Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

chist-era

FET

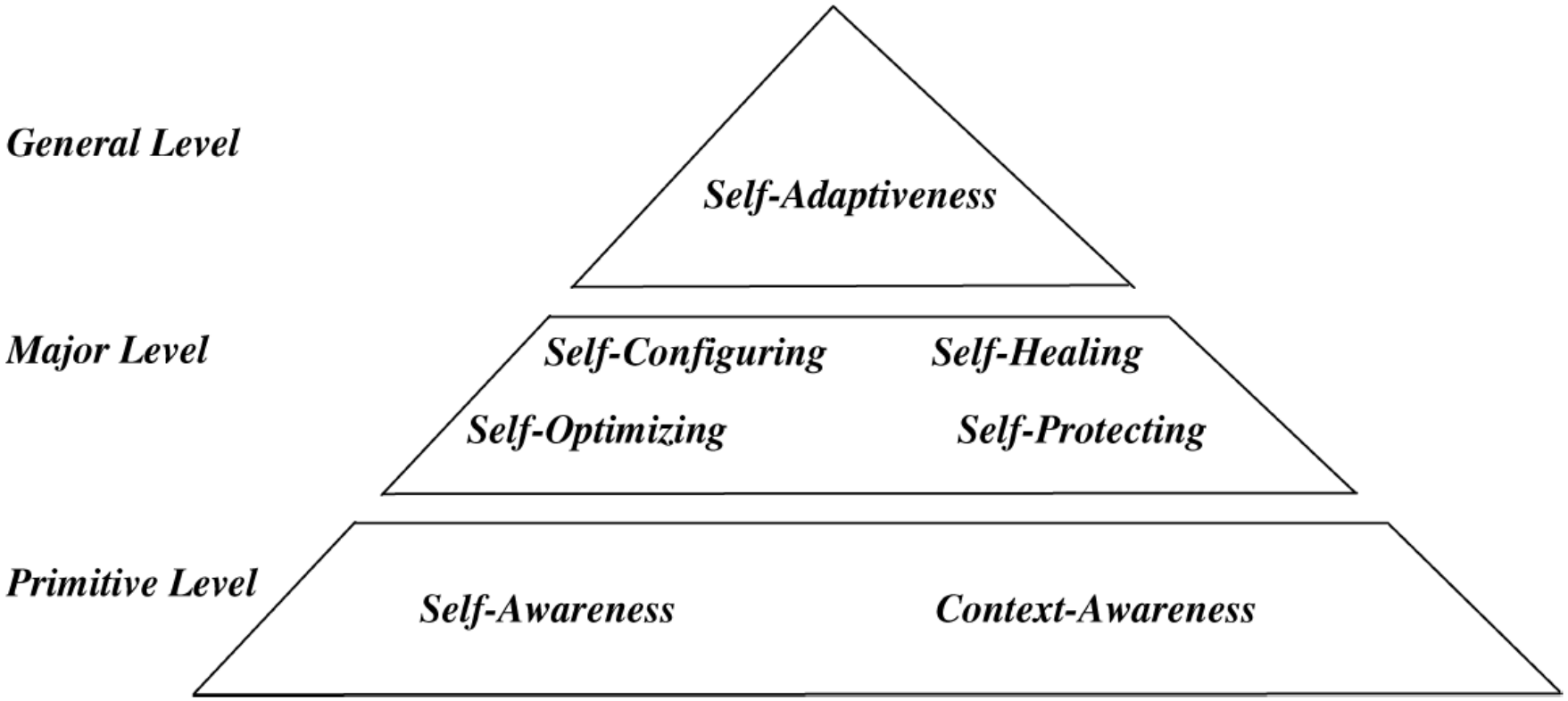
Self-Awareness and Self-Consciousness via Software: An Engineering Perspective

Expert Talk, First CHIST-ERA conference
EUR - Rome, 28 May 2010

Holger Giese
System Analysis & Modeling Group, Hasso Plattner Institute for Software Systems Engineering at the University of Potsdam, Germany

holger.giese@hpi.uni-potsdam.de

My Terminology



Mazeiar Salehie and Ladan Tahvildari. *Self-adaptive software: Landscape and research challenges*. In ACM Trans. Auton. Adapt. Syst., Vol. 4(2):1--42, ACM, New York, NY, USA , 2009.

