HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Towards Smart Systems of Systems

6th IPM International Conference on Fundamentals of Software Engineering (FSEN 2015). 22 - 24 April, 2015. Tehran, Iran.

Holger Giese
System Analysis & Modeling Group,
Hasso Plattner Institute for Software Systems Engineering
University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de

Joint work with Basil Becker, Thomas Vogel, Sebastian Wätzoldt

Outline

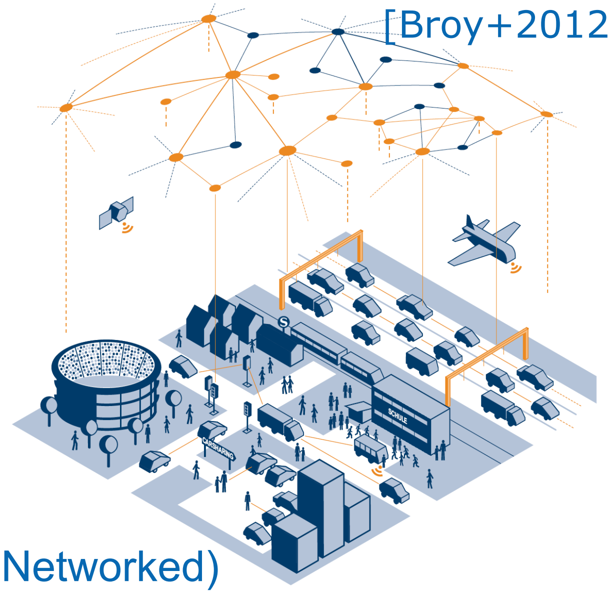
1. **Challenges Ahead**
2. **Available Options**
3. **SMARTSOS**
4. **Conclusions & Outlook**

Outline

1. **Challenges Ahead**
2. **Available Options**
3. **SMARTSOS**
4. **Conclusions & Outlook**

System of Systems

[Broy+2012]

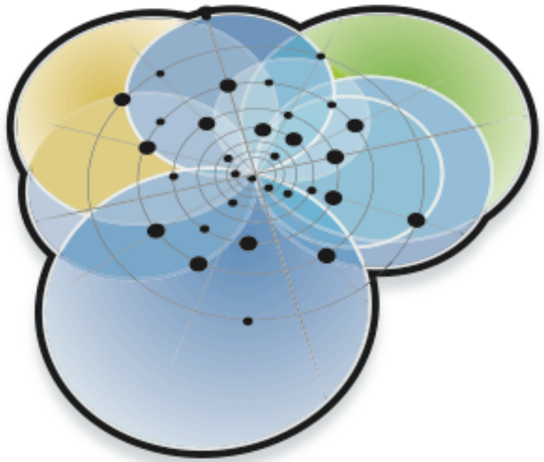


(Networked)
Cyber-Pyhsical Systems

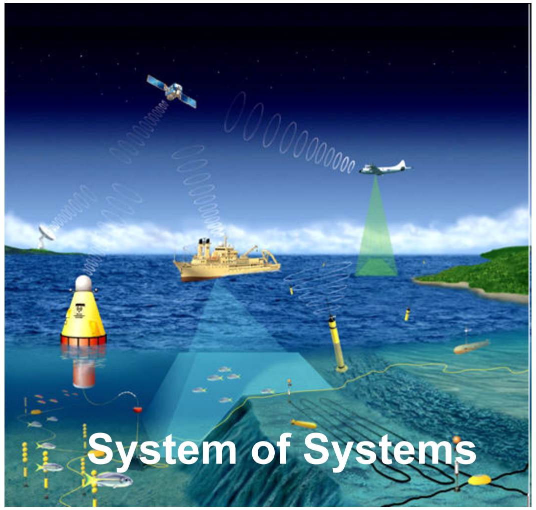
Internet of Things

Smart City

[Northrop+2006]



Ultra-Large-Scale Systems



http://oceanservice.noaa.gov/news/weeklynews/nov13/ioos-awards.html

Smart Factory -
E.g. Industry 4.0

Smart Logistic

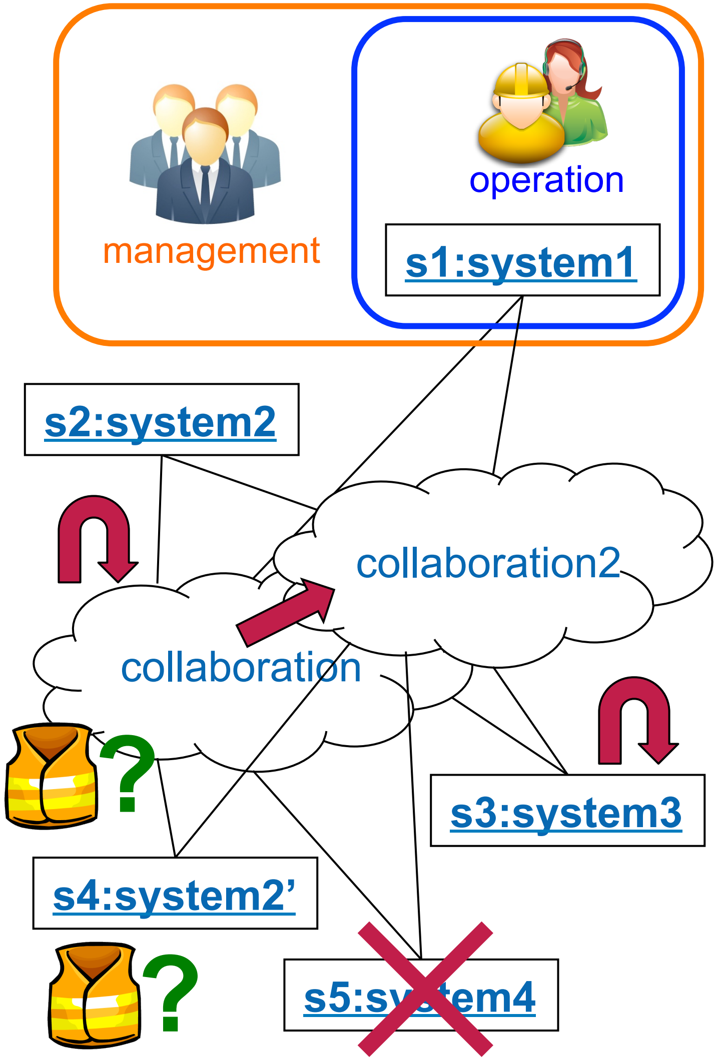
Micro Grids

Smart Home

E-Health

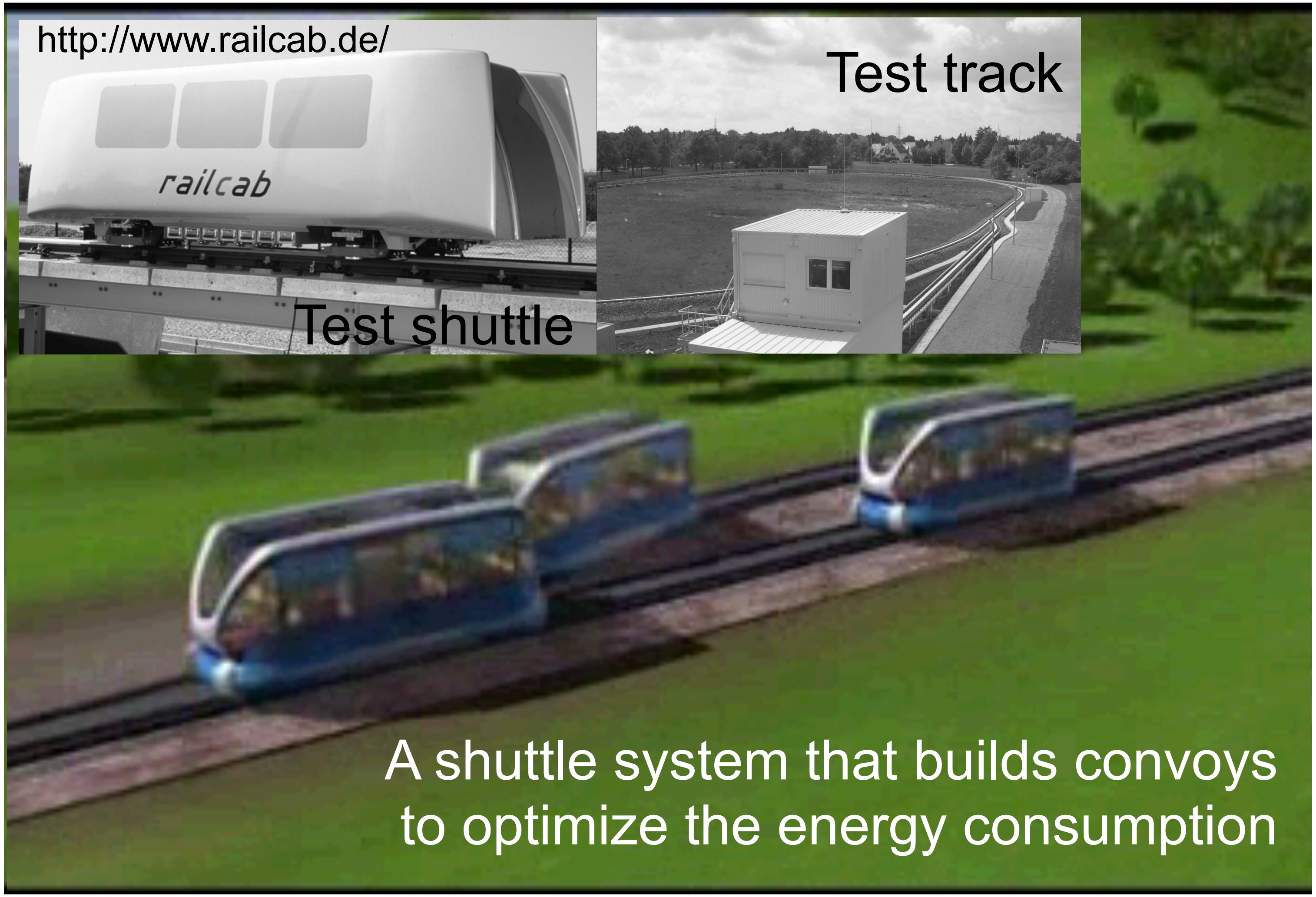
Ambient
Assisted Living

What characterizes a System of Systems?



