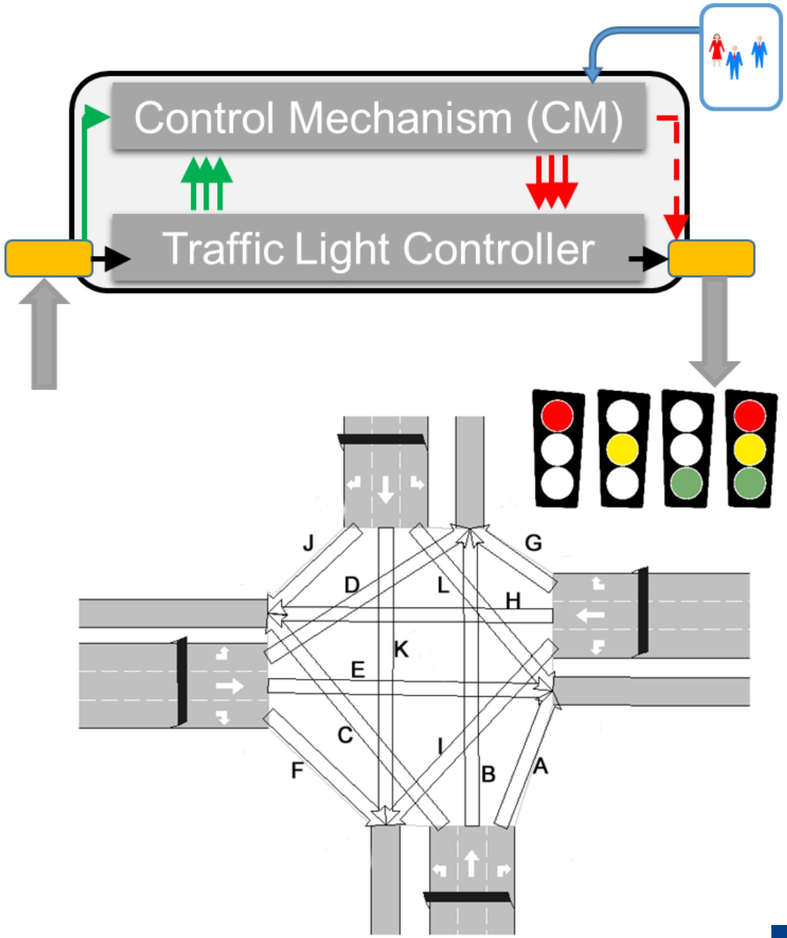
Actions

- Manipulation of parameters to guide behaviour of productive system.
- Activation or deactivation of components and functionality.
- Exchange of algorithms or techniques in use.
- Selection of interaction partners.
- Modification of observation model (i.e. selection, resolution, and frequency of sensor information)

General concept of an intelligent system (4)

Example: Traffic control

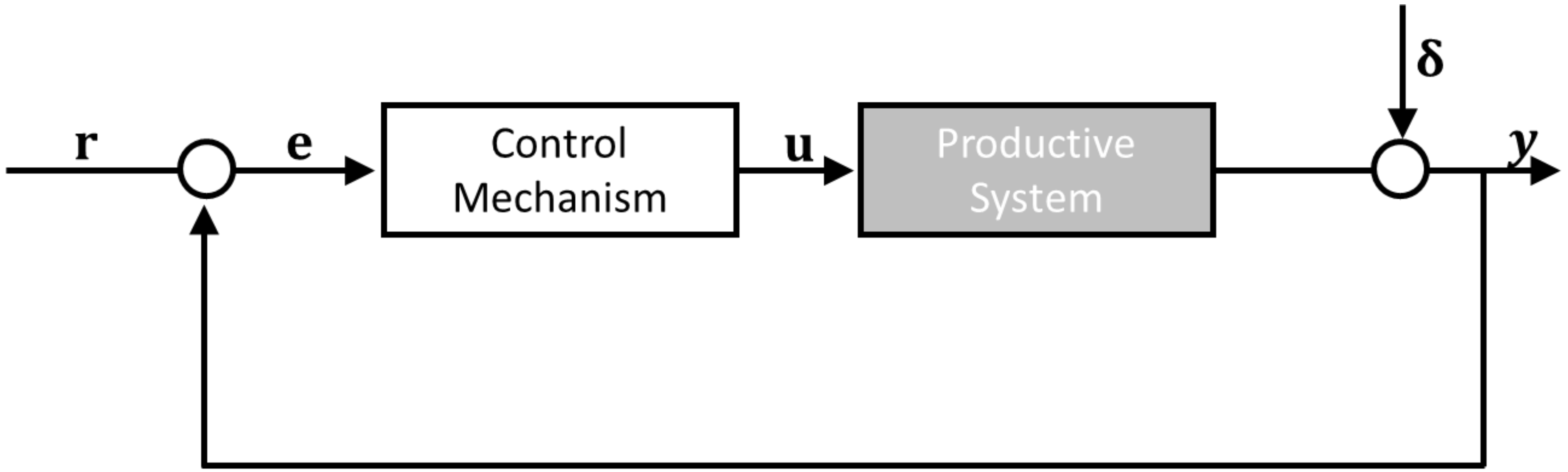
- Productive system: traffic light controller
- Sensor readings: induction loop readings (occupied or not)
- Effectors: traffic signals (green/yellow/red)
- Input CM: vector with traffic flow per hour for each turning
- Output CM: green durations for each traffic light



General concept of an intelligent system (5)

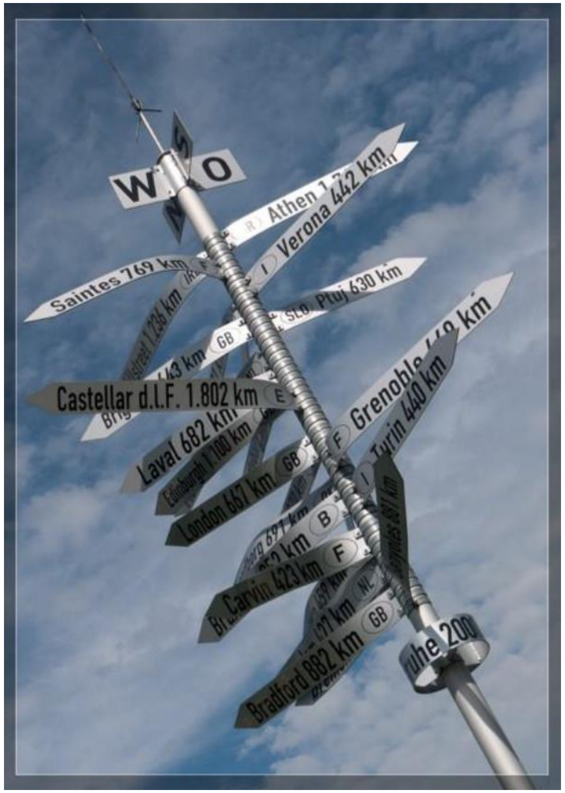
Control loop

- Actions have impact on performance
- Performance is measured from the environment
- Environment is modified by actions
  → Feedback loop (see control theory)

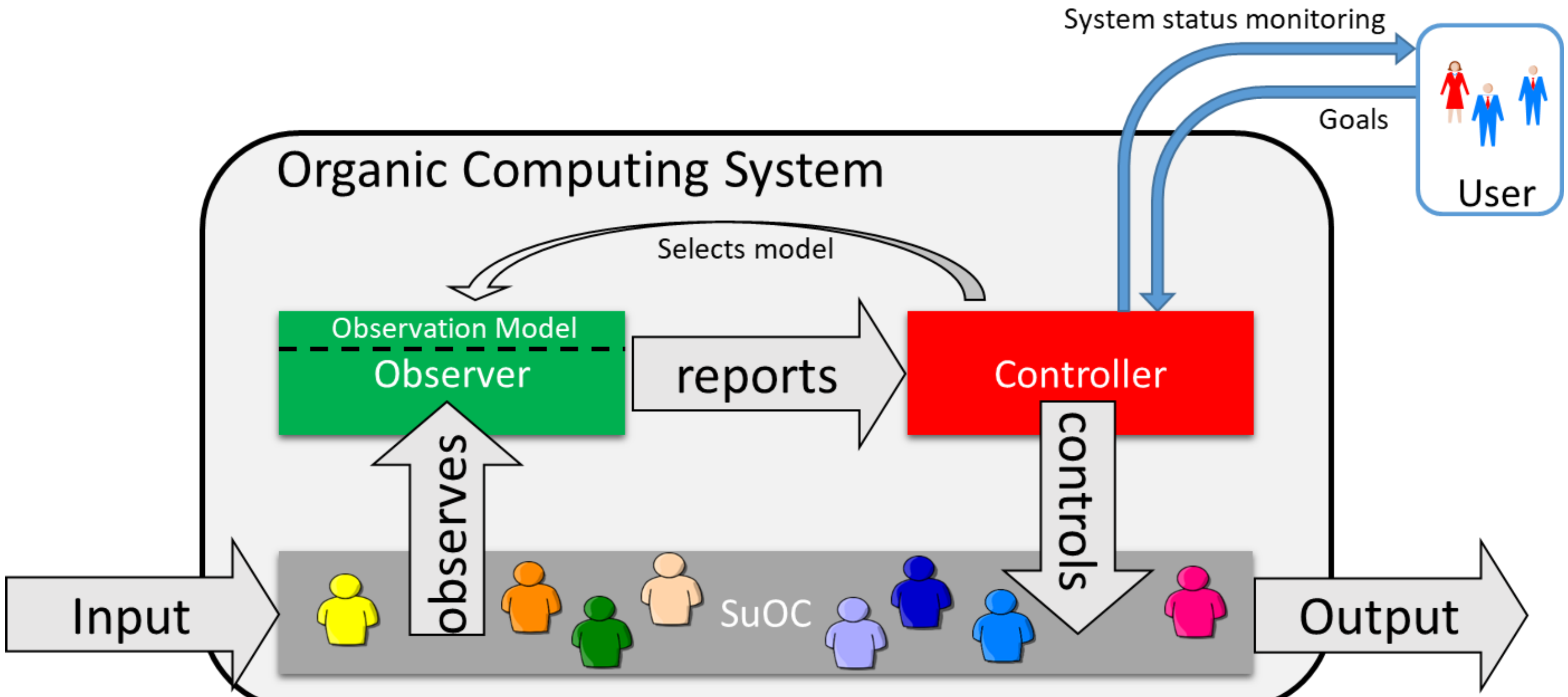


Agenda

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



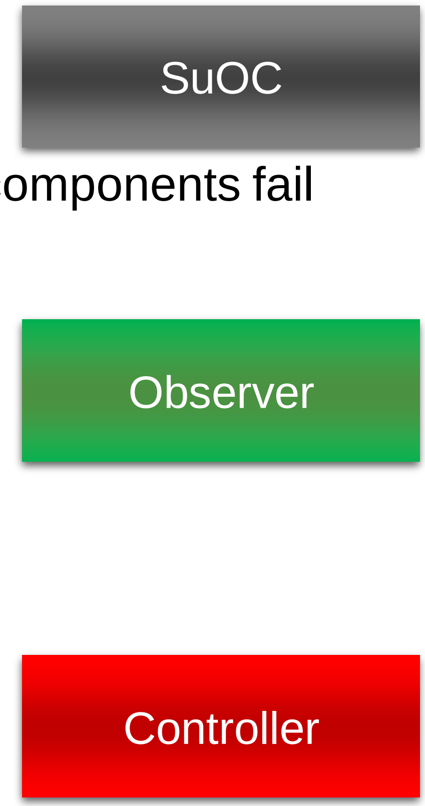
Generic Observer/Controller Pattern



Generic Observer/Controller Pattern (2)

Three major components:

- System under Observation and Control (SuOC):
  - Productive part of the system
  - Functional even if higher-layered observer/controller components fail
  - Not restricted to individual devices
- Observer:
  - Gathers information
  - Analysis, predicts, detects patterns
  - Builds situation description
  - Based on observation model
- Controller:
  - Decides about actions
  - Learns from feedback
  - Modifies observation model



Generic Observer/Controller Pattern (3)

Basic idea: Learning from feedback

- Rule-based system
- Rule:

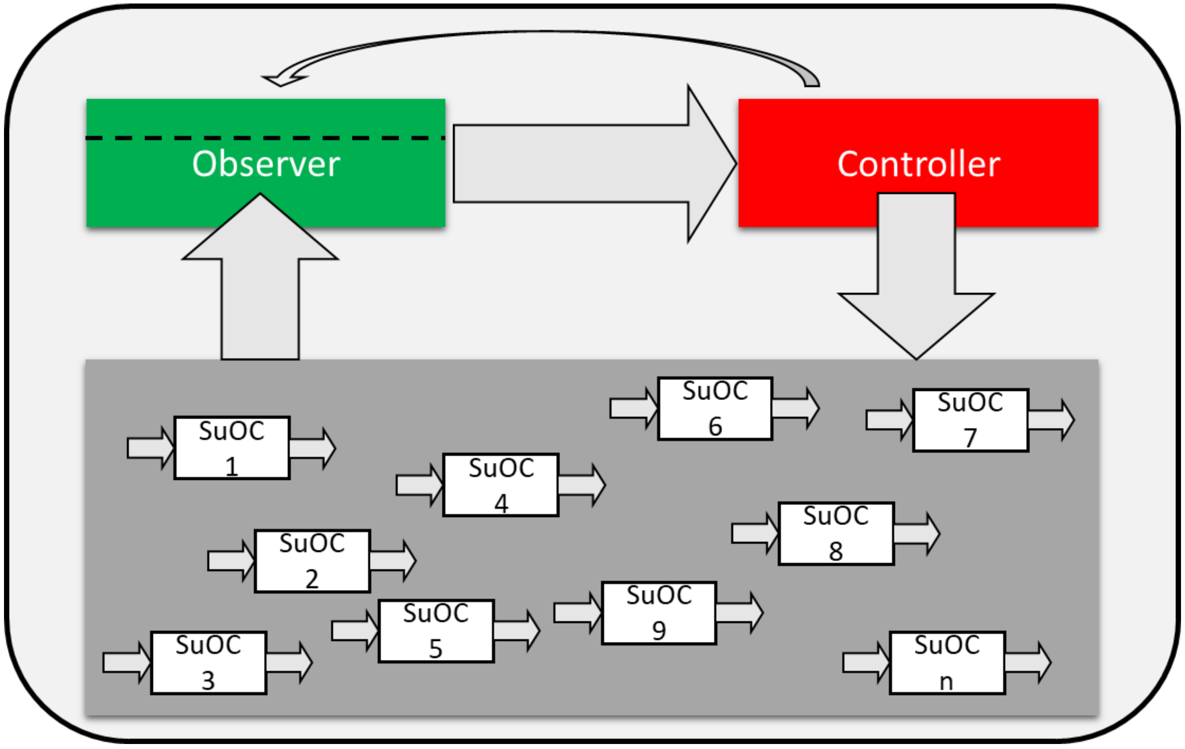
  [Condition] [Action] [Evaluation]
- Rules compete, more than one can fulfil condition
- Evaluation criteria used to decide which to take, updated in each cycle