Problem Statement

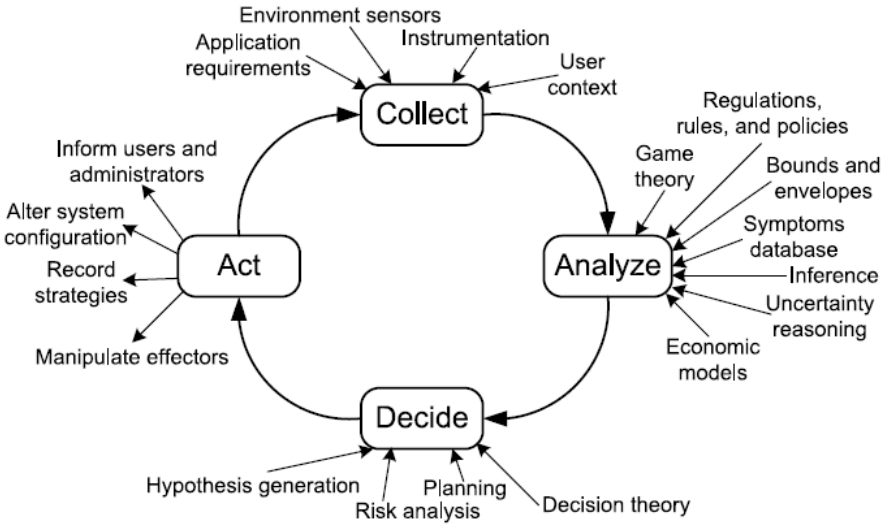
Self-awareness and **self-consciousness** via software will lead to a new generation of adaptive and evolving systems of systems.

Disclaimer: *Self* and *consciousness* are considered only in a weak partial sense.

BUT we do know yet

- How to **cost effectively** engineer this capabilities?
- How to engineer **meaningful** and **trustworthy** systems of systems where the subsystem have this capability?

Adaptation Loop & Self-Consciousness



Roadmap:

- Requirements
- Modeling
- Feedback loops
- Assurance

Self-Awareness:

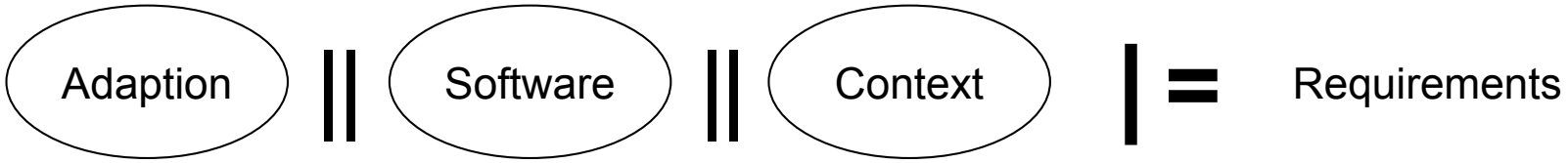
- Includes collect
- Enables analyze
- Basis for act

Self-Consciousness:

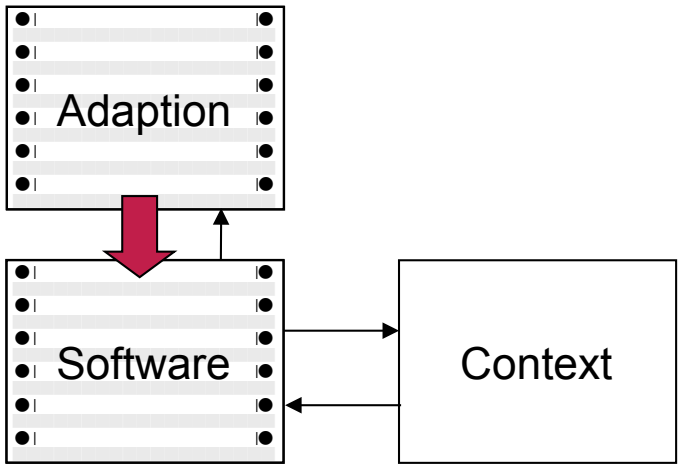
- Includes collect
- Enables analyze
- Basis for decide (alternatives as well as comparison)
- Basis for act