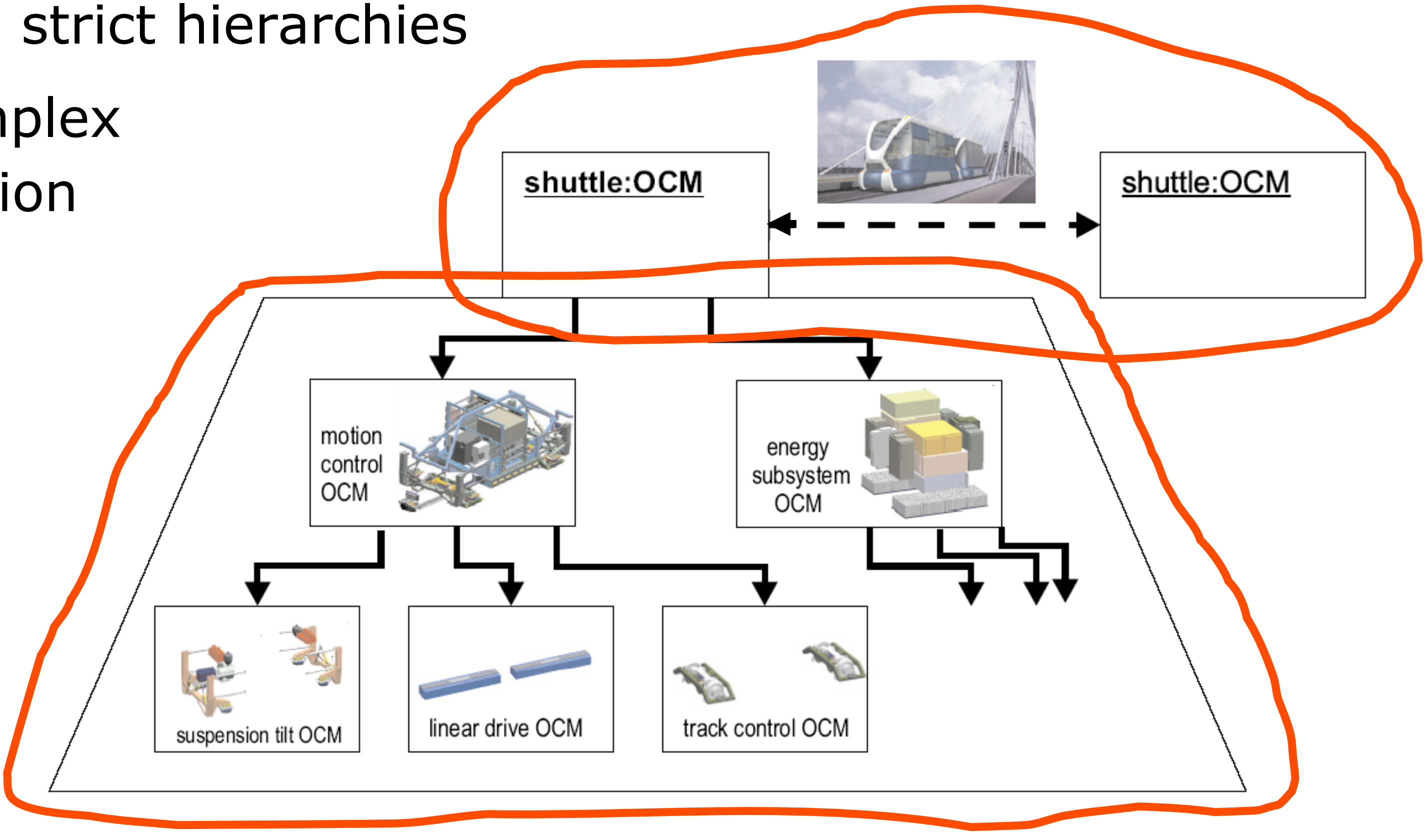
- **Operational** and **managerial independence**
  - operated independent from each other without global coordination
  - no centralized management decisions (may be conflicting)
- **Dynamic architecture** and **openness**
  - must be able to dynamically adapt/ absorb structural deviations
  - subsystems may join or leave over time in a not pre-planned manner
- **Advanced Adaptation**
- **Evolution**
- **Resilience**

RailCab Example: A Short Video …



System and SoS-Level Architectures

- **System of autonomous systems (shuttles)**
- Systems: strict hierarchies
- SoS: complex coordination



Challenge: Operational and Managerial Independence

"A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties:

- **Operational Independence** of the Components: If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. That is, the components fulfill customer-operator purposes on their own.
- **Managerial Independence** of the Components: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems." [Maier1998]

➔ We can only use **Restricted Knowledge**

Challenge: Dynamic Architecture and Openness

"The sheer scale of ULS systems will change everything. ULS systems will necessarily be **decentralized** in a variety of ways, developed and used by a wide variety of stakeholders with conflicting needs, **evolving continuously**, and constructed from heterogeneous parts."
[Northrop+2006]

"The vision of Cyber-Physical System (CPS) is that of open, ubiquitous systems of coordinated computing and physical elements which interactively adapt to their context, are capable of learning, dynamically and automatically reconfigure themselves and **cooperate with other CPS** (resulting in a compound CPS), possess an adequate man-machine interface, and fulfill stringent safety, security and private data protection regulations." [Broy+2012]

➔ We have to enable **SoS-Level Self-Organization**

➔ We have to enable **SoS-Level Structural Dynamics**

➔ We require means for **Runtime Knowledge Exchange**

Challenge: Advanced Adaptation

"**Adaptation** is needed to compensate for changes in the mission requirements […] and operating environments […]"

[Northrop+2006]

"The vision of Cyber-Physical System (CPS) is that of open, ubiquitous systems of coordinated computing and physical elements which interactively **adapt to their context, are capable of learning, dynamically and automatically reconfigure themselves** and cooperate with other CPS (resulting in a compound CPS), possess an adequate man-machine interface, and fulfill stringent safety, security and private data protection regulations." [Broy+2012]

➔ We have to enable **Self-Adaptation** for the systems

Challenge: Evolution

"The sheer scale of ULS systems will change everything. ULS systems will necessarily be decentralized in a variety of ways, developed and used by a wide variety of stakeholders with conflicting needs, **evolving continuously**, and constructed from heterogeneous parts."

[Northrop+2006]

Managerial Independence of the Components: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but **maintain** a continuing operational existence independent of the system-of-systems." [Maier1998]

➔ We have to enable independent **Evolution**

Challenge: Resilience

"The vision of Cyber-Physical System (CPS) is that of open, ubiquitous systems […] which […] and **fulfill stringent safety, security and private data protection regulations**." [Broy+2012]

"Resilience[:] This area is the attribute of a system, in this case a SoS that makes it less likely to experience failure and more likely to recover from a major disruption." [Valerdi+2008]

"Resilience is the capability of a system with specific characteristics before, during and after a disruption to absorb the disruption, recover to an acceptable level of performance, and sustain that level for an acceptable period of time." Resilient Systems Working Group, INCOSE

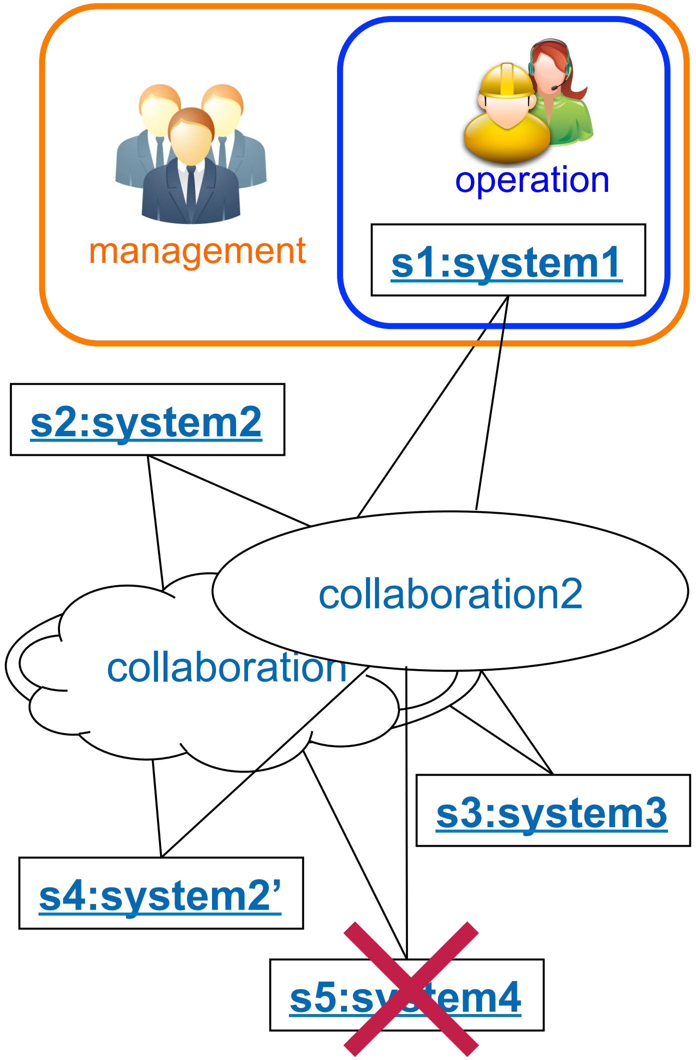
➔ We require **Resilience** for the SoS

Outline

1. Challenges Ahead
2. Available Options
3. SMARTSOS
4. Conclusions & Outlook

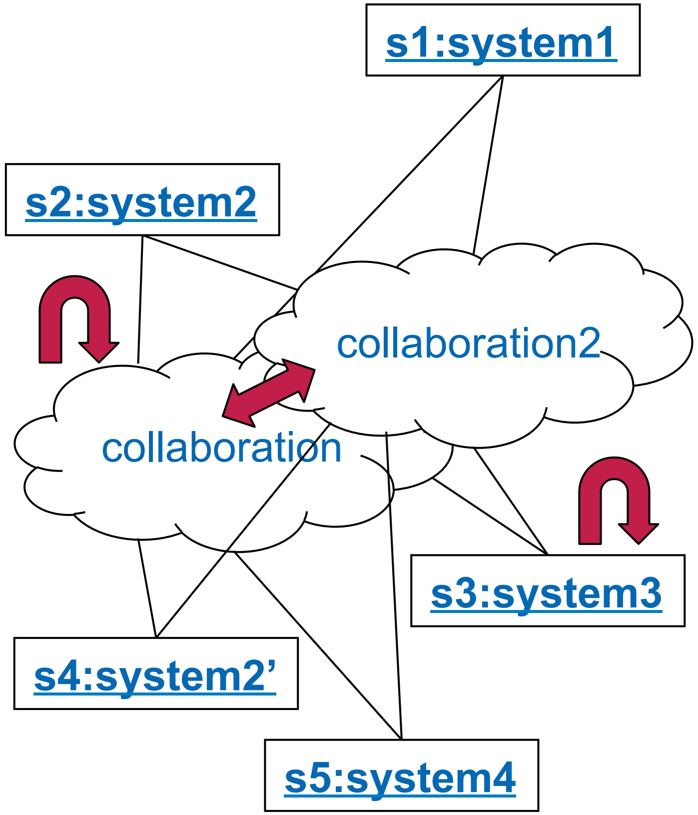
Option: Service-Oriented Architecture

Hasso Plattner Institut



- **Service-Oriented Architecture:**
  - Dedicated services are offered by systems via defined **service contracts** can be offered, looked up, and bound at run-time
  - Interoperability is provided by a **service bus**
- **Service oriented architecture Modeling Language (SoaML)**
  - a UML profile for modeling
  - Support **collaborations** as first class elements (service contracts)
  - Links collaborations with **component-based models**