Traffic example

- Condition: vector with intervals for traffic flows per hour per turning
- Action: Green durations per traffic light
- Evaluation: Averaged delays occurring at intersection

Distribution variants

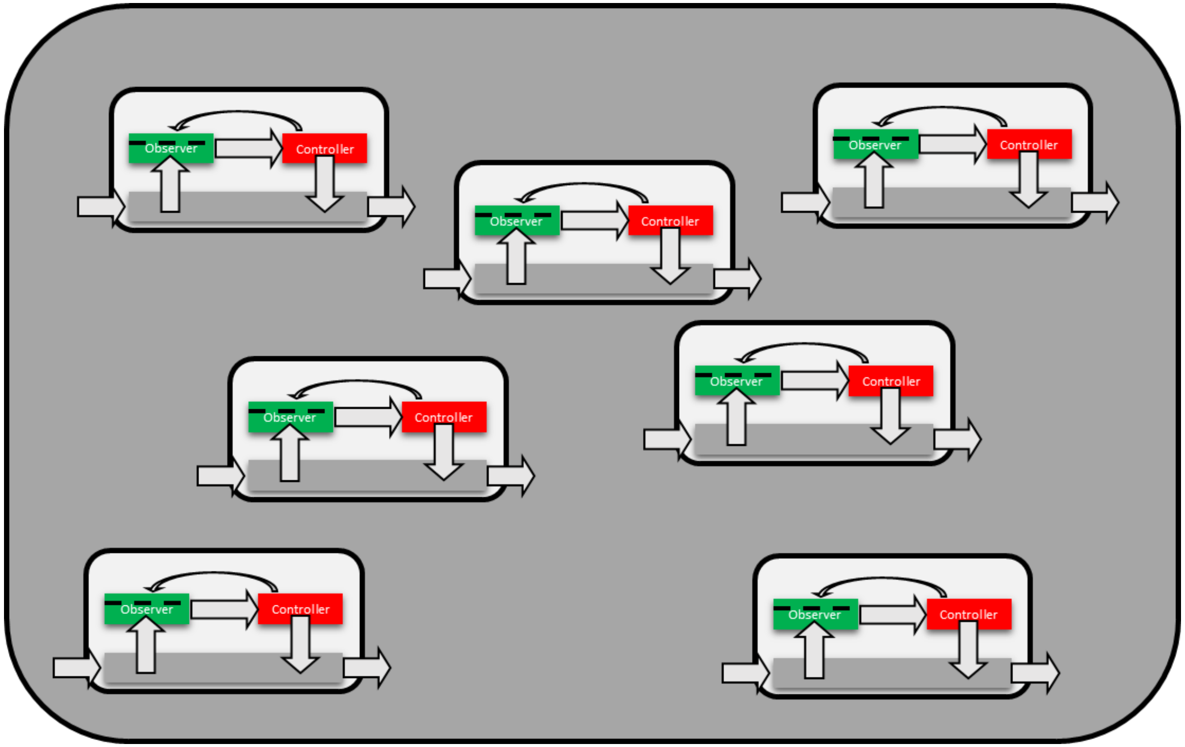
Variant A: Fully centralised



Distribution variants (2)

Christian-Albrechts-Universität zu Kiel

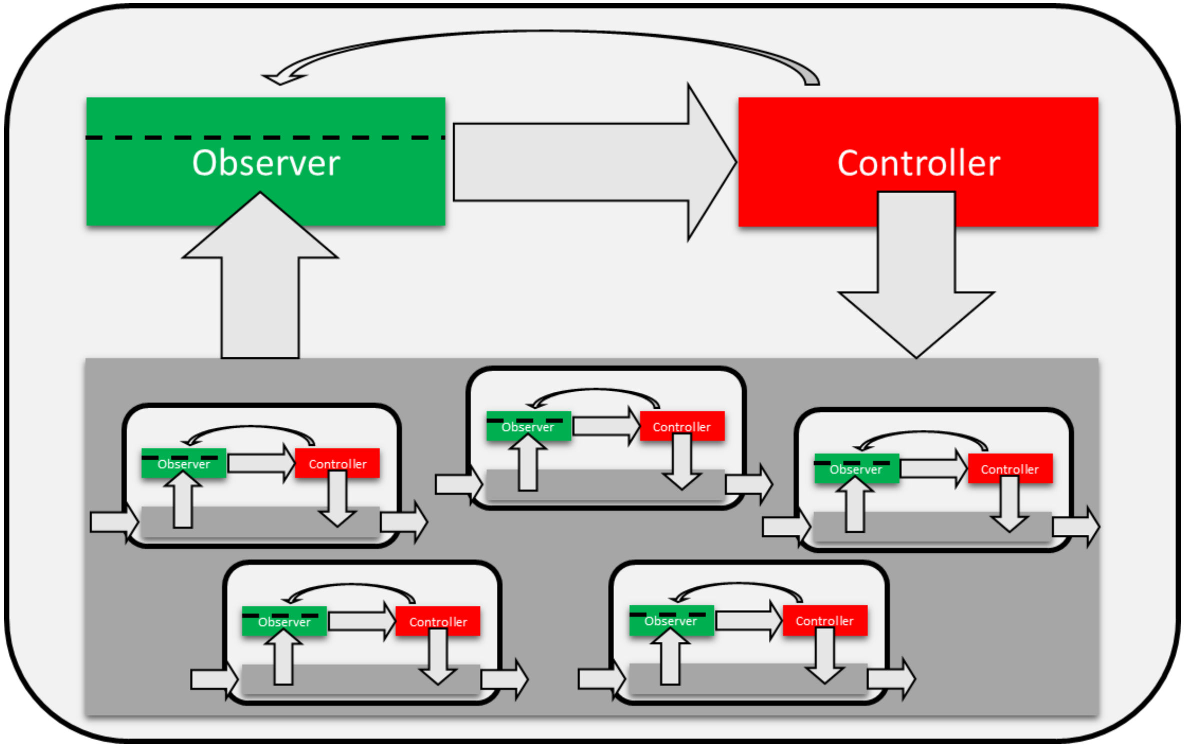
Variant B: Fully decentralised



Distribution variants (3)

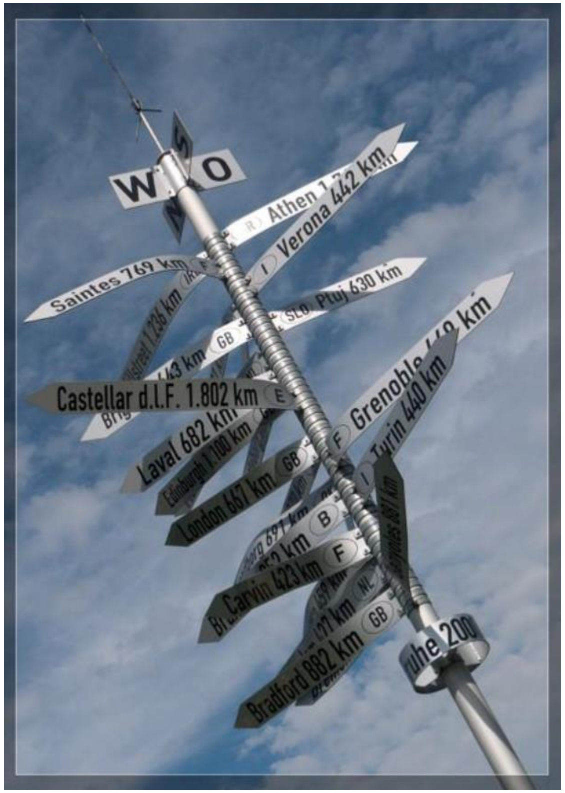
Christian-Albrechts-Universität zu Kiel

Variant C: Hybrid

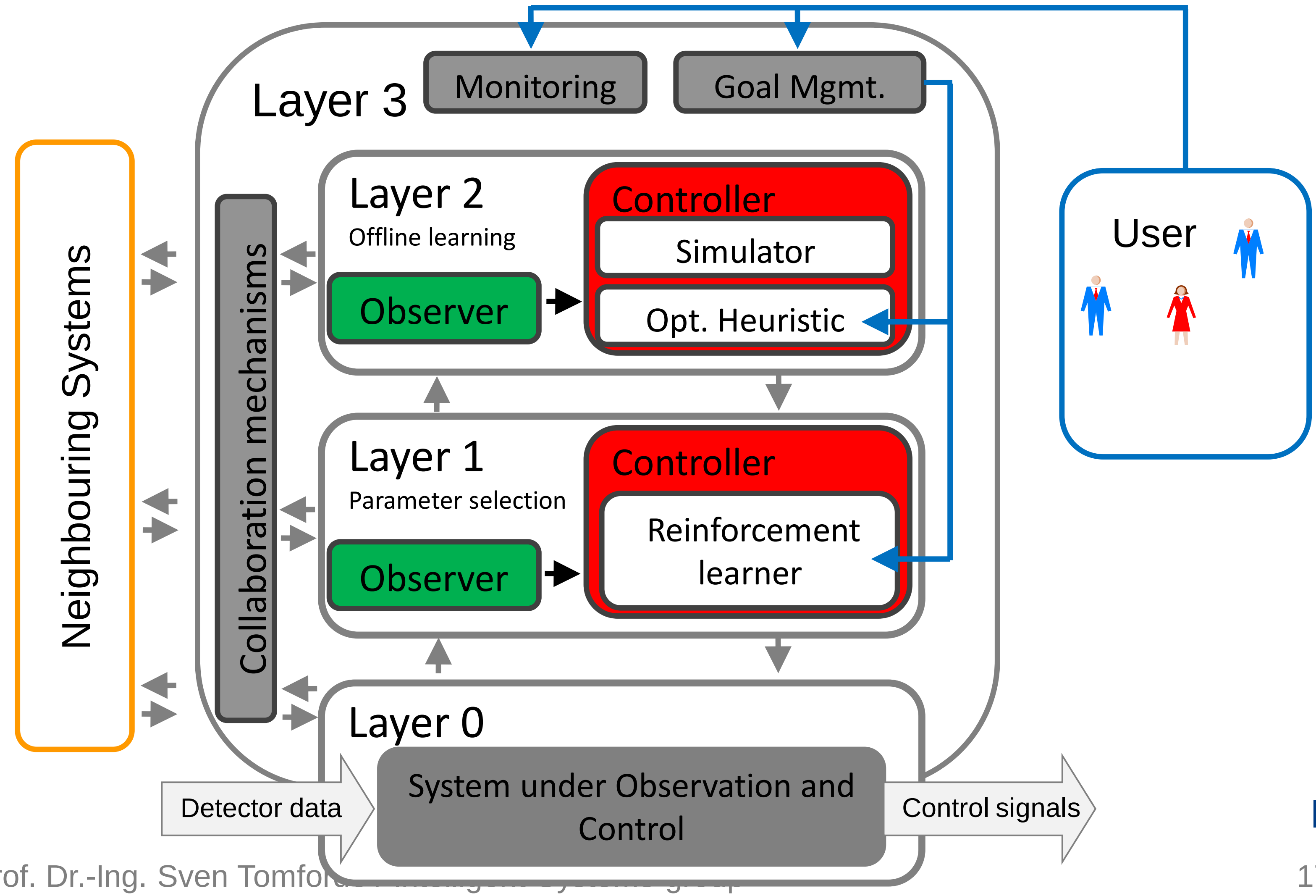


Agenda

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



Multi-level Observer/Controller framework

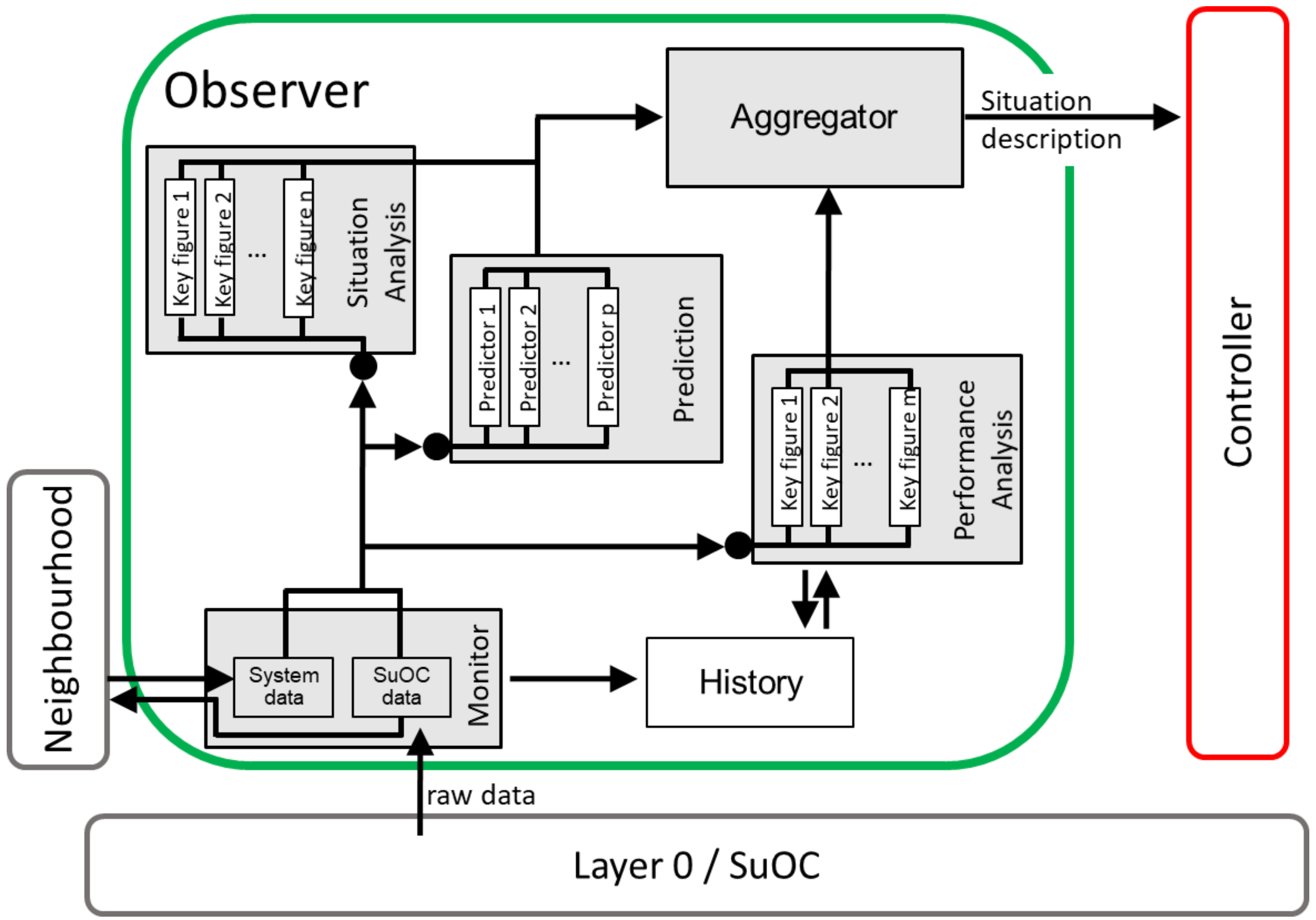


Components of MLOC

Christian-Albrechts-Universität zu Kiel

- Layer 0: Productive system
  - Encapsulation of the SuOC
  - Provide access to observation and control interfaces
- Layer 1: Parameter selection and online learning
  - Observation of Layer 0:
    - Monitoring, pre-processing, data analysis, prediction, aggregation
    - Generation of situation description
  - Control:
    - Change parameters, structure, techniques to current conditions.
    - Select from existing rule-set, update [Evaluation] based on feedback.
- Layer 2: Offline learning
  - Observation of Layer 1: Missing or inappropriate knowledge
  - Control: Generate new rule, add to rule-based of Layer 1
- Layer 3: Collaboration
  - User: Self-explanation, goal modification
  - Other systems: Negotiation, communication, trust, …