Engineer Monolithic Self-Awareness

Development Time:



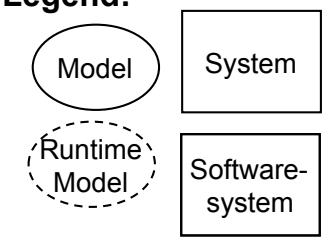
Runtime:



Benefit:

Adaption based on self-awareness (and context-awareness) enables to fulfill the requirements despite dynamic changes of the context (reflected in the software) and software.

Legend:

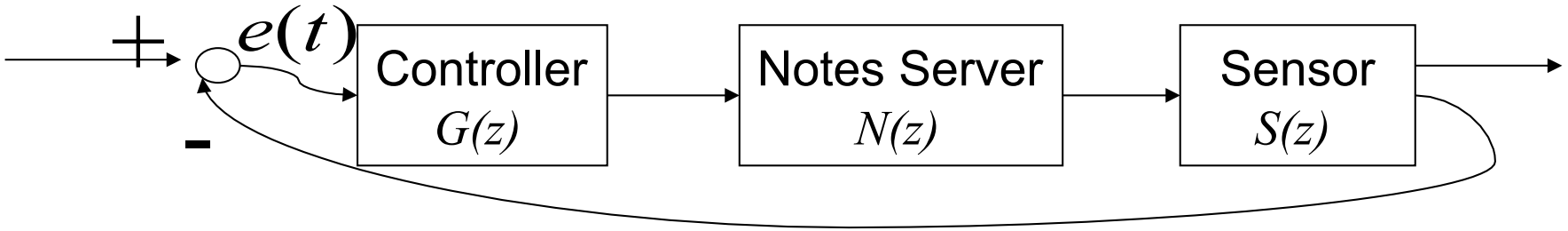


Challenge: engineer adaptation loop

Assumption: known context and software where only parameters change

Autonomic Computing:
Software & Control Theory

$H(z)$ = Closed Loop Transfer Function



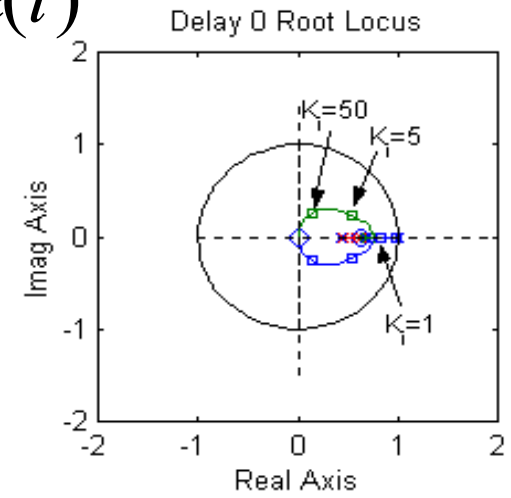
Integral Control Law

$$u(t) = u(t-1) + Ke(t)$$

Transfer Function

$$N(z) = \frac{b_1}{z - a_1}$$

Poles of $H(z)$



Choose a good K

Joseph L. Hellerstein, Yixin Diao, Sujay Parekh and Dawn M. Tilbury. Feedback Control of Computing Systems. John Wiley \& Sons, 2004.