Option: Self-Adaptive & Self-Organization



- **Self-Adaptive Systems:**
  - Make systems self-aware, context-aware, and requirements-aware using some form of **reflection**
  - Enable systems to adjust their structure/behavior accordingly
- **Self-Organization:**
  - The capability of a group of systems to organize their structure/behavior without a central control (emergent behavior)
- **Engineering perspective:**
  - a spectrum from centralized top-down self-adaptation to decentralized bottom-up self-organization with many intermediate forms (e.g. partial hierarchies) exists

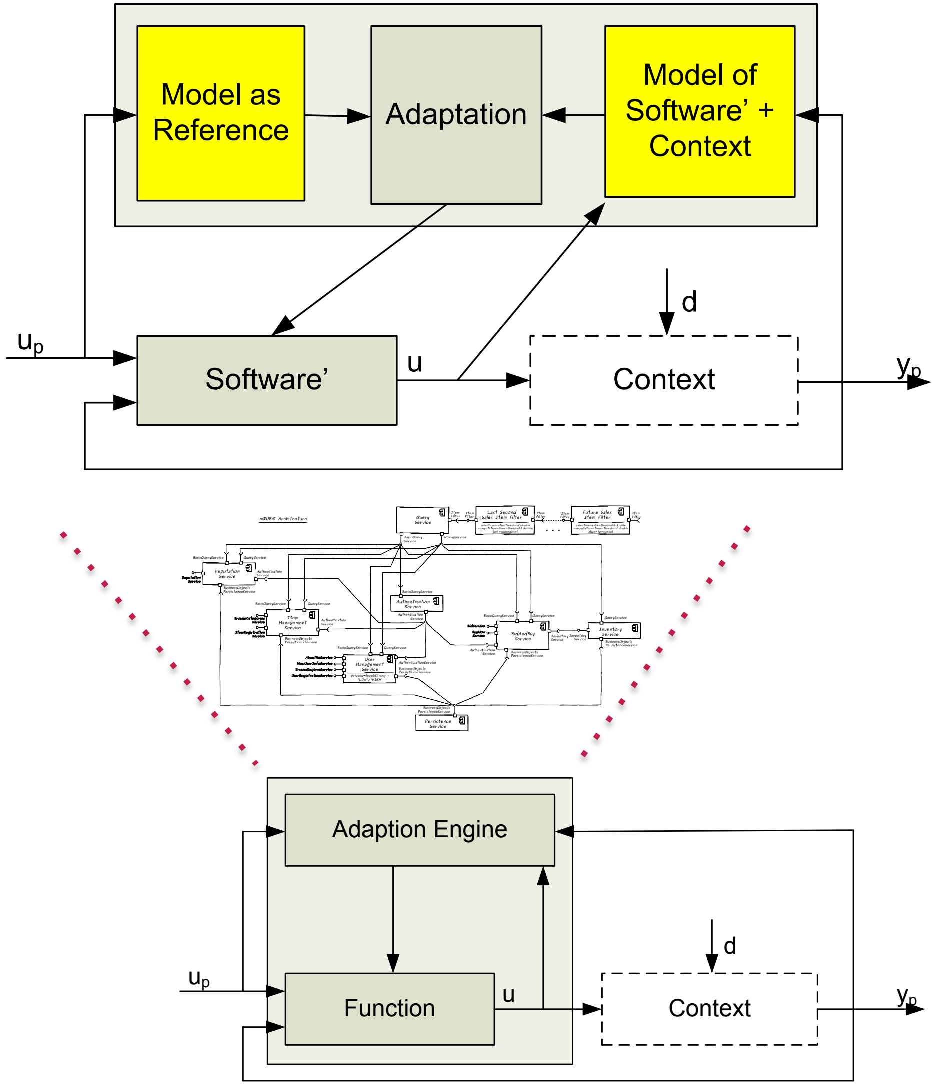
Option: Runtime Models

Runtime models:

- A causal relation between the software and/or context and the runtime model is contained
- Self-Adaptation can operate at a higher level of abstraction

Example:

- A generic approach for runtime models of the architecture running in EJB application servers



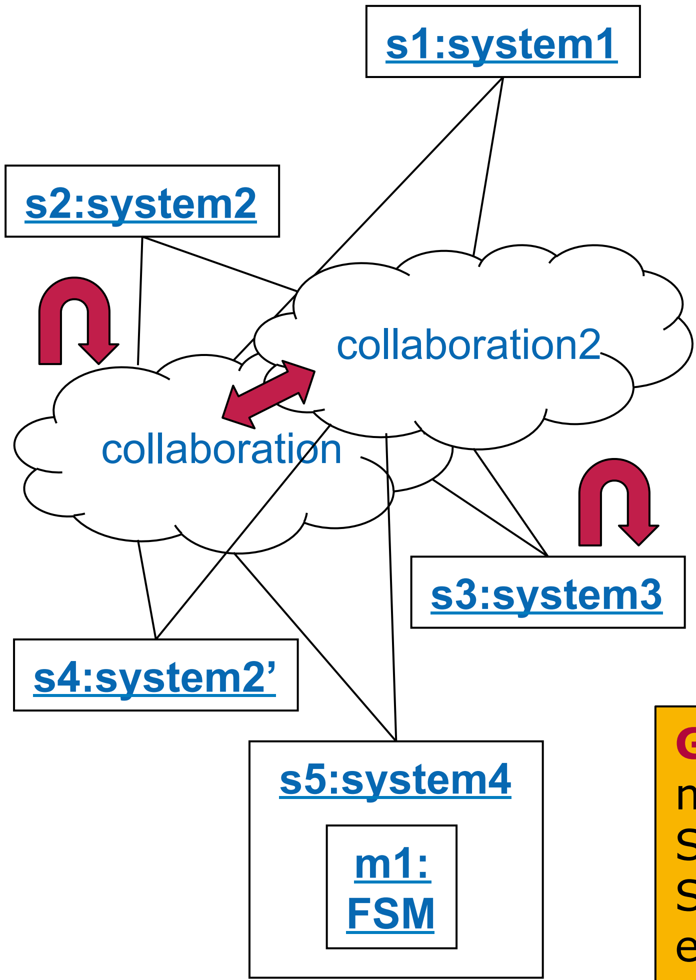
[Vogel+2009,Vogel+2010] [Vogel&Giese2012]

Outline

Hasso Plattner Institut

1. **Challenges Ahead**
2. **Available Options**
3. **SMARTSOS**
4. **Conclusions & Outlook**

SMARTSOS: Main Idea

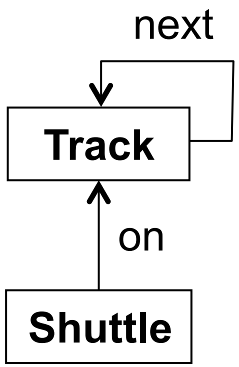


- **Service-Oriented Architecture** can be described by a **graph** of links between the systems that evolve
- **Self-Adaptive** and **Self-Organization** can be described by a **graph** of links between the components resp. systems that evolve/ reconfigure and in case of **reflection** most models can be described by such a **graph** as well
- **Runtime Models** can be described by a dynamic **graph** of models and links between them

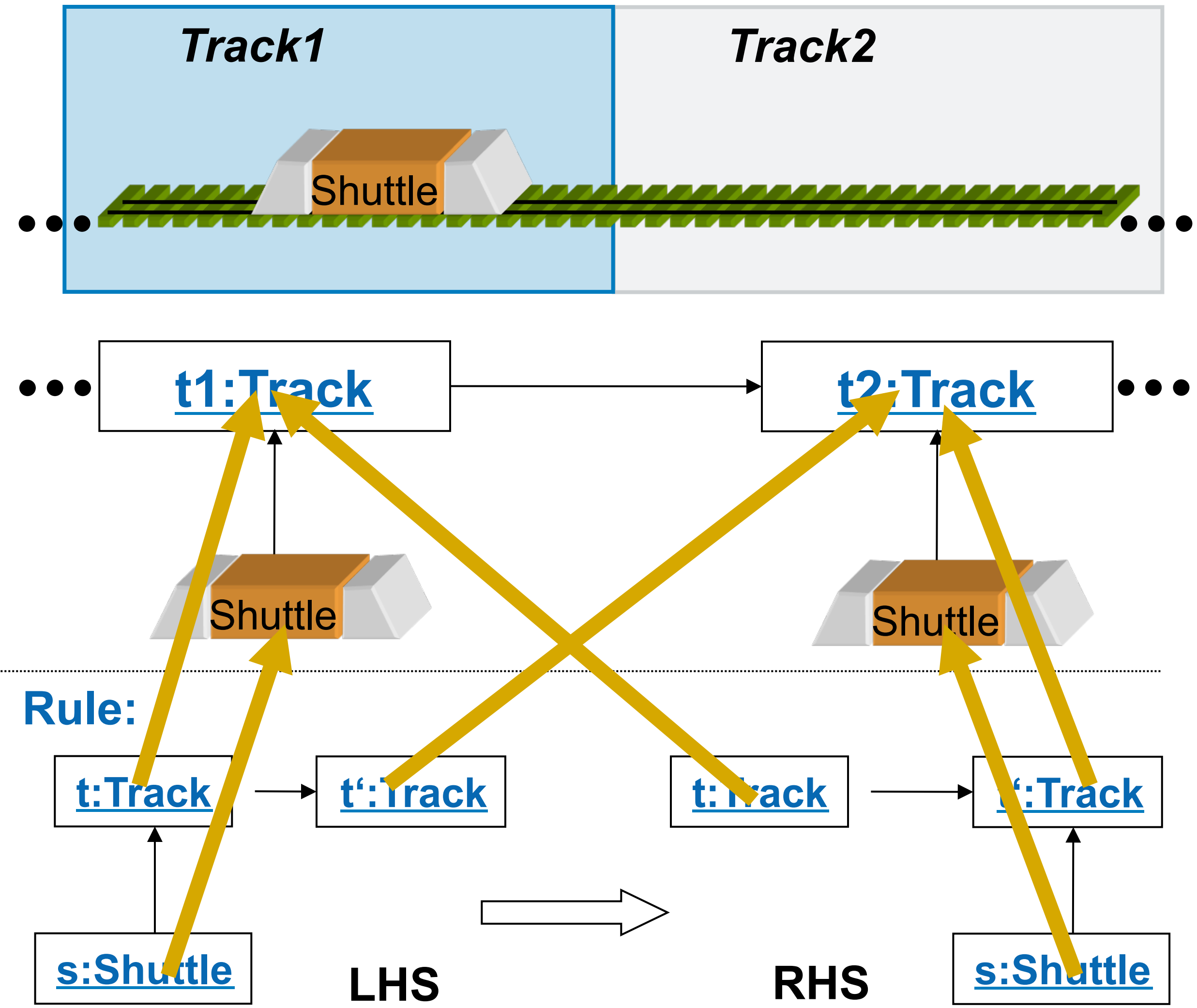
Graph transformation systems encoding models and their linking would allow to combine Service-Oriented Architecture, Self-Adaptive / Self-Organization, and Runtime Models with evolving structures and could be the basis for a **solid foundation** for **Smart SoS ...**