Observer component at Layer 1

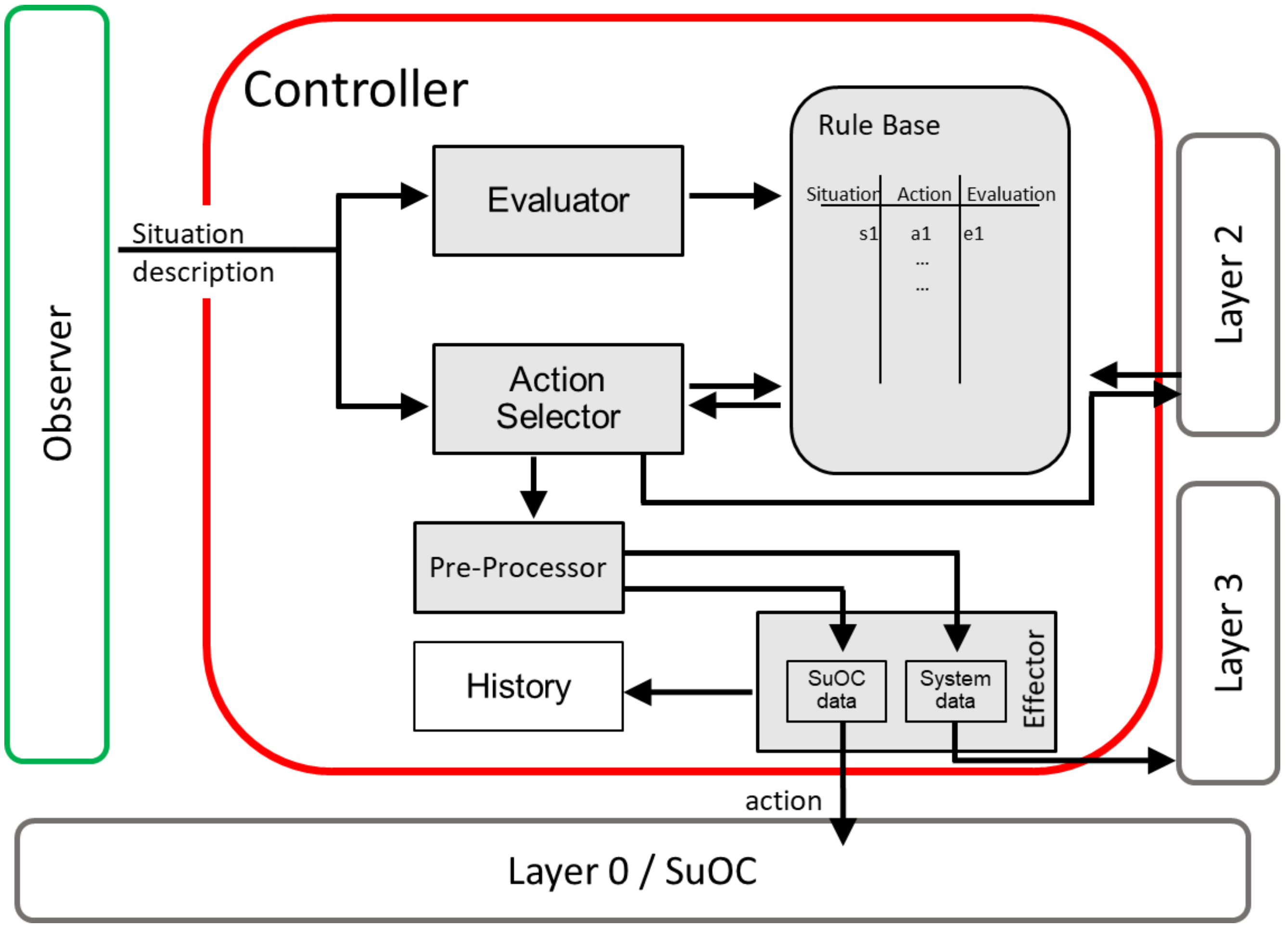


Observer component at Layer 1 (2)

Components:

- Monitor: Gather sensor and status data
- Pre-processing: Estimate missing values, delete wrong values, smoothing, etc.
- Prediction: Forecasts for some of the underlying values
- Situation analysis: Find patterns, detect emergence, etc.
- Performance analysis: Extract values related to goals
- Aggregator: Build situation description
- History: List of observed data (for debugging and explanation)

Controller component at Layer 1



Controller component at Layer 1 (2)

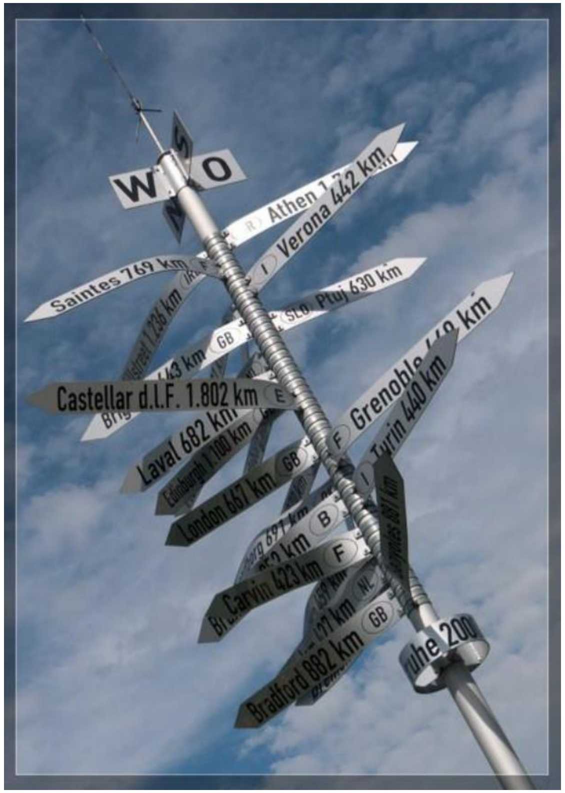
Components:

- Rule base: Set of rules, knowledge of the system
- Action selection: Select rule, action is encoded
- Effector: Activate action via effectors
- Evaluator: Determine success of last action(s), update rules
- Action history: List of situation-action mapping that have been performed

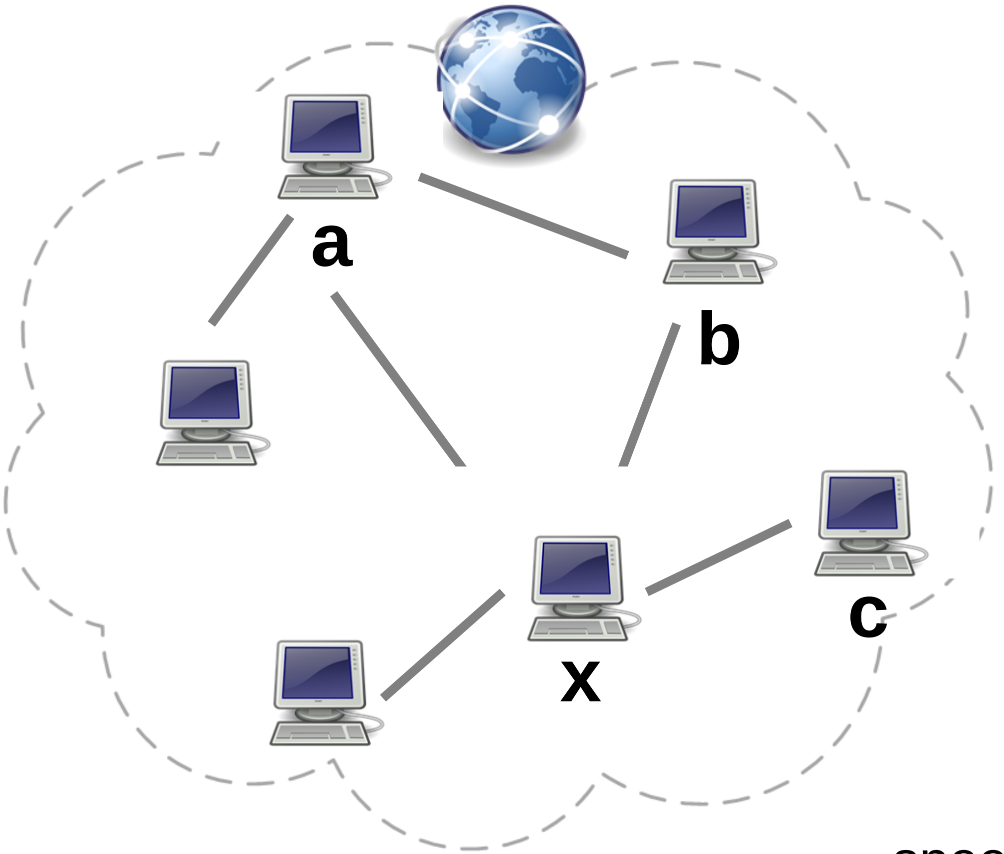
Agenda

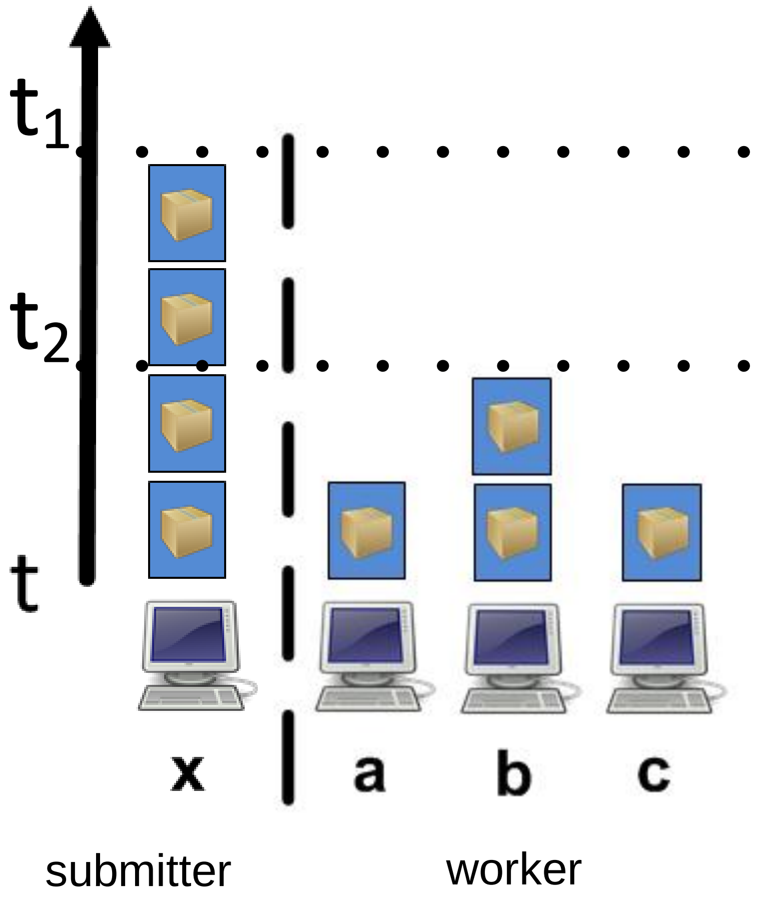
Christian-Albrechts-Universität zu Kiel

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



Decentralised Desktop Grids



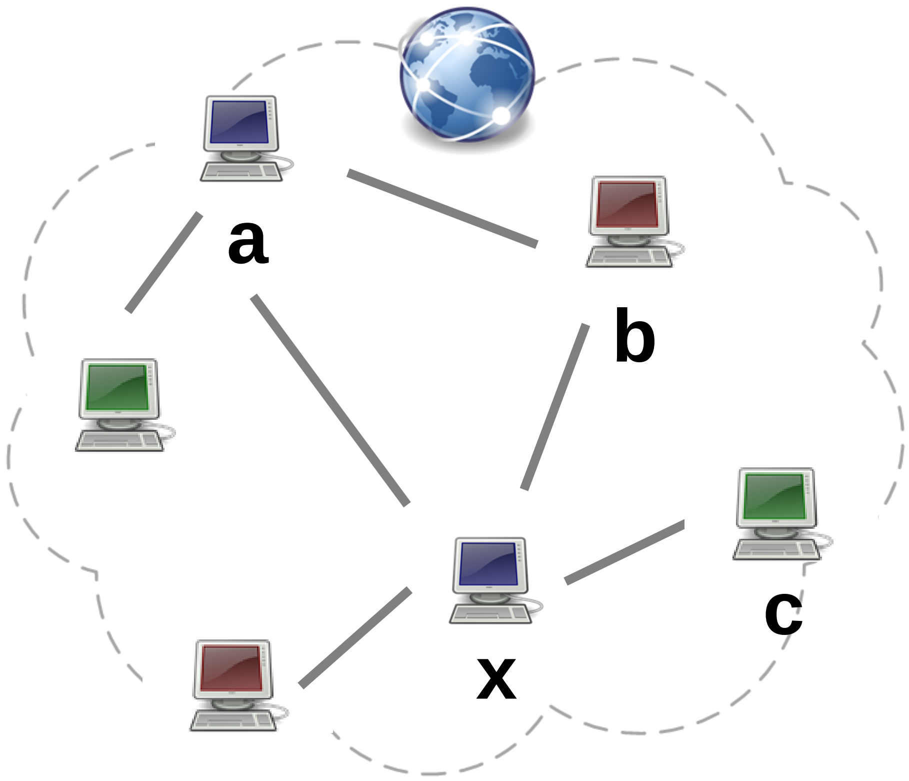


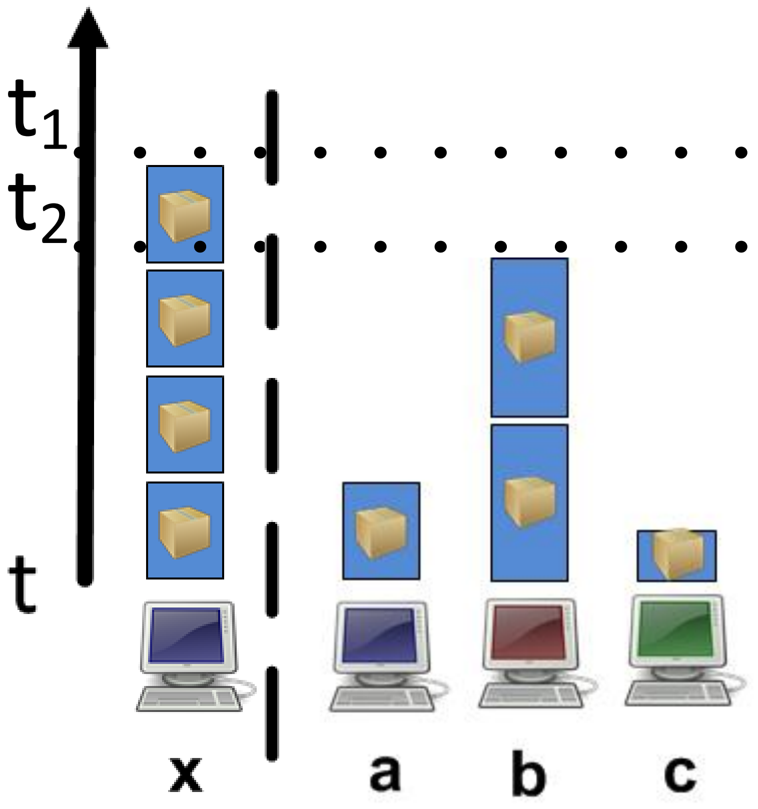
speedup = t_2 / t_1

Decentralised Desktop Grid Systems (2)