Graph Transformation Systems: Naïve Example

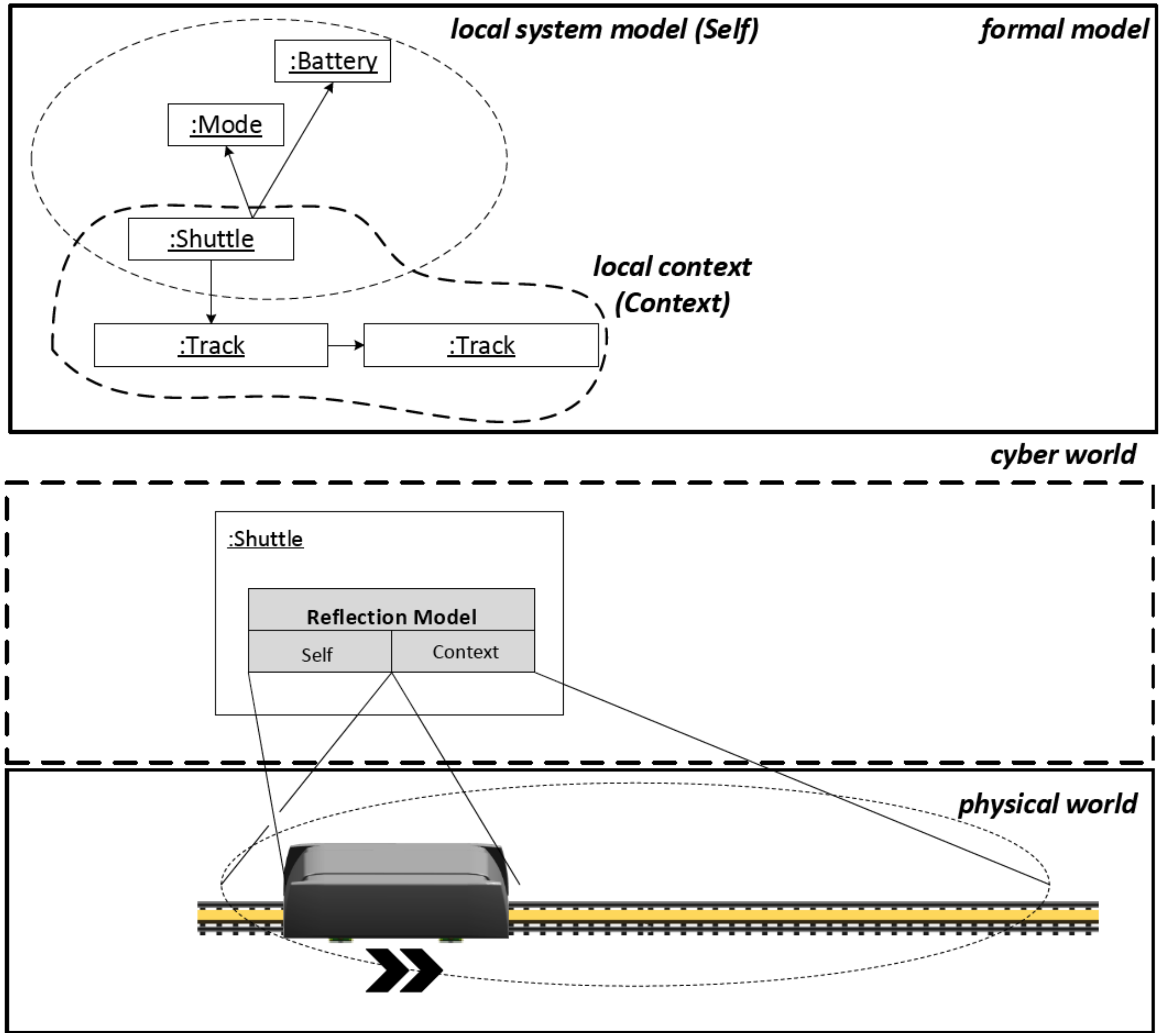
- Map the tracks
- Map the shuttles



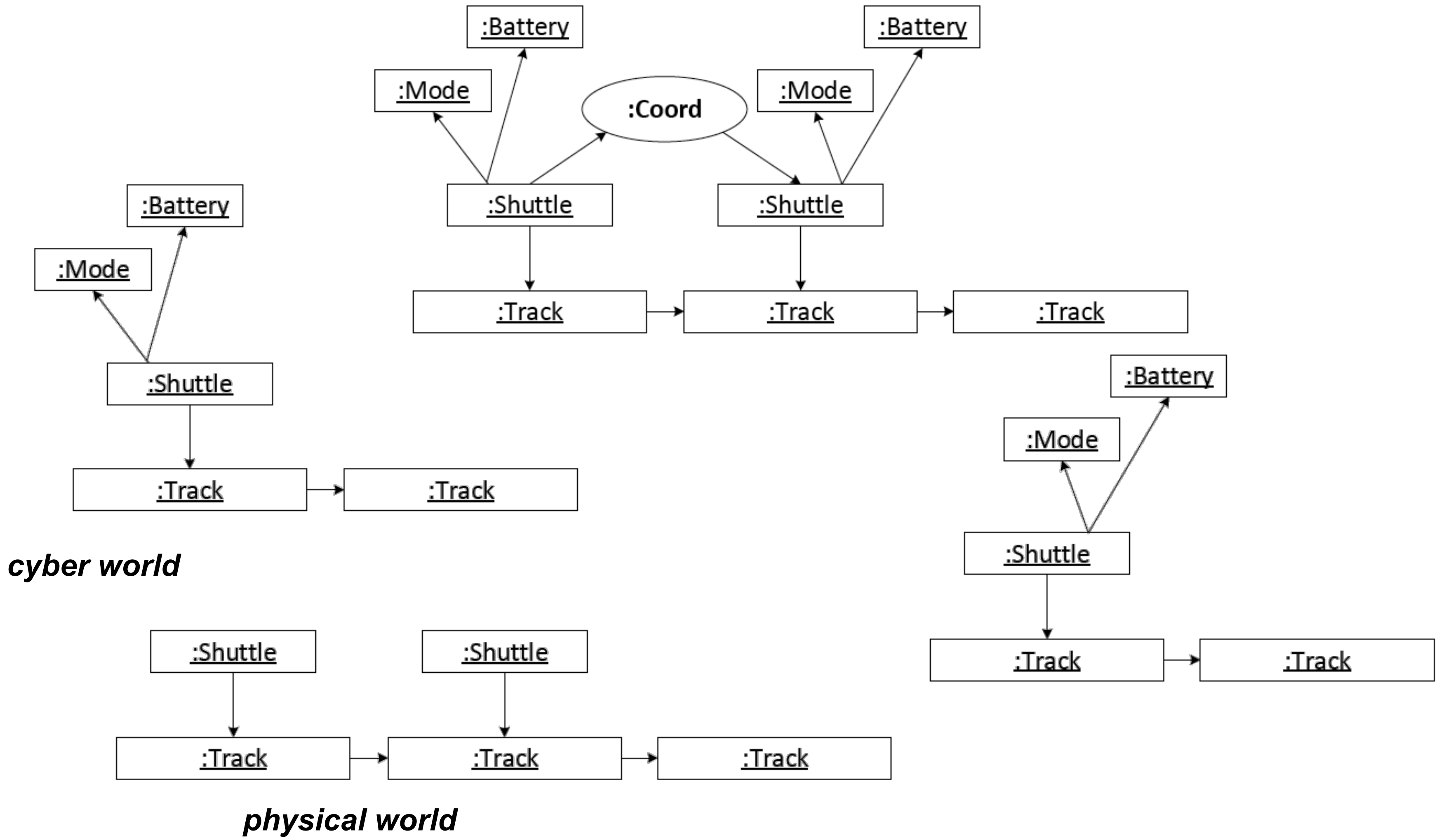
- Map the movement to rules (movement equals dynamic structural adaptation on the abstract level)



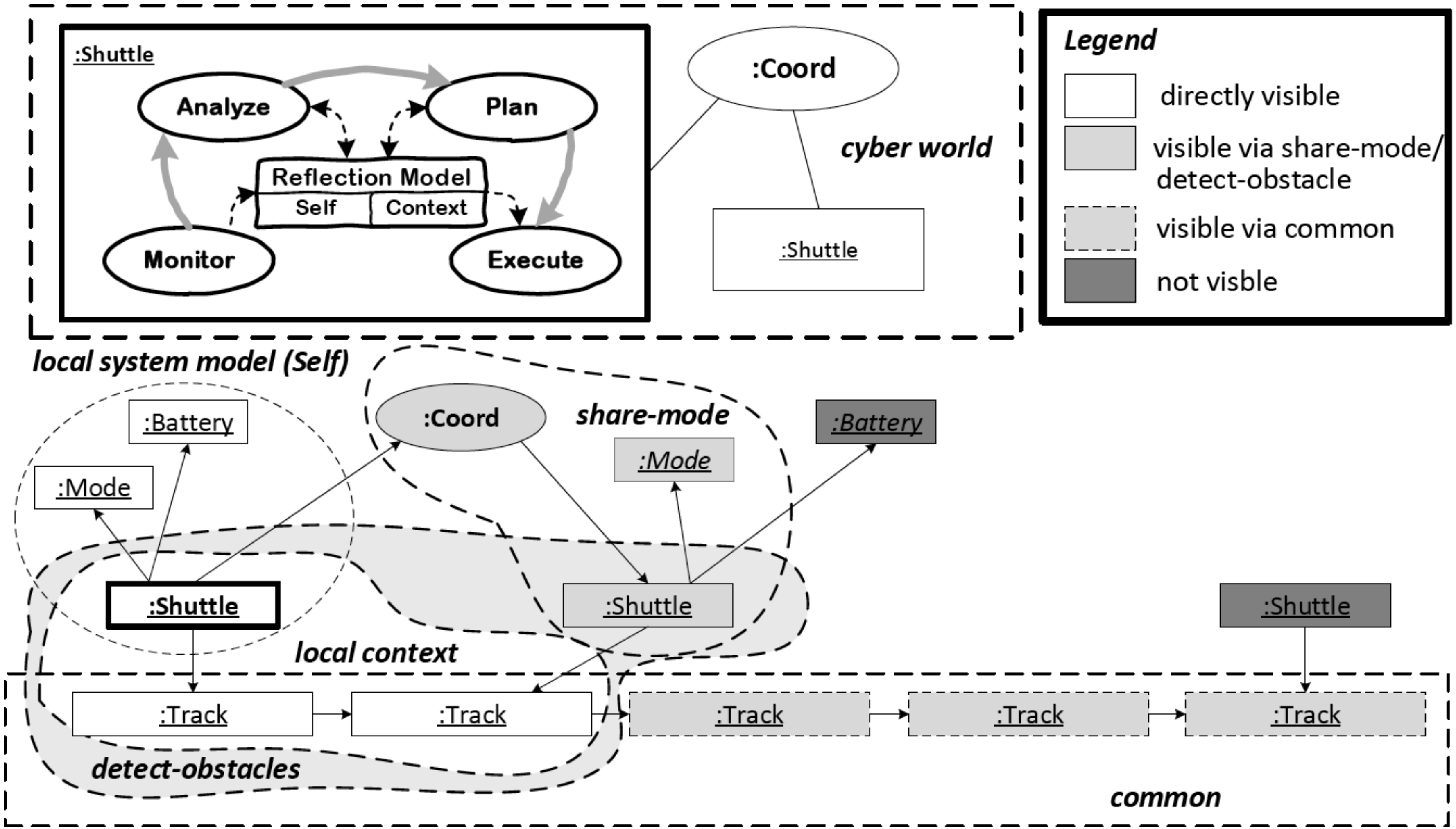
SMARTSOS: Main Idea

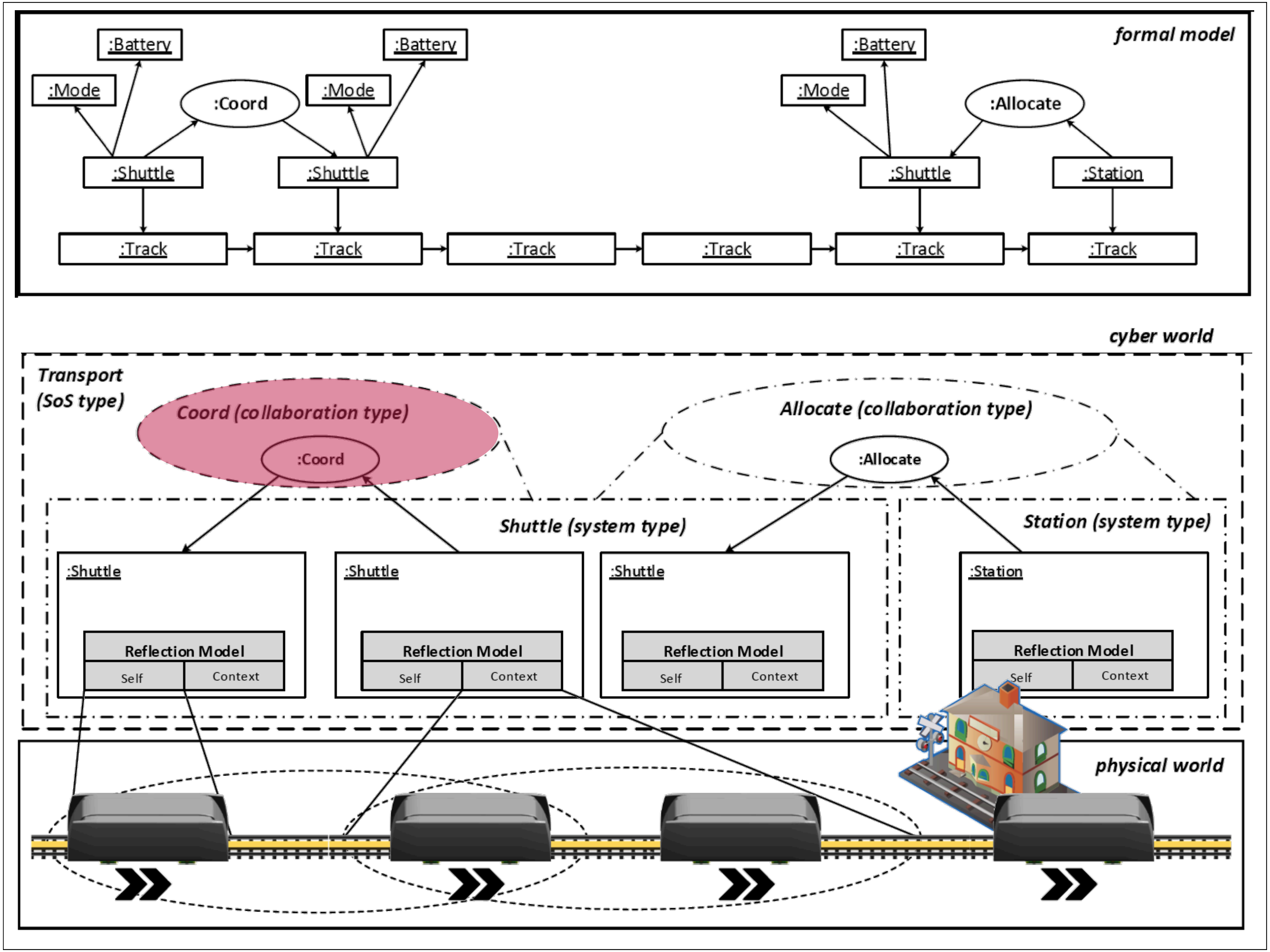


Consistency of Cyber & Physical World



Sharing Runtime Models & Visibility

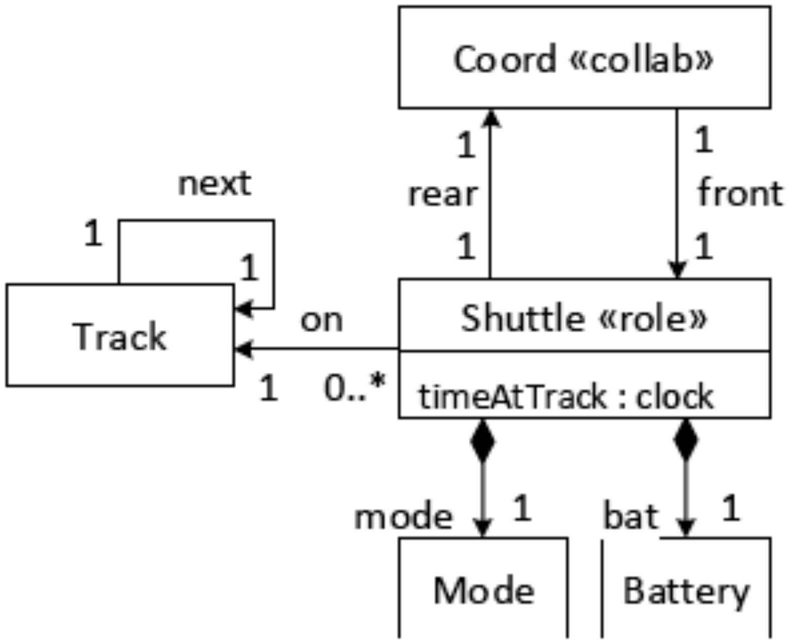




formal model
:Battery
:Mode
:Coord
:Allocate
:Shuttle
:Station
:Track
cyber world
Transport (SoS type)
Coord (collaboration type)
Allocate (collaboration type)
Shuttle (system type)
Station (system type)
Reflection Model
Self
Context
physical world