Christian-Albrechts-Universität zu Kiel

Heterogeneity

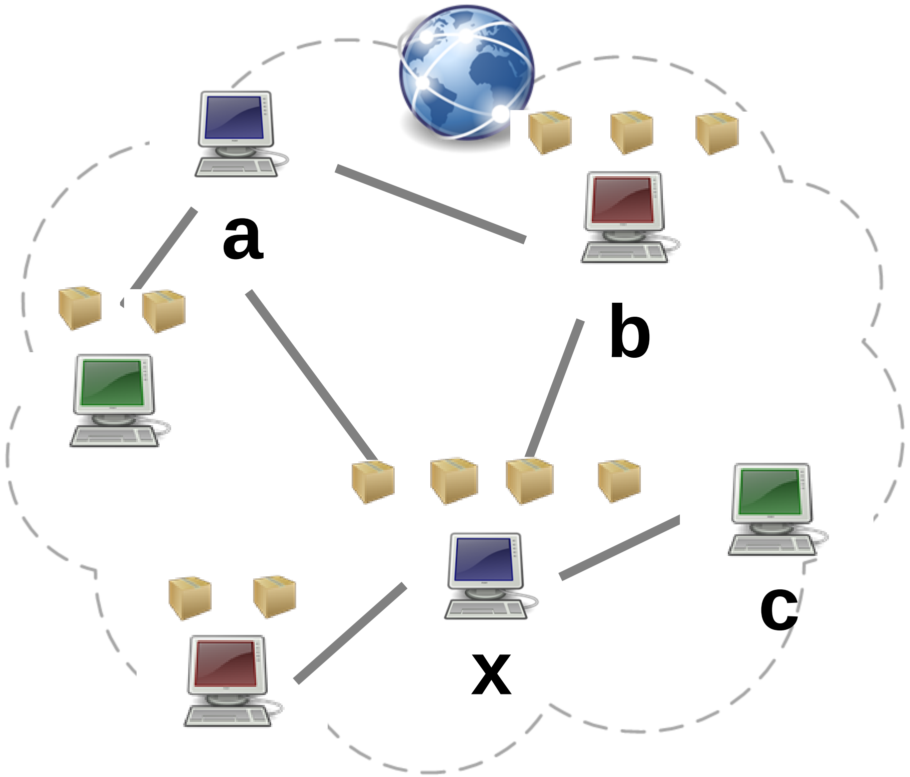


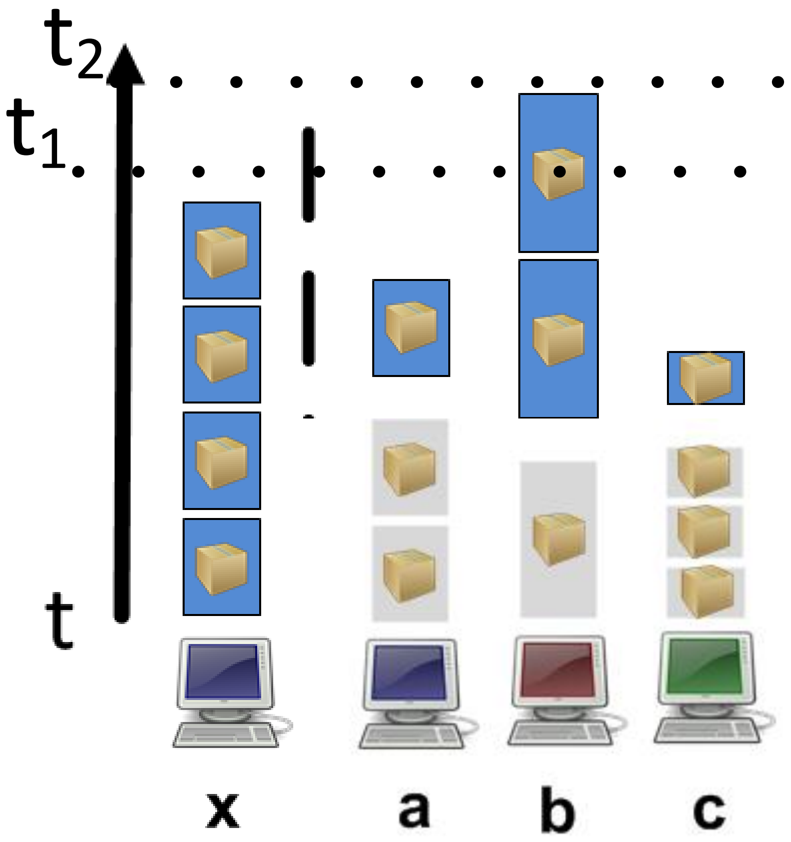


Decentralised Desktop Grid Systems (3)

Christian-Albrechts-Universität zu Kiel

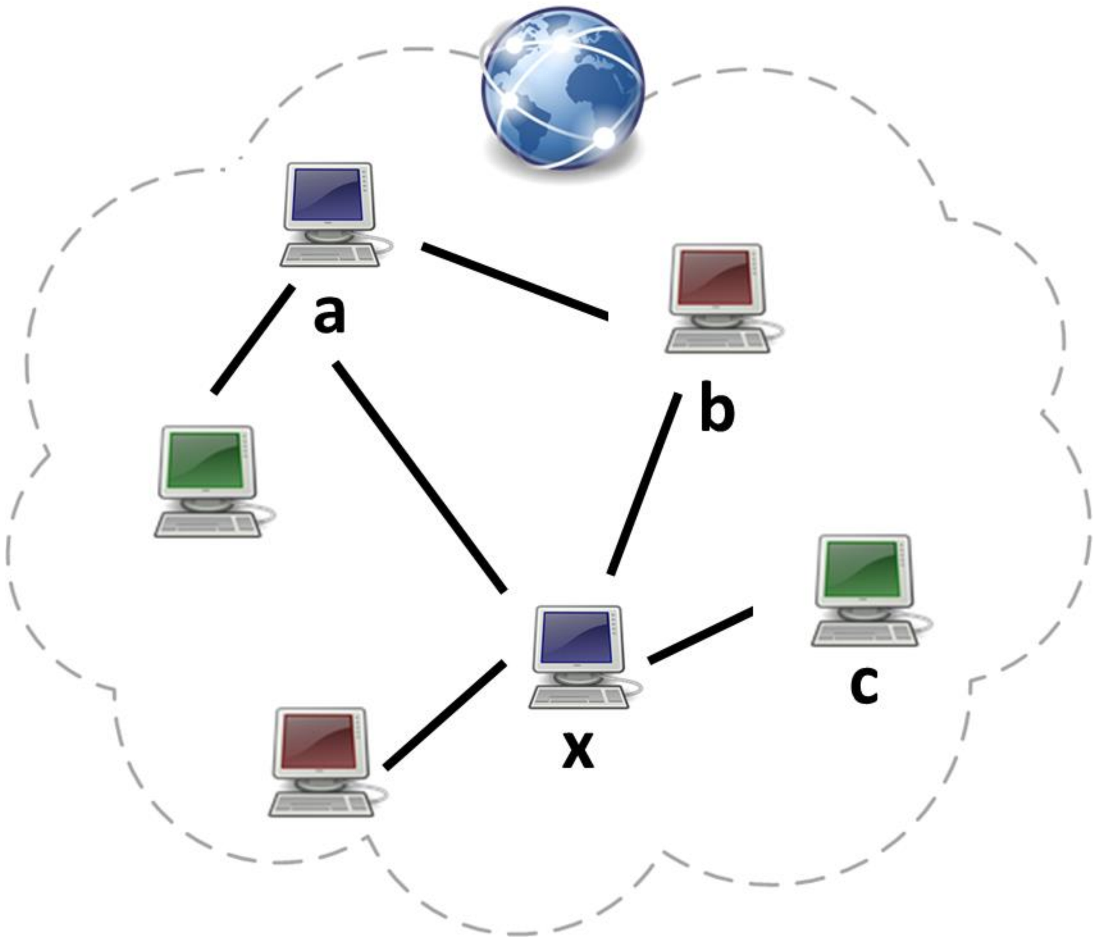
Competition





speedup = t_2 / t_1

Intelligent Desktop Grid Systems

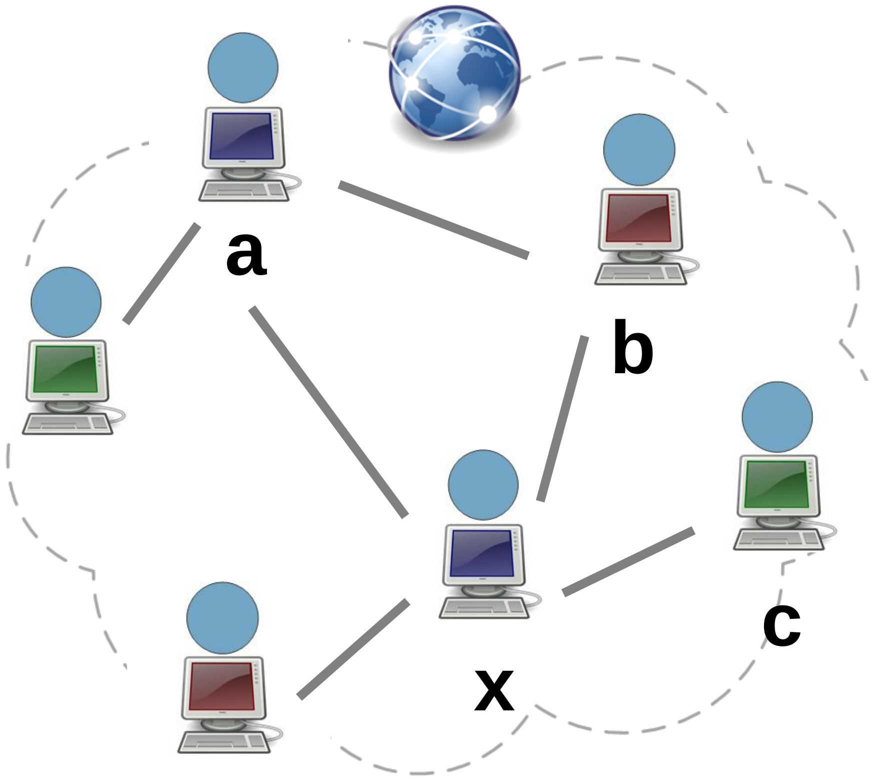


- Entity / Agent ●
  - Utility: Achieve good speedup
  - Autonomous decision-making
  - Black boxes

- OC-based distributed system
- Open(?)

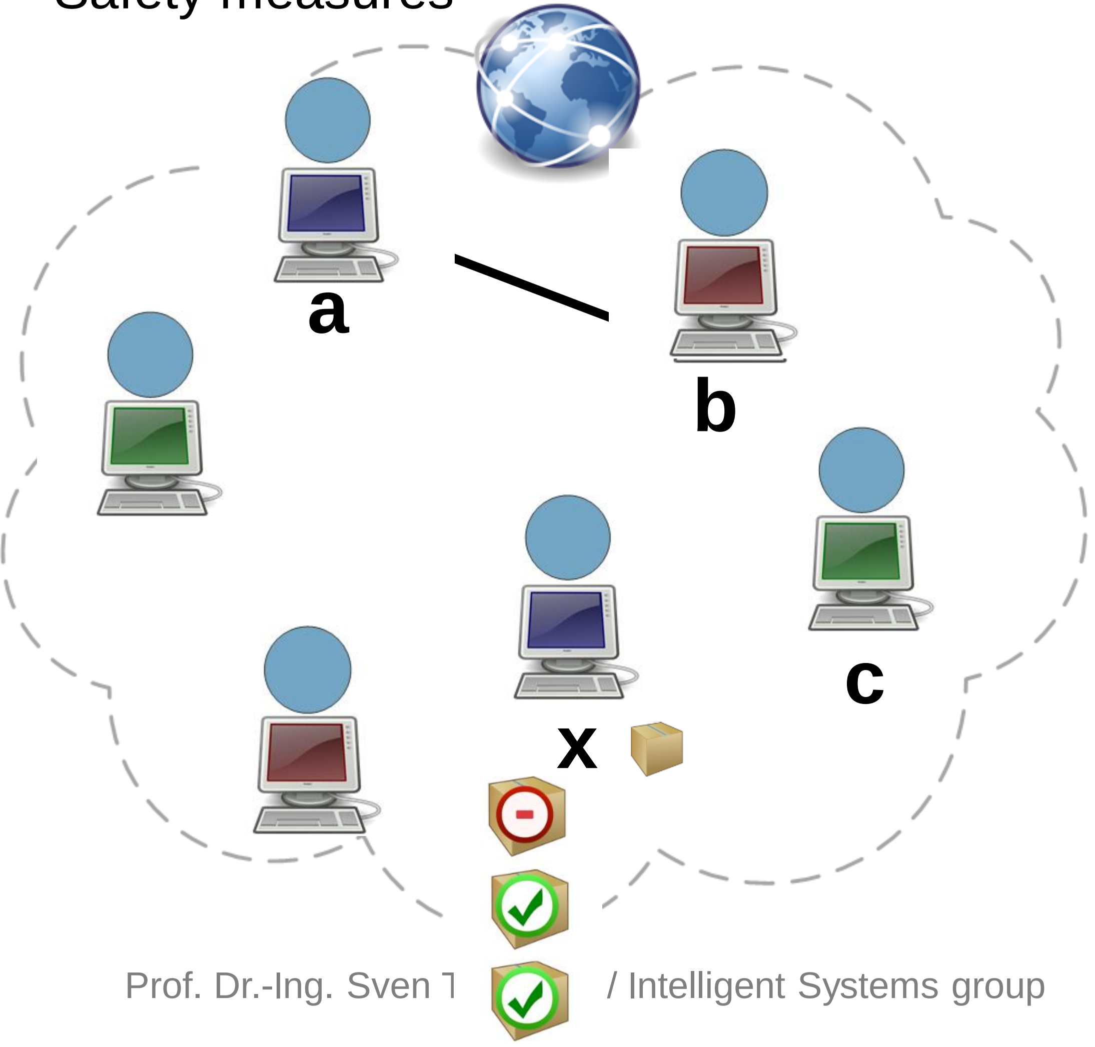
Open Desktop Grid Systems

Cooperation



Open Desktop Grid Systems (2)