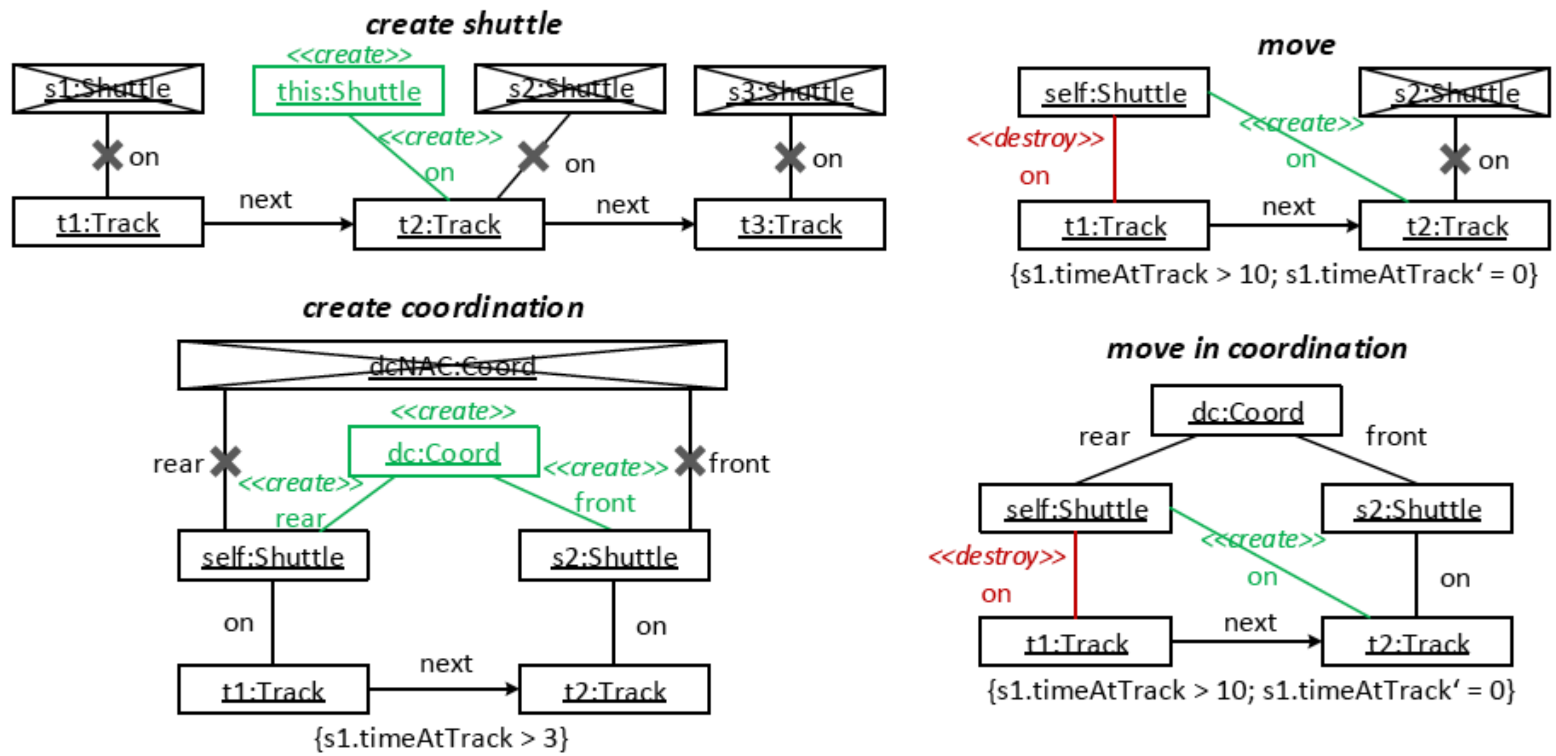
SMARTSOS: Collaboration Types

Definition 2 (see [54]). *A collaboration type* $\mathsf{Col}_i = (\mathsf{col}_i, (ro_i^1, \ldots, ro_i^{n_i}), CD_i, R_i, \Phi_i)$ *consists of a collaboration type node* col_i*, a number of role types* ro_i^j*, an UML class diagram* CD_i*, a function* $R_i : \{\mathsf{col}_i, ro_i^1, \ldots, ro_i^{n_i}\} \mapsto 2^{\mathcal{R}}$ *assigning rules to role types, and a guaranteed property* Φ_i*.*



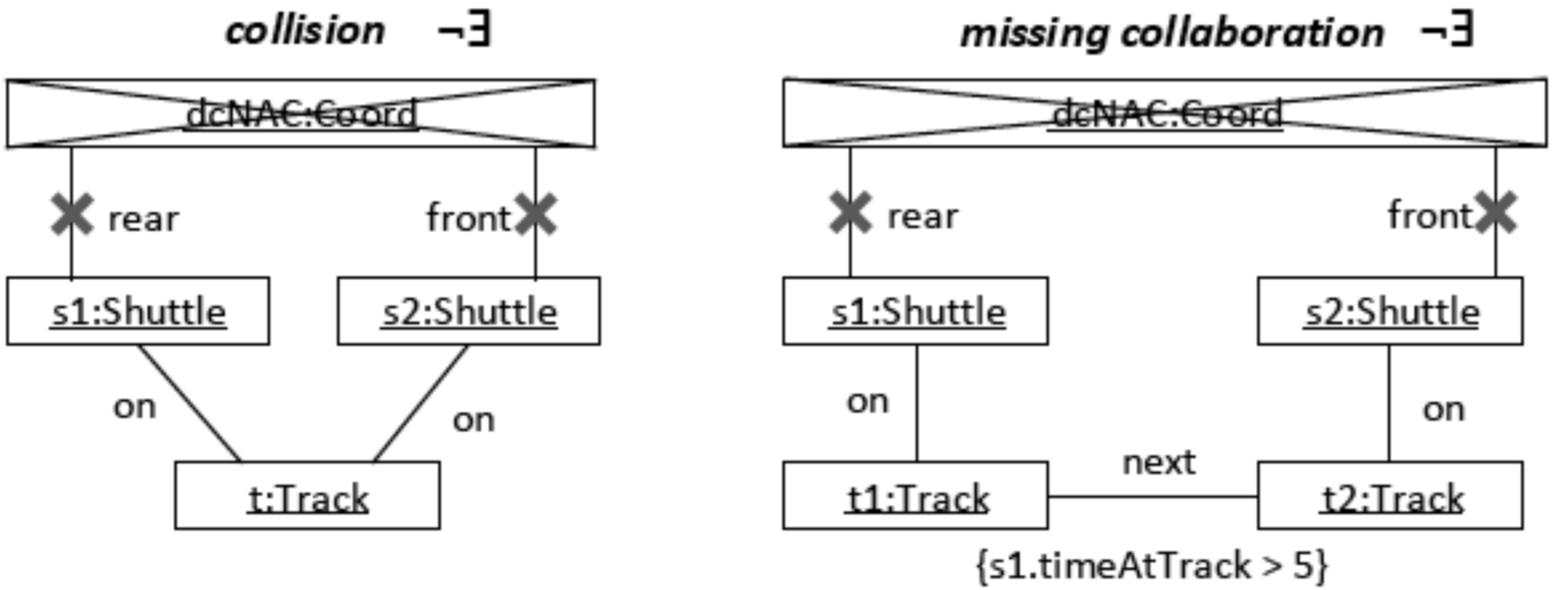
SMARTSOS: Collaboration Types

Definition 2 (see [54]). *A collaboration type* $\mathsf{Col}_i = (\mathsf{col}_i, (ro_i^1, \ldots, ro_i^{n_i}), CD_i, R_i, \Phi_i)$ *consists of a collaboration type node* col_i*, a number of role types* ro_i^j*, an UML class diagram* CD_i*, a function* $R_i : \{\mathsf{col}_i, ro_i^1, \ldots, ro_i^{n_i}\} \mapsto 2^{\mathcal{R}}$ *assigning rules to role types, and a guaranteed property* Φ_i*.*



SMARTSOS: Collaboration Types

Definition 2 (see [54]). *A collaboration type* $\mathsf{Col}_i = (\mathsf{col}_i, (ro_i^1, \ldots, ro_i^{n_i}), CD_i, R_i, \Phi_i)$ *consists of a collaboration type node* col_i*, a number of role types* ro_i^j*, an UML class diagram* CD_i*, a function* $R_i : \{\mathsf{col}_i, ro_i^1, \ldots, ro_i^{n_i}\} \mapsto 2^{\mathcal{R}}$ *assigning rules to role types, and a guaranteed property* Φ_i*.*



SMARTSOS: Collaboration Types

Definition 2 (see [54]). *A collaboration type* $\mathsf{Col}_i = (\mathsf{col}_i, (ro_i^1, \ldots, ro_i^{n_i}), CD_i, R_i, \Phi_i)$ *consists of a collaboration type node* col_i*, a number of role types* ro_i^j*, an UML class diagram* CD_i*, a function* $R_i : \{\mathsf{col}_i, ro_i^1, \ldots, ro_i^{n_i}\} \mapsto 2^{\mathcal{R}}$ *assigning rules to role types, and a guaranteed property* Φ_i*.*

- The roles of the collaborations capture the permitted behavior:
  - Underspecification permits local decisions/self-adaptation. E.g.,
    - Non-determinism provide options for decisions
    - Time intervals allow to optimize timing via self-adaptation
- **Self-Organization** based on runtime models become possible:
  - Required properties must emerge from local rules
  - Context and runtime models can be employed as well (stigmergy, context-aware rules, …)

➔ We support **SoS-Level Self-Organization**, **SoS-Level Structural Dynamics**, and **Runtime Knowledge Exchange**

SMARTSOS: System Types

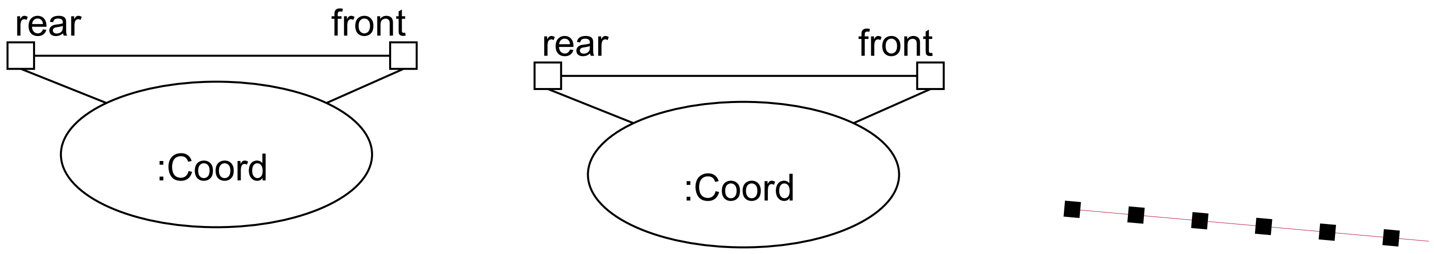
Definition 3 (see [54]). *A system type* $\mathsf{Sys}_i = (\mathsf{sys}_i, (ro_i^1, \ldots, ro_i^{m_i}), CD_i, R_i, I_i, \Psi_i)$ *consists of a system type node* sys_i*, a number of role types* ro_i^j*, a class diagram* CD_i*, a function* $R_i : \{\mathsf{sys}_i, ro_i^1, \ldots, ro_i^{m_i}\} \mapsto 2^{\mathcal{R}}$ *assigning rules to role types, a set of initial rules* $I_i \subseteq R_i(\mathsf{sys}_i)$*, and a safety property* Ψ_i*.*

- The system behavior has to respect the roles (of the collaborations):
  - All rules with side effects have to refine permitted behavior
  - All rules can access the elements visible via collaborations
- **Self-Adaptation** based on runtime models become possible:
  - Self: runtime model of the system itself
  - Local context: local context of the system
  - Shared context: runtime models of other systems

➔ We have enabled **Self-Adaptation** for the systems

SMARTSOS: Correct Collaborations

Definition 8 (see [54]). *A collaboration type* $\mathsf{Col}_i = (\mathsf{col}_i, (ro_i^1, \ldots, ro_i^{n_i}), CD_i, R_i, \Phi_i)$ *is* correct *if for all initial configurations* $G_I \in \mathcal{G}_\emptyset(CD_i)$ *holds that for* $R_i(\mathsf{Col}_i) = R_i(ro_i^1) \cup \cdots \cup R_i(ro_i^n) \cup R_i(\mathsf{col}_i))$ *the overall behavior of the collaboration the reachable collaboration configurations are correct:* $G_I, R_i(\mathsf{Col}_i) \models \Phi_i$.