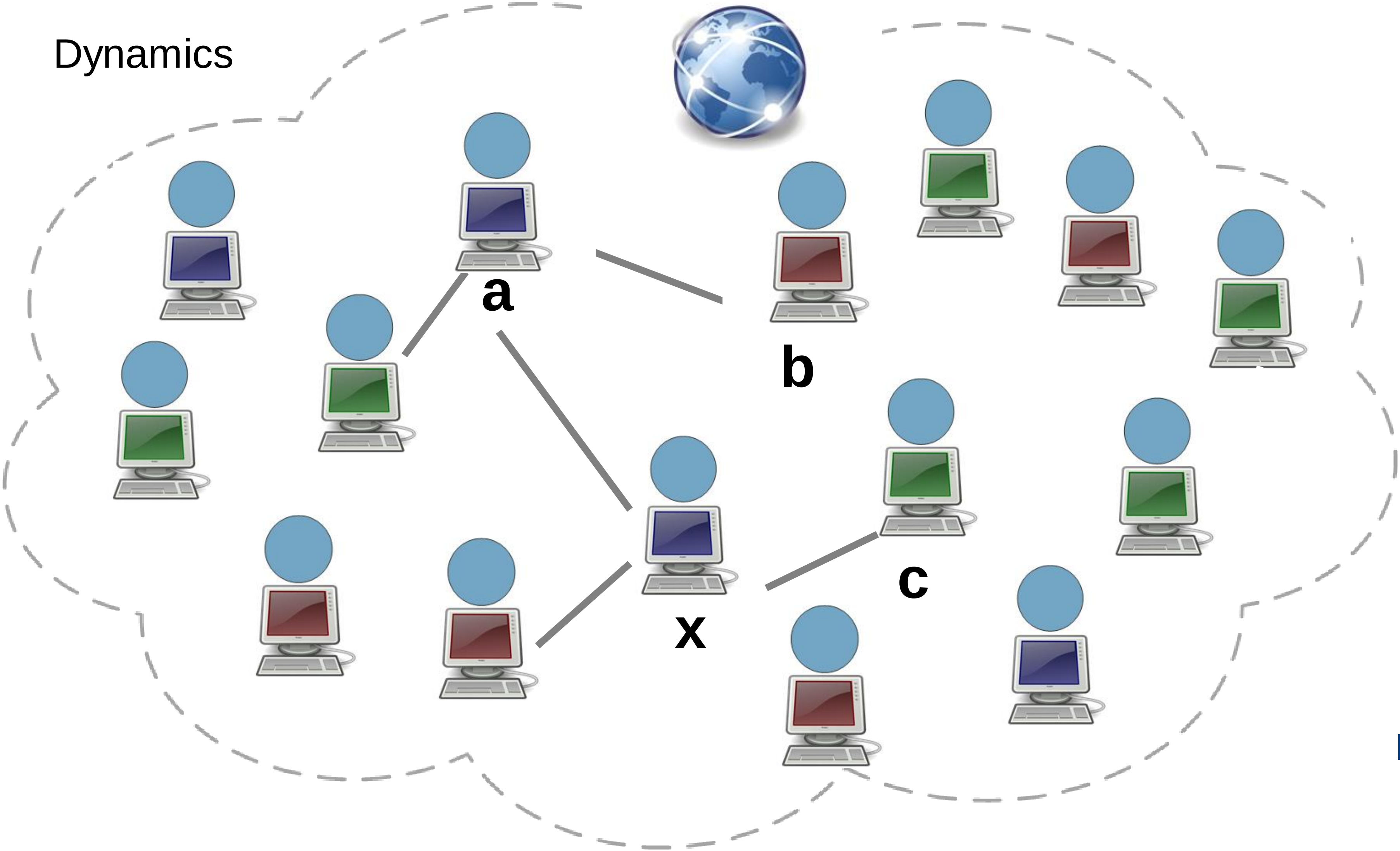
Safety measures



- Agents
  - Can act selfishly
  - Can be uncooperative
  - Run on faulty hardware
- Requires
  - Safety measures: Replication
- Leads to
  - Overhead: Higher workload

Open Desktop Grid Systems (3)

Dynamics

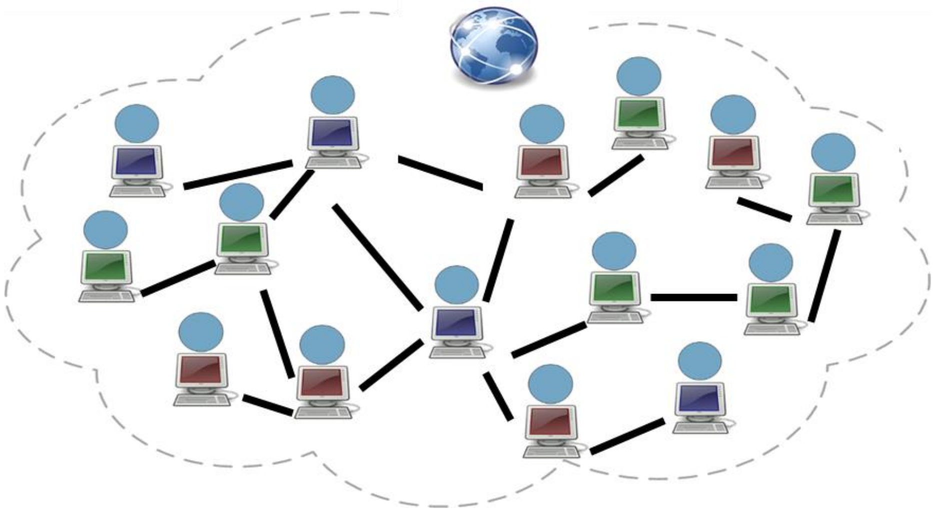


Control of Open Distributed Systems

Christian-Albrechts-Universität zu Kiel

Challenges for Open Distributed Systems (ODS):

- How to realise an ODS that - despite its openness and distributed nature - provides a high performance to participants (i.e. speedup for the grid scenario) and is robust towards disturbances?
- How to decentralise control?
- How to realise the according agents?



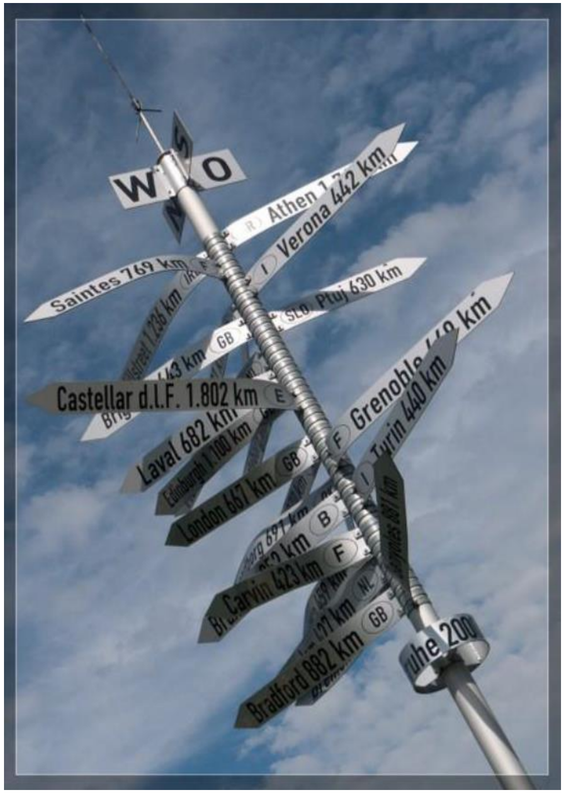
Application examples:

- Intelligent Power Grid Systems
- Vehicular Ad-Hoc Networks
- Cloud Computing
- Internet of Things
- Wireless Sensor Networks
- E-Commerce

Agenda

Christian-Albrechts-Universität zu Kiel

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



Motivation

Building collective OC systems

- Collections of smaller entities instead of one monolithic solution
- Challenge: Allow for collective behaviour of autonomous entities
- Entities act in an overall ecosystem, access common pool resources

Distinguish (from the Oxford dictionary):

- Collaboration:
  "the action of working with someone to produce something"
- Cooperation:
  "the organisation of the different elements of a complex body or activity so as to enable them to work together effectively"
- Competition:
  "activity or condition of striving to gain or win something by defeating or establishing superiority over others"

Tragedy of the commons

I want another cow!
I want another cow!
I want another cow!
GEORGE
OneMinuteEconomics.com

https://www.youtube.com/watch?v=jSuETYEgY68

Tragedy of the commons (2)

The “tragedy of the commons“

- Is a situation in a shared-resource system
- Individual users acting independently according to their own self-interest
- They behave contrary to the common good of all users by depleting or spoiling that resource through their collective action

Connection to OC:

- Distributed resources
- Self-motivation of autonomous systems (egoism) vs. system-wide goals

Enduring organisations

Christian-Albrechts-Universität zu Kiel

Systems (consisting of many autonomous subsystems) have to persist:

- If their subsystems change
- If the purpose changes (→ flexibility in term of user goals)
- If the environment changes
- If novel components appear, others disappear

Goal: Establish enduring organisations

- Elinor Ostrom (Nobel prize winner in economics)
- Investigated social systems
- Found eight principles for successful enduring organisations

Source: [wikipedia.org]

Ostrom‘s principles

Christian-Albrechts-Universität zu Kiel

1. Clearly defined boundaries: Those who have rights or entitlement to appropriate re-sources from the common pool resources are clearly defined, as are its boundaries.
2. Congruence between appropriation and provision rules and the state of the prevailing local environment: The rules must prevent overuse or degradation of the common goods.
3. Collective choice arrangements: In particular, those affected by the operational rules participate in the selection and modification of those rules. This principle prevents third parties imposing their interests.
4. Monitoring, of both state conditions and appropriator behaviour, is by appointed agencies: Agencies are either accountable to the resource appropriators or are appropriators themselves. This principle means that only such people may monitor the common pool resources who are involved in the common pool themselves. This prevents corruption and manipulated monitoring.

Ostrom's principles (2)

5. A flexible scale of graduated sanctions for resource appropriators who violate communal rules: This principle defines in which way a peasant violating the rules can be sanctioned.
6. Access to fast, cheap conflict resolution mechanisms: A result of this principle is that the reaction to conflicts can occur fast, by e.g. changing the rules of farming or sanctioning a peasant.
7. Existence of and control over their own institutions is not challenged by external authorities: This rule states that the enduring institution must be self-ruling. External authorities overriding the rules might endanger the stability of the system.
8. Systems of systems: Common pool resources can be layered or encapsulated. This principle means that hierarchies of enduring institutions are possible in order to save communication overhead or to simplify decision-making processes.

Ostrom‘s principles in a technical context

Christian-Albrechts-Universität zu Kiel

Considering Ostrom‘s principles:

- Clear status: Who is part, who has access?
- Rules have to take conditions into account
- Rules as result of negotiation
- Elected authority is in charge
- Fine-grained sanctions for violation
- Fast and cheap conflict resolution mechanisms available
- Authorities are not external, no overruling
- Composition in terms of systems-of-systems

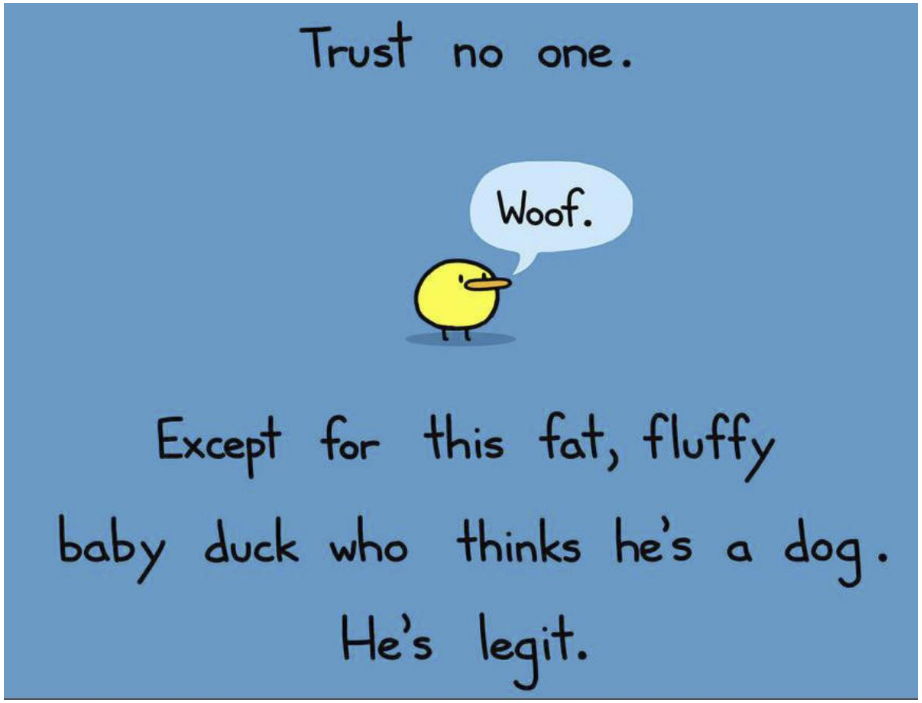
A prerequisite: trust

Goal: Establish such authorities!

- Must be trusted by all participants

Consequence:

- What is trust?
- How to incorporate trust in technical systems?


Trust no one.
Woof.
Except for this fat, fluffy baby duck who thinks he's a dog. He's legit.

Trust

“You can't trust any bugger further than you can throw him, and there's nothing you can do about it, so let's have a drink”

Terry Pratchett

Technical / computational trust

Trust

- Investigated by sociologists, biologists, psychologists, economists, and computer scientists.
- More than 70 definitions between 1960 and 2000!
- Common understanding that trust is the subjective expectation of a trustor that a trustee will perform a certai action.
  → Prediction based on the assessment of competence, willingness, and risk.
- Trust is a fundamental prerequisite for cooperation, while strong distrust hinders cooperation.
- Reputation: perception that an agent creates through past actions about its intentions and norms.

See: C. Castelfranchi and R. Falcone, Trust Theory - A Socio-Coginitive and Computational Model. John Wiley & Sons Ltd., 2010, ISBN: 9780470028759

Technical / computational trust (2)

What is “trust”?

- Trust is relative to a certain (class of) transaction(s)
  - A may trust B to drive his car but not to baby-sit
- Trust is a measurable belief
  - A may trust B more than A trusts C for the same transaction
- Trust is directed
  - A may trust B to be a profitable customer but B may distrust A to be a retailer worth buying from
- Trust evolves over time
  - The fact that A trusted B in the past does not in itself guarantee that A will trust B in the future. B’s performance and other relevant information may lead A to re-evaluate his trust in B

Building organisations based on trust

Christian-Albrechts-Universität zu Kiel

System composition

- We aim at enduring institutions
- System is – in general – open: join/leave at any time
- Unknown interaction partners: benevolent, malevolent?
- Whom to trust?

Trust mechanism and metric: See paper!

Approach:

- Establish trust and reputation system
- Steer autonomous entities by norms
- Sanctions and incentives have impact on trust and reputation

Normative systems

Motivation

- Important: Agents should/will keep their autonomy
- Requirement: No strict rules or constraints are possible

Approach: Establish norms!

- Norms specify desired behaviour, but may be broken.
- A norm encodes the desired behaviour (i.e. as a rule) and augments this with sanctions and incentives.
- If a norm violation is monitored the agent is sanctioned.
- Agent can weigh its own benefit from violating a norm against the risk of being sanctioned.

Norms

Christian-Albrechts-Universität zu Kiel

Norm

- May contain multiple policies (i.e. context and sanction / incentive).
- Context contains one or multiple conditions (must be true to trigger sanction).
- Sanction / incentive can be based on trust level (i.e. increase / decrease).
- Is valid for all agents, no individual steering possible!
- Incentive
  - Reputation is increased
  - Monetary incentives
- Sanction
  - Reputation is decreased
  - Loss of monetary incentives.

Norms (2)

Christian-Albrechts-Universität zu Kiel

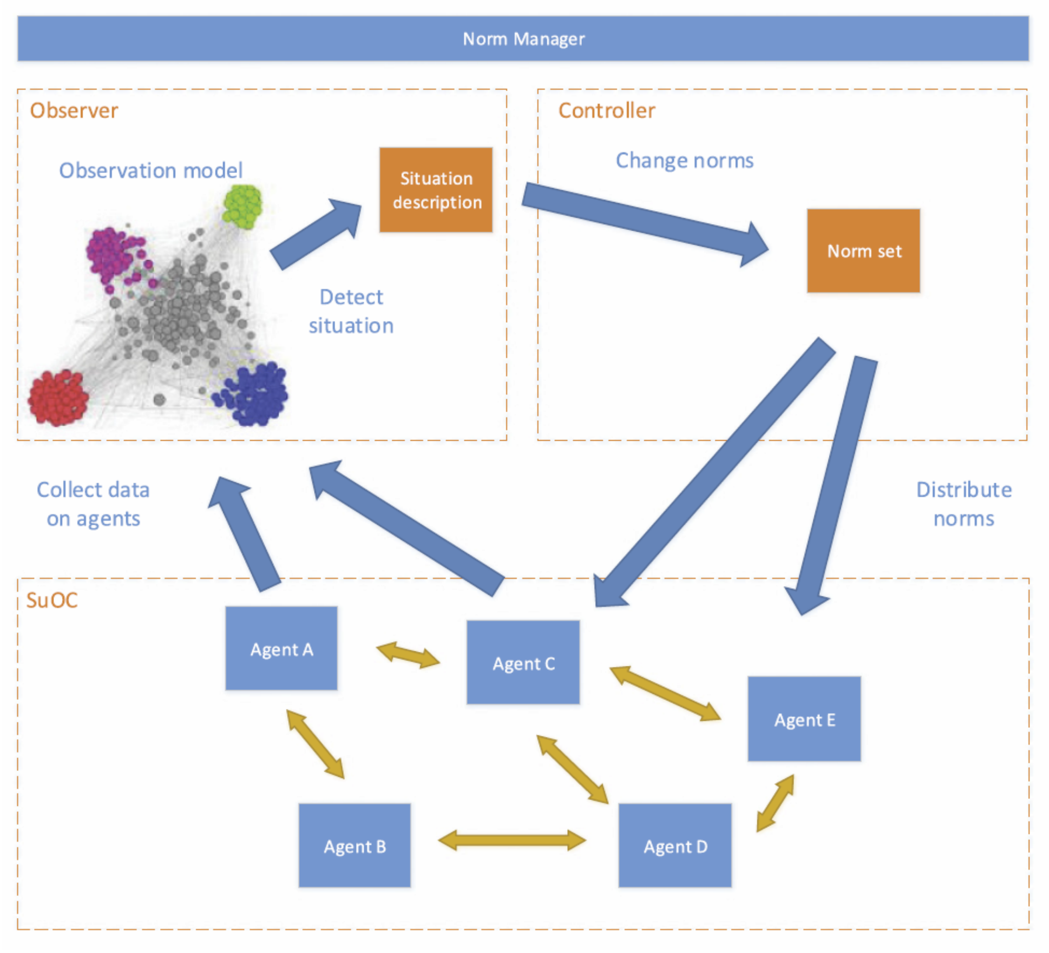
Example: Trusted Desktop Grid

$$\left.\begin{array}{l}\textbf{context}\ \ \overbrace{\text{Worker::returnJob()}}^{\text{Target/Role}}\ \ \textbf{inv}: \\ \qquad \underbrace{\texttt{self.returnJob(job, requester)} = \texttt{true}}_{\text{Postcondition}}\end{array}\right\}\ \text{Condition section}$$

$$\left.\begin{array}{l}\textbf{incentivised} \\ \quad \textbf{if}\ \ \underbrace{\texttt{requester.reputation} < \text{highReputationThreshold}}_{\text{Sanction Condition}} \\ \qquad \textbf{then}\ \ \underbrace{\text{self.reputation} \mathrel{+}= \text{highWorkDoneIncentivized}}_{\text{Incentive}} \\ \quad \textbf{if}\ \ \underbrace{\texttt{requester.reputation} \geq \text{highReputationThreshold}}_{\text{Sanction Condition}} \\ \qquad \textbf{then}\ \ \underbrace{\text{self.reputation} \mathrel{+}= \text{lowWorkDoneIncentivized}}_{\text{Incentive}}\end{array}\right\}\ \text{Incentivized section}$$

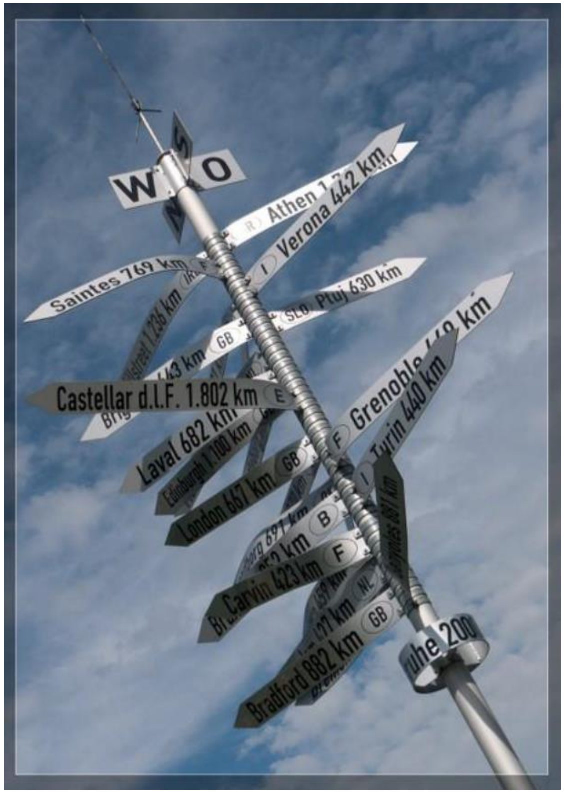
$$\left.\begin{array}{l}\textbf{sanctioned} \\ \quad \underbrace{\text{self.reputation} \mathrel{+}= \text{-cancelSanction}}_{\text{Sanction}}\end{array}\right\}\ \text{Sanction section}$$

Normative control



Agenda

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



Motivation

Structuring systems of systems

- Systems: grow larger, become more distributed, become highly heterogeneous and dynamic
- System engineers and administrators must also deal with non-functional properties:
  - scalability,
  - support for diversity,
  - detection and resolution of conflicting goals,
  - cope with openness and with a certain degree of unpredictable change
- Goal: (re)integrate simpler and smaller-scale self-* systems, sub-systems, and components into larger-scale, increasingly complex systems-of-systems; rather than (re)building them from scratch each time
- This system (re)integration process has to be carried-out dynamically and by the system itself; rather than performed offline and manually

Goal

Christian-Albrechts-Universität zu Kiel

Continuous system self-(re-)integration

- Self-integrate available resources
- Discover resources opportunistically
- Depending on goals
- Whenever change in internal resources, external context, or goals is detected
- Different OC systems:
  - potentially belonging to different authorities,
  - designed separately,
  - achieve different purposes,
  - are executed in shared environments,
  - interfere with each other in unforeseen ways
- Result: complex challenges
  - contention for resources
  - conflicting actions on shared resources

Challenges

Mastering such open collections raises challenges:

- Scalability: manageable complexity from recycled simplicity.
- Diversity for reusability, adaptability and robustness: able to support the interconnection and co-existence of diverse sub-systems, with specific architectures, control policies, self-* processes, configurations and technologies.
- Interference and conflicts: sub-systems that can achieve their objectives in isolation, may disturb each-other when coupled too tightly.
- Abstracting system descriptions
- Time-tuning self-* processes.

Goals as reference points

Reference point:

- When everything within and without an OC system may change, from its internal resources and integration architecture to its external environment, what is the reference?
- Here: Stakeholders' goal (i.e. purpose of the system)
- Stakeholder: human developers, owners, users, administrators or other systems.
- Definition: “A goal is an evaluable property that should be achieved, or a verifiable statement that can be deemed true (or not), of a state or behaviour of a system under consideration.”
- Examples: utility function, user requirements, constraints, etc.

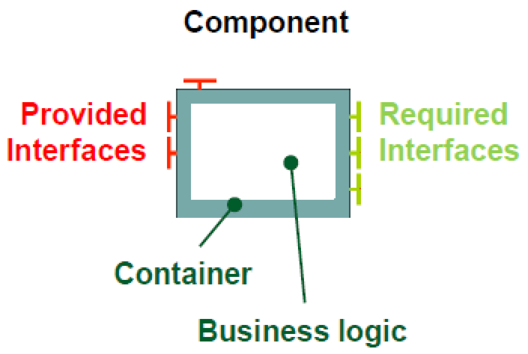
“Classic” Software Engineering

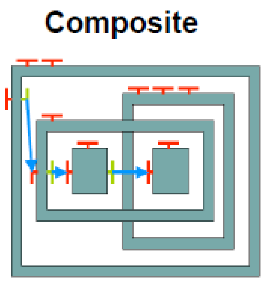
Divide & conquer

- Component-oriented Software (COS)
  - Encapsulation
  - Limited access through interfaces
  - Containers for non-functional services
  - Composition: “flat” or via encapsulation
- Service-oriented Architectures (SOA)
  - Dynamic service discovery & binding
- Goal-oriented Requirements Engineering (GORE)
  - Translate user goals into technical requirements

Limitations

- Wide abstraction gap: from goals to specific service integration
- Brittle: breaks easily when integrating unknown components
- Limited interaction types: reactive request (/reply)





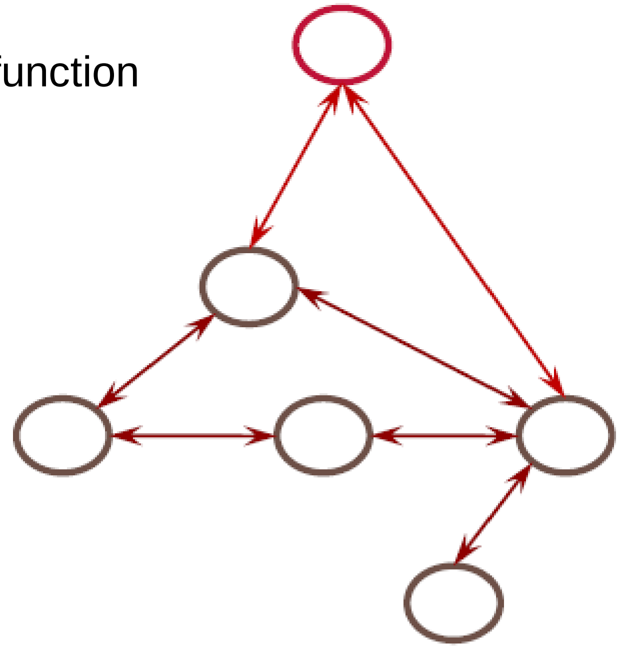
From OC to holonic intelligent systems

Support for dynamic change

- OC, AC (self-chop), SASO, CAS, …etc.
- Control loops for achieving some system property or function
- Coordination of control loops for achieving a global property or function

MAS, AI

- Goal-orientation, autonomy
- Organisations: roles & interrelations
- Dynamic binding of agents to roles
- Dynamic problem solving



Limitations

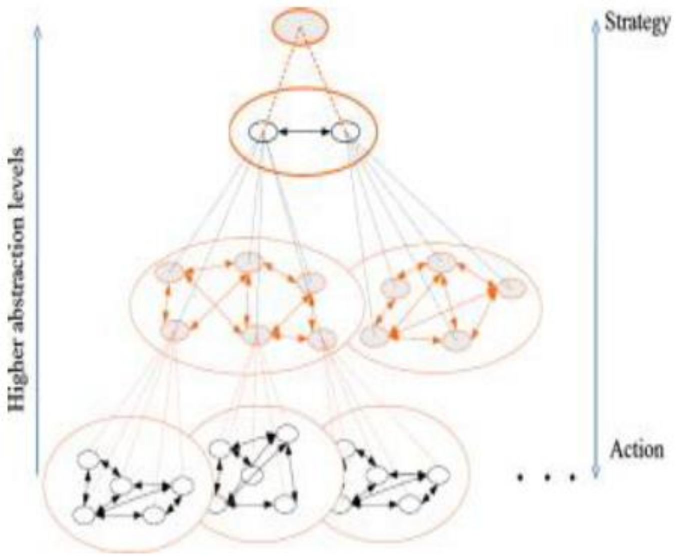
- Mostly for achieving a single goal (property or function)
- Difficult to deal with multiple goals, multi-scale goals
  → Need more support for (self-)integration of several self-* systems

Holonics

Manage limited rationality

Holonic architectures

- Encapsulated hierarchy: systems made of systems
  → For dealing with limited rationality [Simon62]
- Dual nature: semi-autonomous entities
  → For dealing with individualistic and transient [Koestler67]



Useful properties

- Controlled semi-isolation => limited state-space
  → Optimisation, stability, diversity
- Abstraction, aggregation
  → Progressive divide & conquer (limited rationality)
- Progressive dynamics
  → Avoids or limits chain reactions & oscillations

[Simon62] H. Simon, "The architecture of complexity". Proc. Am. Philos. Soc. 106(6), 467–482, (1962)
[Koestler67] A. Koestler, "The Ghost in the Machine", Hutchinson Publisher, London (1967)

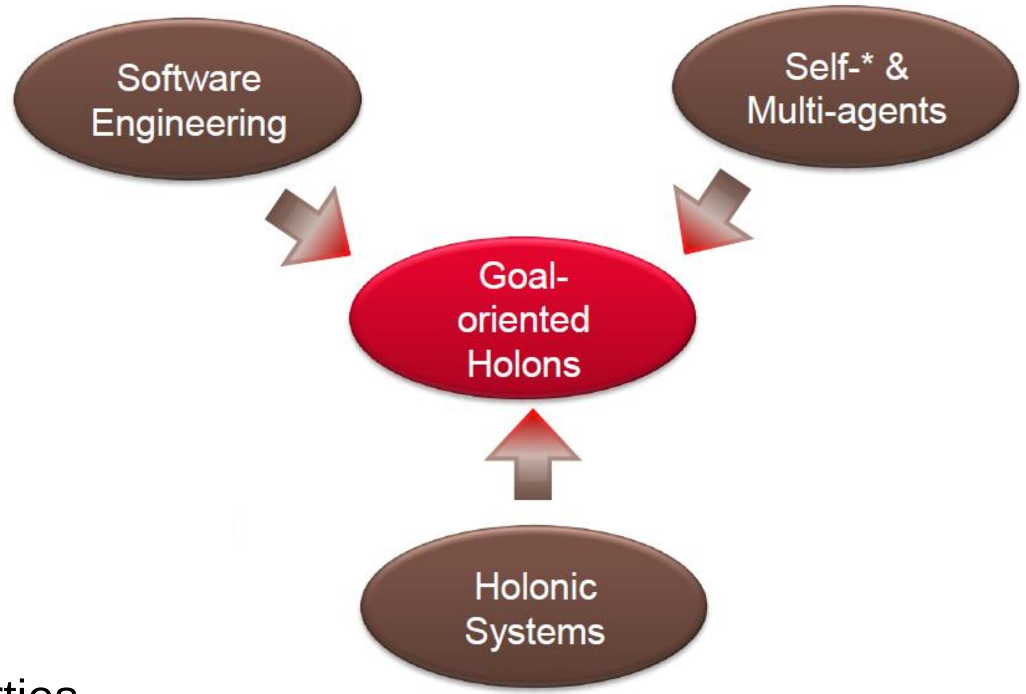
Holonics (2)

Limitations

- New field
- Little software engineering support
- Only limited experiences
- Application scenarios?