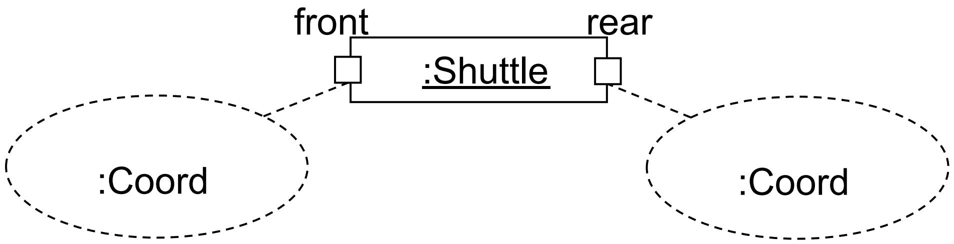


SMARTSOS: Correct Systems

Definition 9 (see [54]). *A system type* $\mathsf{Sys}_i = (\mathsf{sys}_i, (ro_i^1, \ldots, ro_i^{m_i}), CD_i, I_i, \Psi_i)$ *is correct if for all initial configurations* $G_I \in \mathcal{G}_\emptyset(CD_i)$ *holds that the reachable configurations are correct* $G_I, R_i(\mathsf{sys}_i) \cup COMP(\mathsf{Sys}_i) \cup I_i \models \Psi_i \quad (1)$ *and the system behavior* $R_i(\mathsf{sys}_i)$ *refines the orthogonally combined role behavior and creation behavior*

$$R_i(\mathsf{sys}_i) \sqsubseteq R_i(ro_i^1) \cup \cdots \cup R_i(ro_i^{m_i}) \quad \cup \quad I_i. \qquad (2)$$

To add the collaboration behavior to the system behavior for each role without the role itself, we employ here $COMP(\mathsf{Sys}_i) = \bigcup_{1 \leq l \leq m_i} COMP(\mathsf{Sys}_i, ro_i^l)$ *with* $COMP(\mathsf{Sys}_i, ro_i^l) = R_j(\mathsf{Col}_j)$ *which is covered by* $R_i(\mathsf{sys}_i)$ *to derive a related closed behavior.*

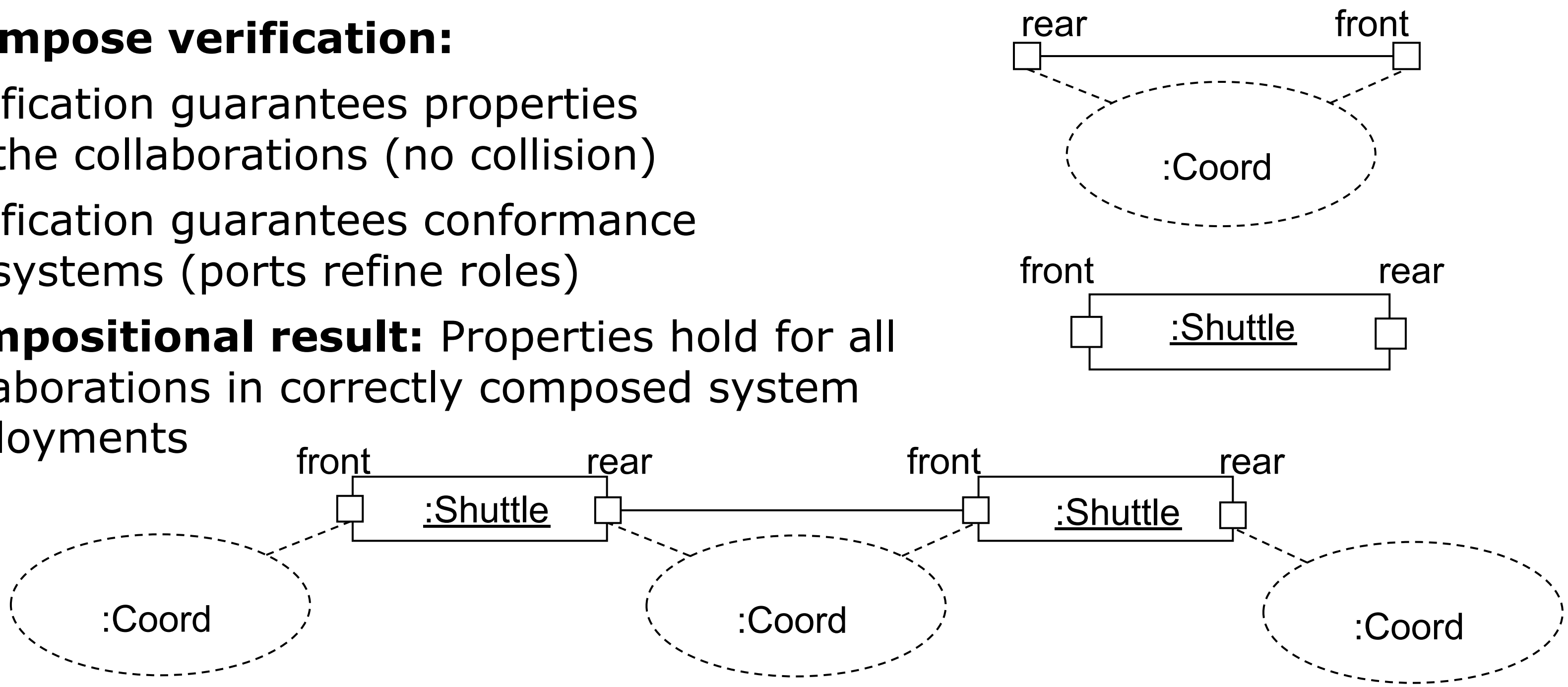


SMARTSOS: Scalable Correctness SoS

Theorem 1 ([54]). *A system of systems* $\mathsf{sos} = (\mathsf{SoS}, G_\emptyset)$ *with system of system type* $\mathsf{SoS} = ((\mathsf{Col}_1, \ldots, \mathsf{Col}_n), (\mathsf{Sys}_1, \ldots, \mathsf{Sys}_m))$ *is* correct *if (1) the system of system type* SoS *is* type conform, *(2) all collaboration types* $\mathsf{Col}_1, \ldots, \mathsf{Col}_n$ *are* correct, *and (3) all system types* $\mathsf{Sys}_1, \ldots, \mathsf{Sys}_m$ *are* correct.

Decompose verification:

- Verification guarantees properties for the collaborations (no collision)
- Verification guarantees conformance for systems (ports refine roles)
- **Compositional result:** Properties hold for all collaborations in correctly composed system deployments



➔ We have a first element for the **Resilience** of the SoS

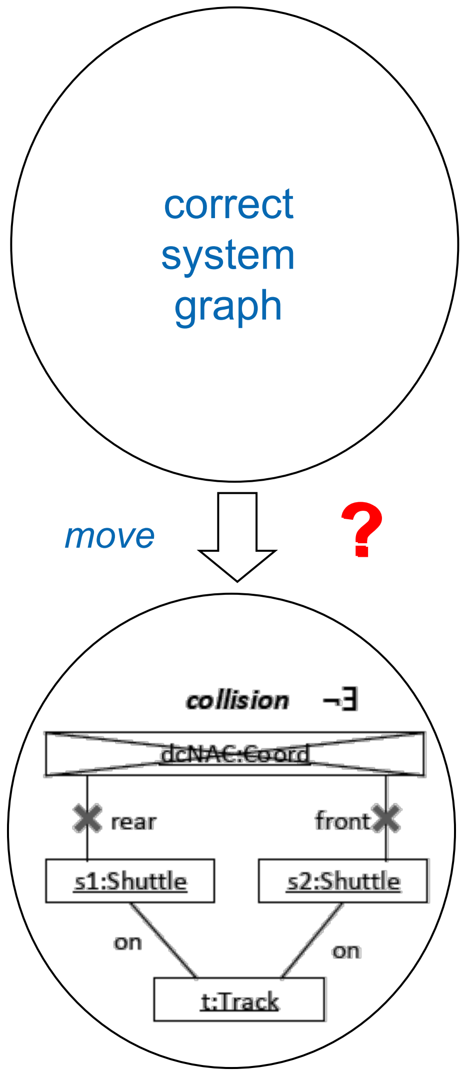
SMARTSOS: Correctness of a Collaboration

Verification Problem:

- Infinite many initial states or reachable state are possible
- State and sequence properties would be of interest

Checking Options:

- Model Checking (mapping to GROOVE; only debugging)
  - Limited to small configurations and finite models
  - Extension for continuous time have been developed
- Invariant Checker for state properties (our development)
  - Analyze that changes can not lead from safe to unsafe situations (**inductive invariants**)
  - Supports infinite many start configurations specified only by their structural properties
  - Supports infinite state models
  - Extension of time and discrete variables exist
  - Incremental check for changed rules
  - Extension of hybrid behavior



[ICSE2006, ISORC2008]

SMARTSOS: Correctness of a Collaboration

Observation: any possible counter-example must contain an intersection between the nodes of the RHS of the rule and the forbidden graph F. Therefore, if (P,r) is a **counterexample**, then:

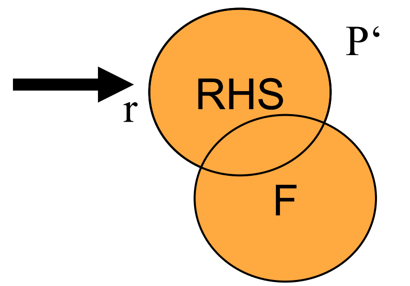
(1) exists a P’ which is the combination of a RHS of a rule r and a forbidden graph pattern F,

(2) it holds $P \rightarrow_r P'$ (which implies that no rule r’ with higher priority can be applied), and

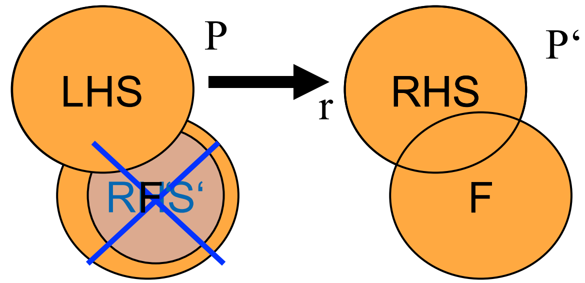
(3) there exists no forbidden graph F’ which matches P (as then the graph before was not correct already)

Idea: Algorithm constructs all possible counter-examples and checks whether any could be a real one.

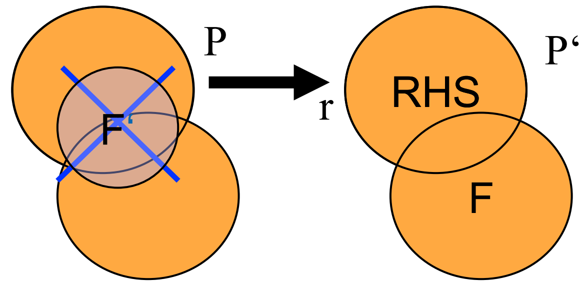
(1)



(2)



(3)



SMARTSOS: Correctness of a Collaboration

	Time	#Rules	#Prop.
Correctness of the Coord Collaboration	340 ms	4	2
Role refinement by the shuttle systems	47 ms		

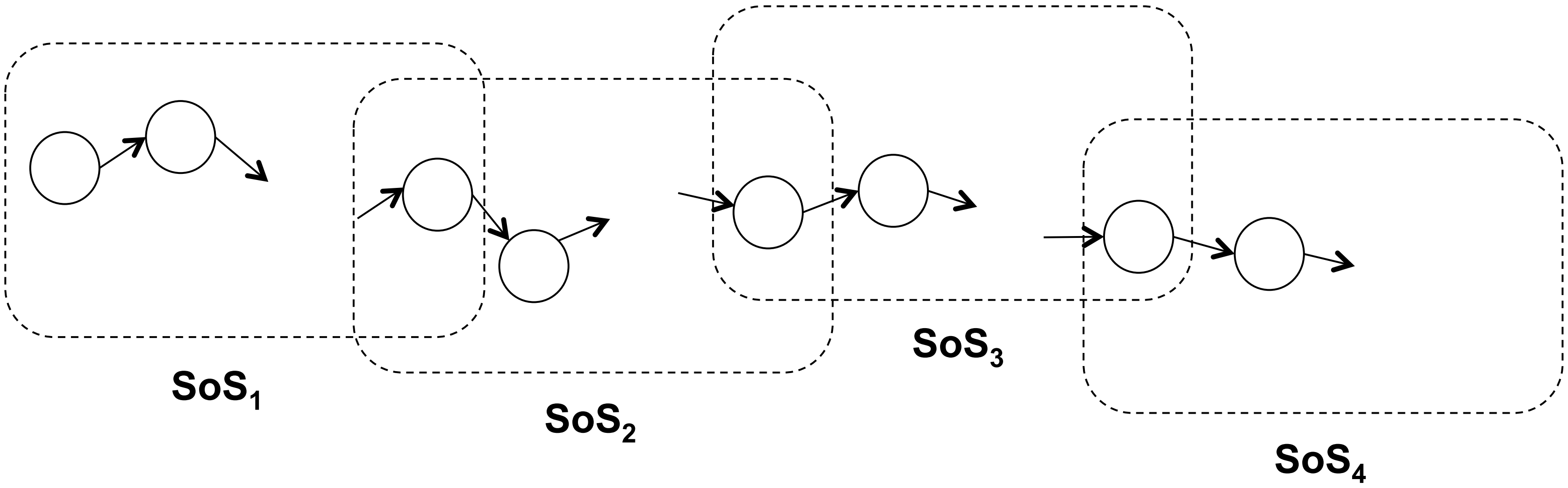
- The checking is quite fast as it does not consider the state space
- The complexity mainly depends on the complexity of the rules and properties (resp. the possible overlaps)
- Time leads to a low number of constraints (here 22) that only require a negligible fraction of the effort

BUT: only state properties!

SMARTSOS: Evolution & Correctness

35

Definition 13 (see [54]). *An extended evolution sequence* $(\mathsf{SoS}_1, G_S^1), \ldots, (\mathsf{SoS}_n, G_S^n)$ *with* $\mathsf{SoS}_n = ((\mathsf{Col}_1, \ldots, \mathsf{Col}_p), (\mathsf{Sys}_1, \ldots, \mathsf{Sys}_q))$ *is correct if for any combined path* $\pi_1 \circ \cdots \circ \pi_n$ *such that* π_i *is a path in* SoS_i *leading from* G_S^i *to* G_S^{i+1} *for* $i < n$ *and that* π_n *is a path in* SoS_n *starting from* G_S^n *holds:* $\pi_1 \circ \cdots \circ \pi_n \models \Phi_1 \wedge \cdots \wedge \Phi_p \wedge \Psi_1 \wedge \cdots \wedge \Psi_q$. *An evolution sequence* $\mathsf{SoS}_1, \ldots, \mathsf{SoS}_n$ *is correct if all possible related extended evolution sequence* $(\mathsf{SoS}_1, G_S^1), \ldots, (\mathsf{SoS}_n, G_S^n)$ *are correct.*



SMARTSOS: Evolution & Correctness

Theorem 2 (see [54]). *An evolution sequence of systems* $\mathsf{SoS}_1, \ldots, \mathsf{SoS}_n$ *is* correct *if the related dynamic evolving system of system type* $E(\mathsf{SoS}_1, \mathsf{SoS}_n)$ *is correct.*

Lemma 2 (see [54]). *For a correct collaboration type* Col *holds also that its dynamic extension* $E(Col)$ *is correct. For a correct system type* Sys *holds also that its dynamic extension* $E(\mathsf{Sys})$ *is correct.*

Due to Lemma 2, it is sufficient to simply check the collaboration and system types and this already guarantees that any extended evolution sequence will also show correct behavior.

Consequences:

- → Verification support for Evolution allows to operate with only **Restricted Knowledge** as available in case of independent operation and management
- → Verification enables independent **Evolution** to some extent

Outline

1. Challenges Ahead
2. Available Options
3. SMARTSOS
4. Conclusions & Outlook

Conclusions

Graph transformation systems encoding models and their linking allow to combine Service-Oriented Architecture, Self-Adaptive / Self-Organization, and Runtime Models with evolving structures and are a suitable basis for a **solid foundation** for **Smart SoS ...**

- Collaborations support **SoS-Level Self-Organization**, **SoS-Level Structural Dynamics**, and **Runtime Knowledge Exchange**
- Runtime models and via collaborations shared runtime models enabled **Self-Adaptation** of the systems
- Compositional Verification is a first element for the **Resilience** of the SoS
- Verification support for Evolution allows to operate with only **Restricted Knowledge** as available in case of independent operation and management and enables independent **Evolution** to some extent

Outlook

- The suggested model is a rather strong **idealization**:
  - If wrong likely also related less idealized design will fail as well
  - More accurate models can be used (verification gets harder)
    - the systems may copy (with some measurement effects) their context to capture delays etc.
    - the systems may hand over copies of their context to other systems such that the visible shared context is exchanged
- The formal model requires a strong **separation** into collaborations
- The formal model does only support time, but **extensions** exists
  - Hybrid, probabilistic, more complex attributes types and updates,
- The formal model requires maybe too much expertise, but ideas to better support the average modeler are under development

Bibliography

[Brooks+2008] Christopher Brooks, Chihhong Cheng, Thomas Huining Feng, Edward A. Lee and Reinhard von Hanxleden. Model Engineering using Multimodeling. In 1st International Workshop on Model Co-Evolution and Consistency Management (MCCM '08), September 2008.

[Broy+2012] Manfred Broy, MaríaVictoria Cengarle and Eva Geisberger. Cyber-Physical Systems: Imminent Challenges. In Radu Calinescu and David Garlan editors, Large-Scale Complex IT Systems. Development, Operation and Management, Vol. 7539:1-28 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012.

[Burmester+2008] Sven Burmester, Holger Giese, Eckehard Münch, Oliver Oberschelp, Florian Klein and Peter Scheideler. Tool Support for the Design of Self-Optimizing Mechatronic Multi-Agent Systems. In International Journal on Software Tools for Technology Transfer (STTT), Vol. 10(3):207-222, Springer Verlag, June 2008.

[Giese+2011] Holger Giese, Stefan Henkler and Martin Hirsch. A multi-paradigm approach supporting the modular execution of reconfigurable hybrid systems. In Transactions of the Society for Modeling and Simulation International, SIMULATION, Vol. 87(9):775-808, 2011.

[Giese&Schäfer2013] Holger Giese and Wilhelm Schäfer. Model-Driven Development of Safe Self-Optimizing Mechatronic Systems with MechatronicUML. In Javier Camara, Rogério de Lemos, Carlo Ghezzi and AntÃ³nia Lopes editors, Assurances for Self-Adaptive Systems, Vol. 7740:152-186 of Lecture Notes in Computer Science (LNCS), Springer, January 2013.

[Maier1998] Mark W. Maier. Architecting principles for systems-of-systems. In Systems Engineering, Vol. 1(4):267--284, John Wiley & Sons, Inc., 1998.

[Northrop+2006] Northrop, Linda, et al. Ultra-Large-Scale Systems: The Software Challenge of the Future. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

Bibliography

[Pereira+2013] Eloi Pereira, Christoph M. Kirsch, Raja Sengupta and Jo~ao Borges de Sousa. Bigactors - A Model for Structure-aware Computation. In ACM/IEEE 4th International Conference on Cyber-Physical Systems, Pages 199--208, ACM/IEEE, Philadelphia, PA, USA, 2013.

[Sztipanovits2011] Janos Sztipanovits with Ted Bapty, Gabor Karsai and Sandeep Neema. MODEL-INTEGRATION AND CYBER PHYSICAL SYSTEMS: A SEMANTICS PERSPECTIVE. FM 2011, Limerick, Ireland. 22 June 2011

[Sztipanovits+2012] Janos Sztipanovits, Xenofon Koutsoukos, Gabor Karsai, Nicholas Kottenstette, Panos Antsaklis, Vineet Gupta, B. Goodwine, J. Baras and Shige Wang. Toward a Science of Cyber-Physical System Integration. In Proceedings of the IEEE, Vol. 100(1):29-44, January 2012.

[Valerdi+2008] Ricardo Valerdi, Elliot Axelband, Thomas Baehren, Barry Boehm, Dave Dorenbos, Scott Jackson, Azad Madni, Gerald Nadler, Paul Robitaille and Stan Settles. A research agenda for systems of systems architecting. In International Journal of System of Systems Engineering, Vol. 1(1-2):171--188, 2008.

[Vogel+2009] Thomas Vogel, Stefan Neumann, Stephan Hildebrandt, Holger Giese and Basil Becker: Model-Driven Architectural Monitoring and Adaptation for Autonomic Systems. In: Proc. of the 6th International Conference on Autonomic Computing and Communications (ICAC'09), Barcelona, Spain, ACM (15-19 June 2009)

[Vogel+2010] Thomas Vogel and Stefan Neumann and Stephan Hildebrandt and Holger Giese and Basil Becker. Incremental Model Synchronization for Efficient Run-Time Monitoring. In Sudipto Ghosh, ed., Models in Software Engineering, Workshops and Symposia at MODELS 2009, Denver, CO, USA, October 4-9, 2009, Reports and Revised Selected Papers, vol. 6002 of Lecture Notes in Computer Science (LNCS), pages 124-139. Springer-Verlag, 4 2010.

[Vogel&Giese2012] Thomas Vogel and Holger Giese. A Language for Feedback Loops in Self-Adaptive Systems: Executable Runtime Megamodels. In Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012), pages 129-138, 6 2012. IEEE Computer Society.