→ Need support for achieving these properties in engineered (self-)integrating systems.

Goal-oriented holonic architectures



Main features:

- Holonic design and properties
- Self-* functions encapsulated as components / services
- Goal-oriented interactions

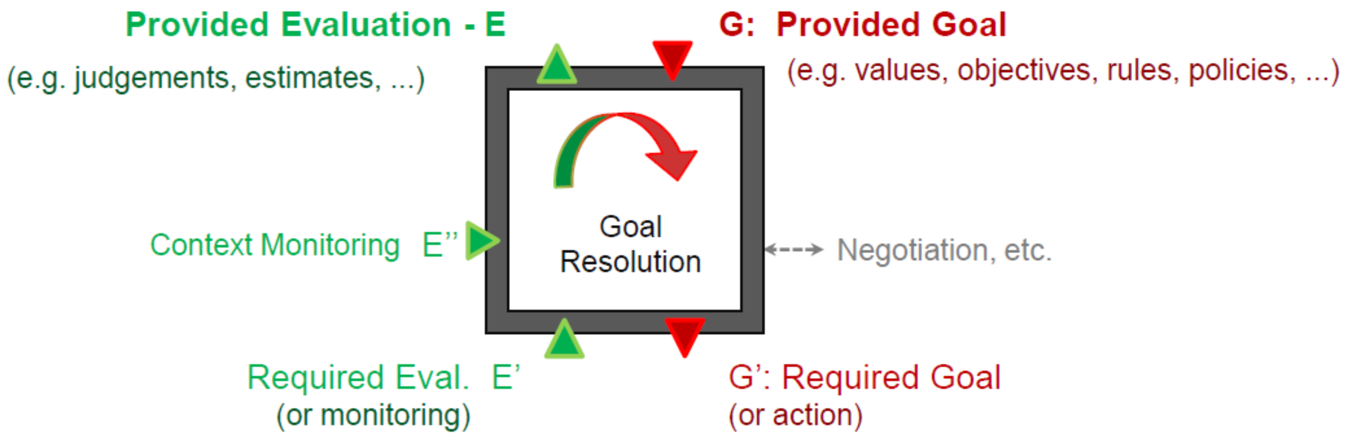
Abstract design of a goal-oriented holon

Christian-Albrechts-Universität zu Kiel

“Classic” Component-oriented SE → Goal-oriented holonic SE

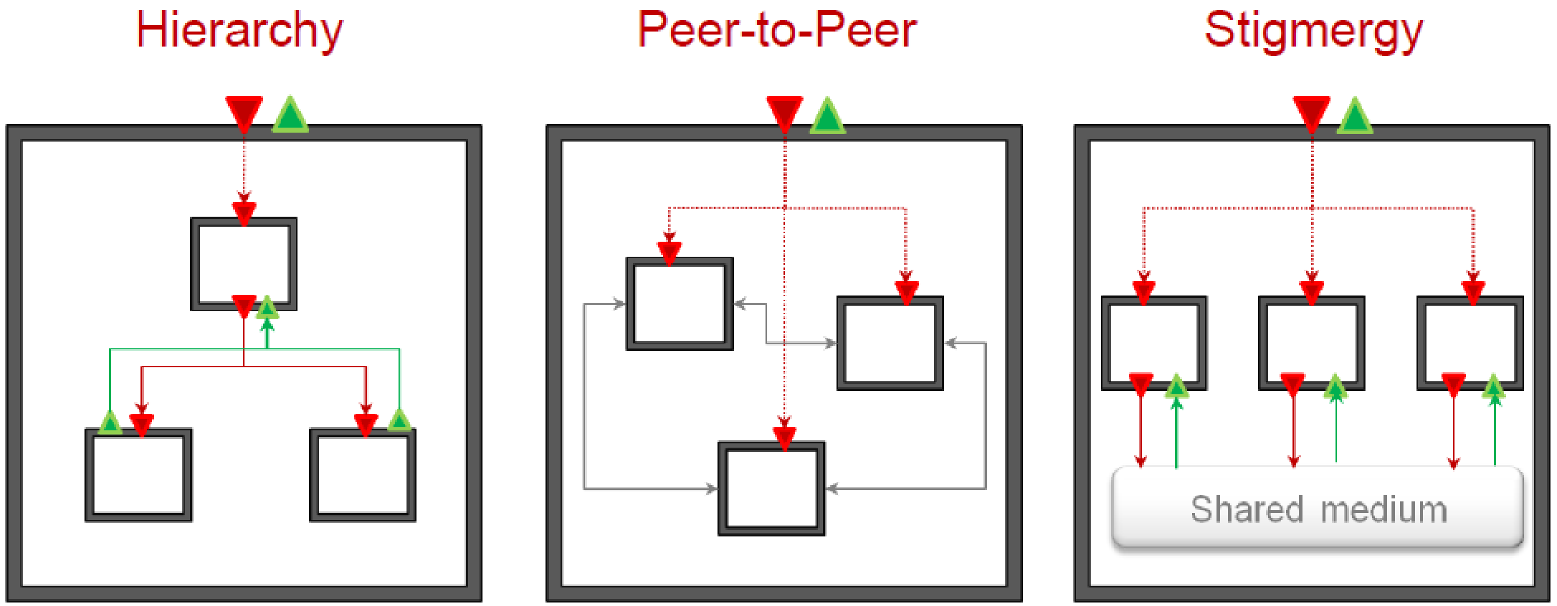
• Interfaces	→ Goals
• Request / Reply methods	→ Request / Evaluate goals
• Dynamic service binding	→ Dynamic Goal matching
• Container	→ Membrane

To Observer … From Controller …



Integration of goal-oriented holonics

- Goal matching, request / reply, execution / evaluation
- Integration patterns



Goal definitions

Christian-Albrechts-Universität zu Kiel

Basics of goal definitions

- Common elements:
  - V: Viability Domain & Evaluation Function → What?
  - SR: Resource Scope → Where?
  - ST: Time Scope → When?

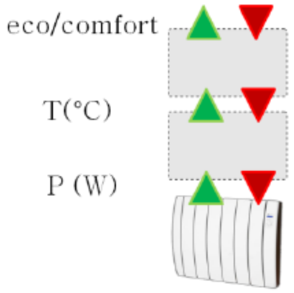
- Details are application-specific and abstraction level-specific

- Examples for the smart house domain:
  - G_{Cmf} = (Comfort, home, forever)
  - G_{Tmp} = ([T_{min}- T_{max}], rooms, intervals)
  - G_{Ph} = (P, heater, interval)

Goal operations

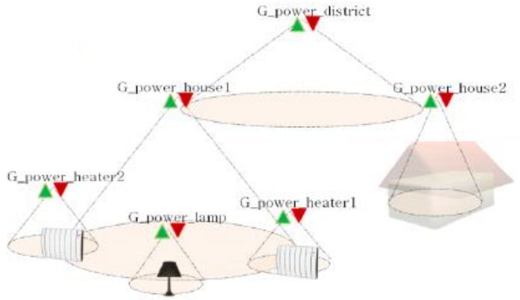
Translation & Inverse-translation: Change the type of a goal element

- E.g. V: Comfort → Temperature → Power
- E.g. SR: All electric devices → All TVs and all heaters



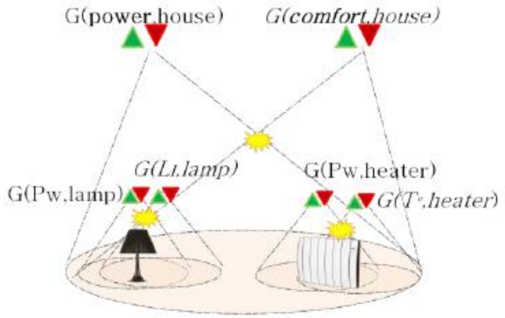
Splitting & Composition: Change the values of a goal element

- E.g. SR: District → each house → each device
- E.g. ST: Forever → sequential time intervals

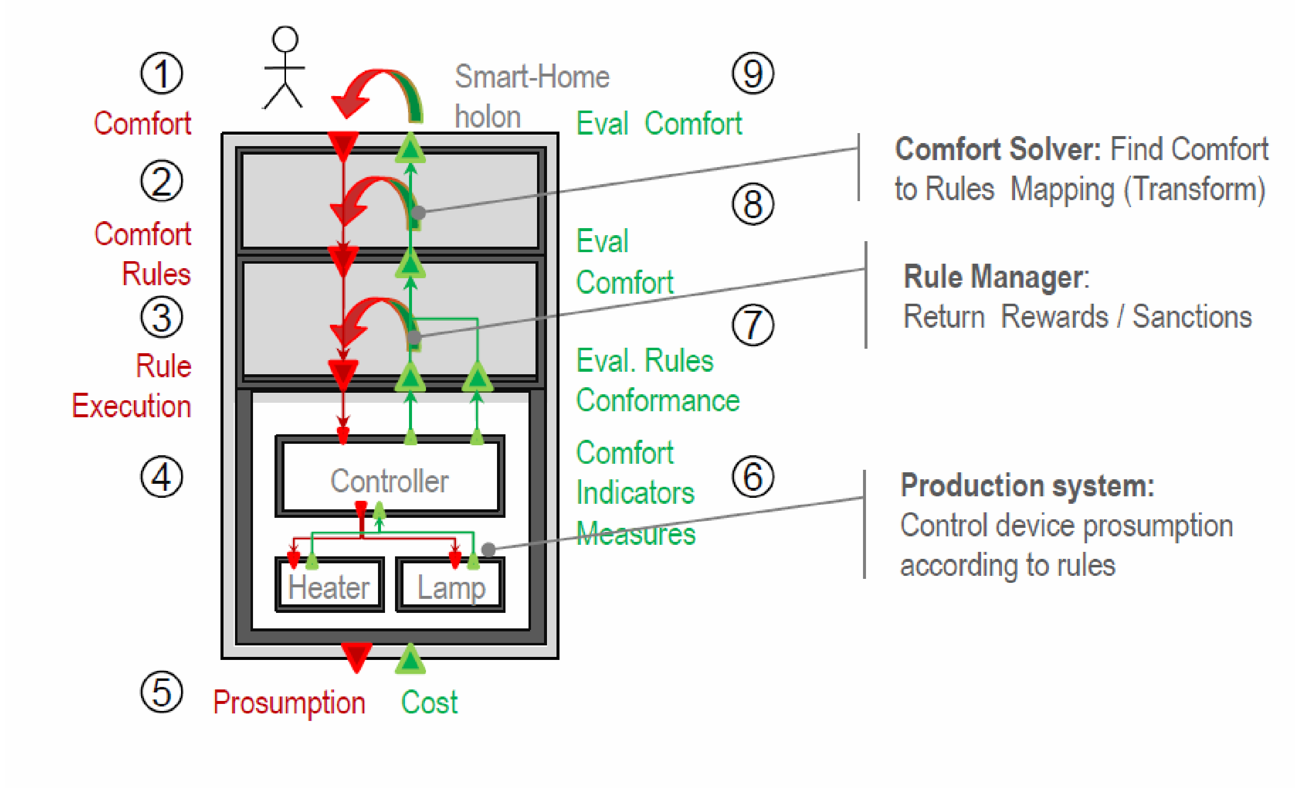


Composition can lead to Conflicts: When goals with incompatible viability domains have scopes that intersect

- E.g. minimise power consumption & maximise comfort => Intersection over devices such as heaters and lamps



Example1: Multi-level translation of a single goal in a smart home

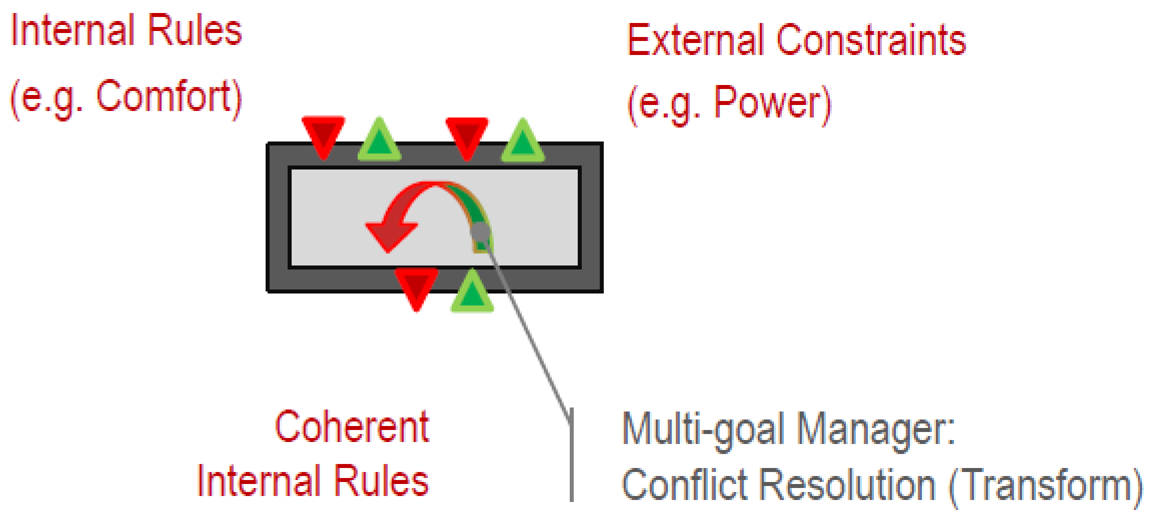


①
Comfort
②
Comfort Rules
③
Rule Execution
④
⑤
Prosumption
Cost
Smart-Home holon
⑨
Eval Comfort
⑧
Eval Comfort
⑦
Eval. Rules Conformance
Comfort Indicators Measures
⑥
Controller
Heater
Lamp
Comfort Solver: Find Comfort to Rules Mapping (Transform)
Rule Manager: Return Rewards / Sanctions
Production system: Control device prosumption according to rules

Typical issue addressed:

- Multi-layer translation from Goals to Rules to Rule Enforcement

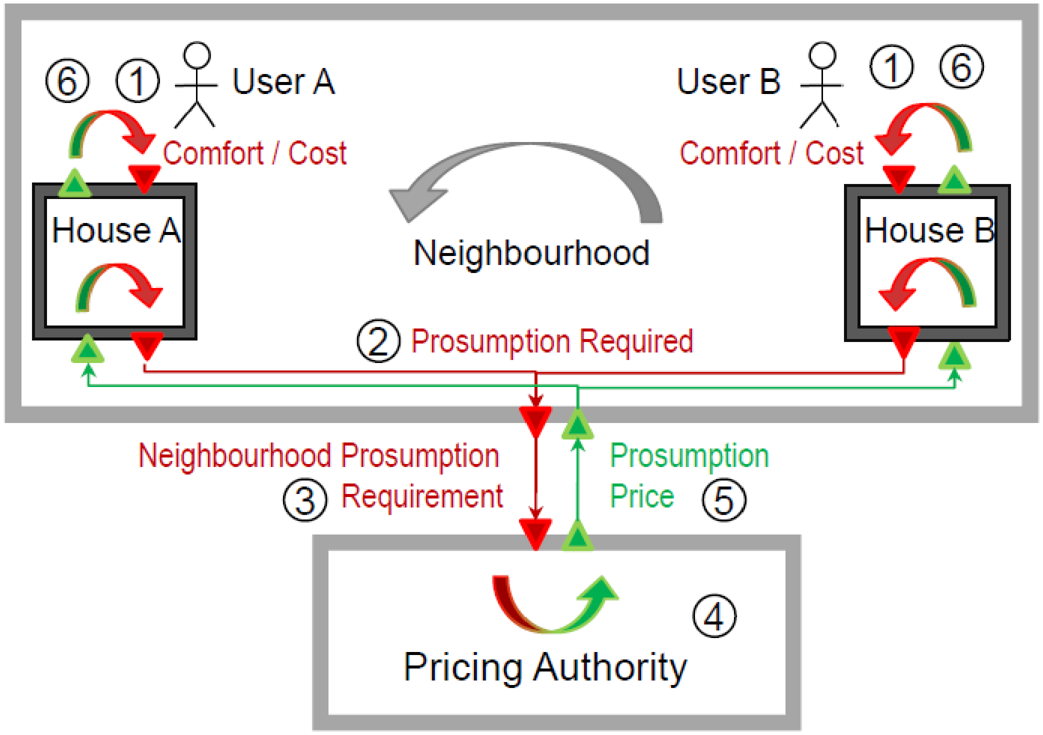
Example 2: Multi-goal conflict within a smart home



Typical issue addressed:

- Goal conflict resolution (application-specific implementation)

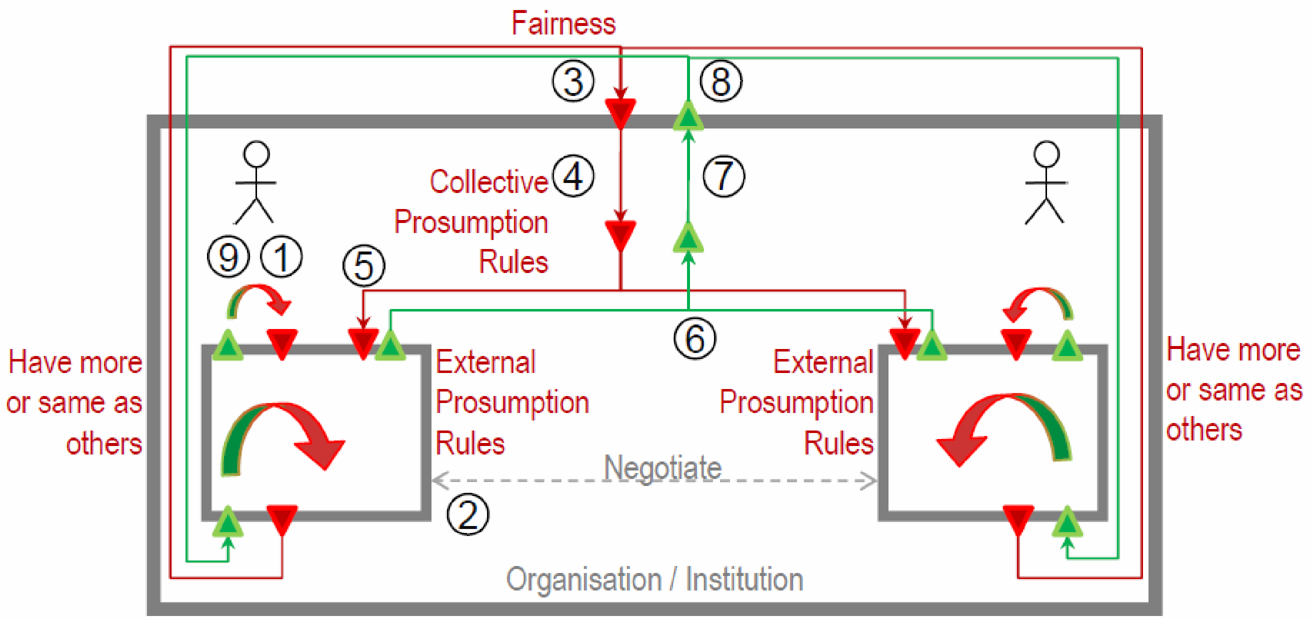
Example 3: Market-oriented energy distribution



Typical issue addressed:

- Top-down facilitation of bottom-up coordination

Example 4: Self-governing energy commons


Fairness
3
8
Collective Prosumption Rules
4
7
9
1
5
6
Have more or same as others
External Prosumption Rules
External Prosumption Rules
Have more or same as others
Negotiate
2
Organisation / Institution

Typical issue addressed:

- Bottom-up goal definition and top-down goal enforcement

Summary

A generic architectural model based on goal-oriented holons

- Goal formalisation → viability and scope
- Goal operations → translation and splitting
- Other interactions → negotiation, coordination, etc.
- Design patterns for conflict resolution →hierarchy, stigmergy, P2P
  => holonic integration
- Specific properties: controlled semi-isolation, abstraction and progressive dynamics

Intended use

- Understand, analyse and define the problem domain
- Design high-level system architecture, or structure
- Communicate, explain and discuss high-level solutions with peers

Summary (2)

Intended advantages

- Support for dynamic (self-) integration of self-* systems, addressing limited rationality
- Support for multi-goal, conflict-resolution, openness; diversity & plurality; better stability

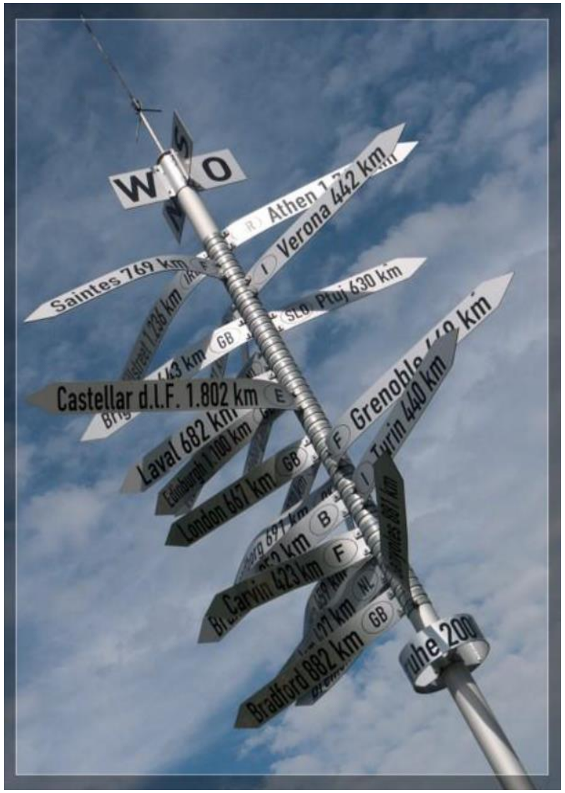
Where to from here?

- More research on specific aspects: membrane, patterns, conflicts, goal operations, yoyo dynamics,...
- Define a methodology for developing and evolving systems based on the proposed paradigm
- Develop reusable support: frameworks, tools, specification languages, IDEs, etc.

Agenda

Christian-Albrechts-Universität zu Kiel

- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



Conclusion

Christian-Albrechts-Universität zu Kiel

This chapter:

- Introduced the basic design ideas used in organic systems.
- Refined these concepts by explaining the Observer/Controller approach.
- Extended the scope towards enduring institutions within technical systems.
- Described socio-inspired concepts for mastering autonomous entities: normative system control based on trust
- Structured complex systems according to the idea of goal-oriented holonics.

By now, students should be able to:

- Outline the basic design concept used in OC.
- Sketch the Observer/Controller approach and the multi-level extension.
- Compare distribution variants.
- Explain the responsibilities of the contained components.
- Describe how a social-inspired system structure looks like.
- Summarise what goal-oriented holonics means.

Conclusion (2)

Christian-Albrechts-Universität zu Kiel

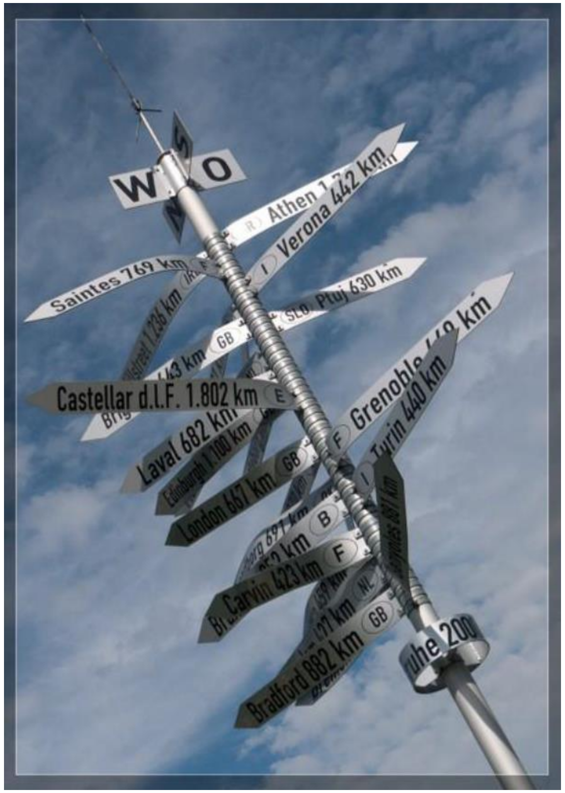
Design is just the basis!

- We will fill in the gaps in the next chapters
- Starting with the sensors and the process of gathering data / information
- Processing and analysing data
- Actually deciding about behaviour
- Learning and optimising this process over time

Agenda

Christian-Albrechts-Universität zu Kiel

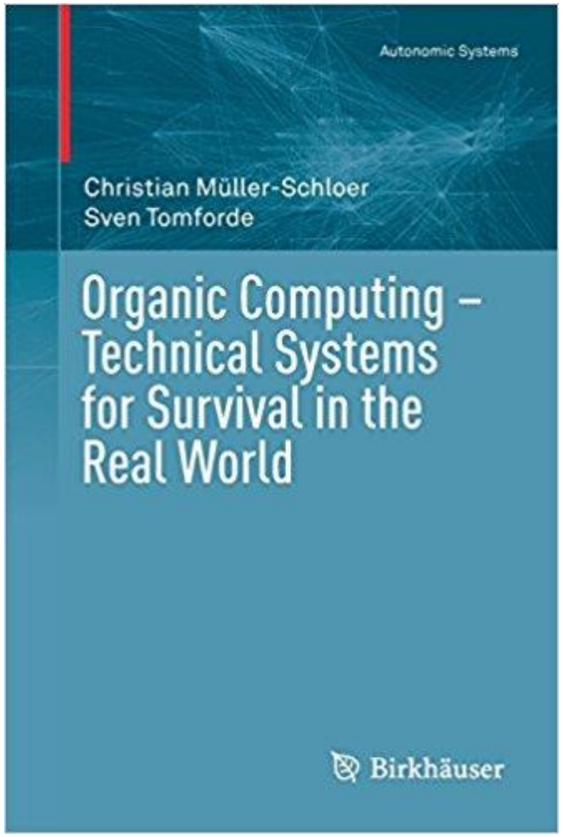
- General design concept for organic systems
- Observer/Controller architecture
- Multi-level Observer/Controller framework
- Application scenario: Trusted Desktop Grid
- Normative system control based on trust
- Goal-oriented holonics
- Conclusion
- Further readings



Further readings

"Organic Computing" and its characteristics are defined based on the book:

- Christian Müller-Schloer and Sven Tomforde: Organic Computing – Technical Systems for Survival in the Real World, Birkhäuser Verlag, Basel, 2018, ISBN 978-3319684765



Autonomic Systems
Christian Müller-Schloer
Sven Tomforde
Organic Computing – Technical Systems for Survival in the Real World
Birkhäuser

Further readings (2)

System architecture:

- Christian Müller-Schloer, Hartmut Schmeck, Theo Ungerer: „Organic Computing - A Paradigm Shift for Complex Systems“. Springer 2011
- Sven Tomforde: “Runtime adaptation of technical systems”. Südwestdeutscher Verlag für Hochschulschriften, ISBN-13: 978-3838131337, 2012.

Social Organic Computing and Trust:

- Sarah Edenhofer, Sven Tomforde, Jan Kantert, Lukas Klejnowski, Yvonne Bernard, Jörg Hähner, Christian Müller-Schloer: “Trust Communities: An Open, Self-Organised Social Infrastructure of Autonomous Agents“. In: Trustworthy Open Self-Organising Systems. Autonomic Systems Series, ISBN 978-3-319-29201-4, Birkhäuser, Basel, CH 2016

Holons:

- Ada Diaconescu, Sylvain Frey, Christian Müller-Schloer, Jeremy Pitt, Sven Tomforde: “Goal-oriented Holonics for Complex System (Self-)Integration: Concepts and Case Studies”. In: Proceedings of the 10th IEEE International Conference on Self-Adaptive and Self-Organising Systems, held September 12 – 16, 2016 in Augsburg, Germany. IEEE 2016

End

Christian-Albrechts-Universität zu Kiel

- Questions….?