Beyond Parameter Changes …

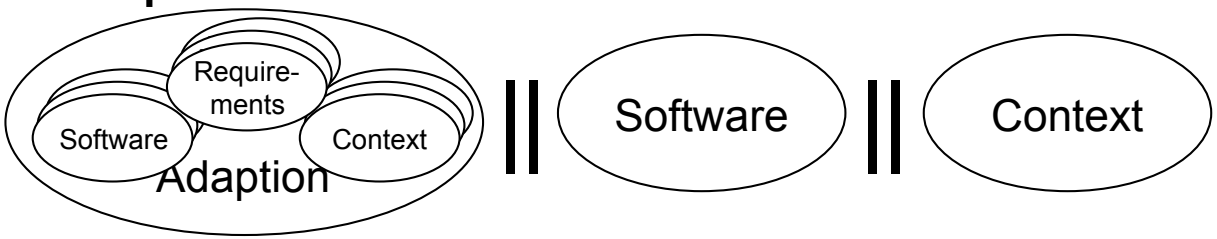
Example: "Ambient Intelligence (AmI): …

By adding intelligent user interfaces and integrating sensing devices, it should be possible to **identify and model** user activities, preferences and behaviours, and create individualised profiles."

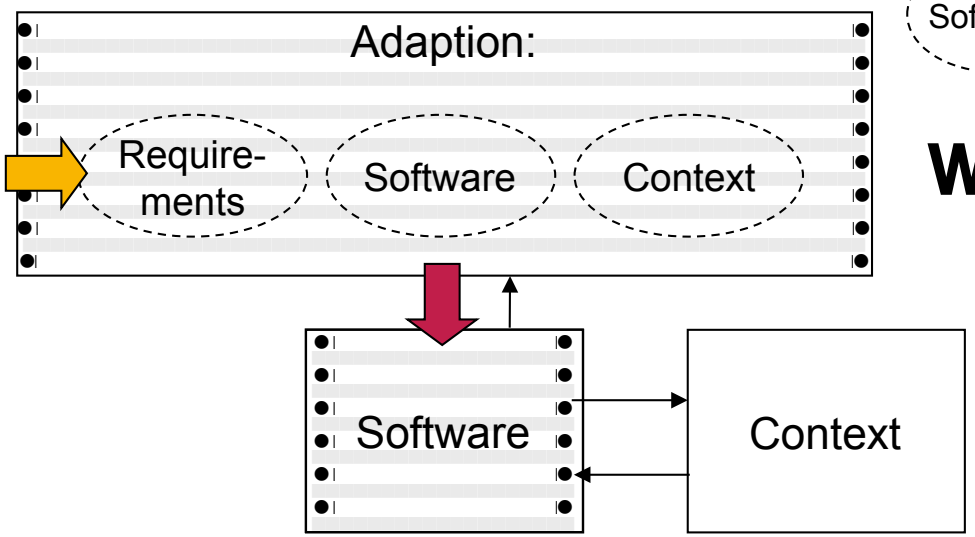
Gasson, Mark; Warwick, Kevin (2007), "D12.2: Study on Emerging AmI Technologies",FIDIS Deliverables 12 (2)

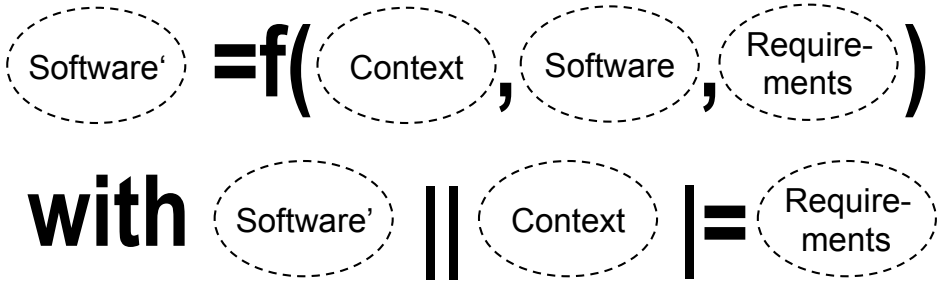
Engineer Monolithic Self-Consciousness

Development Time:



Runtime:





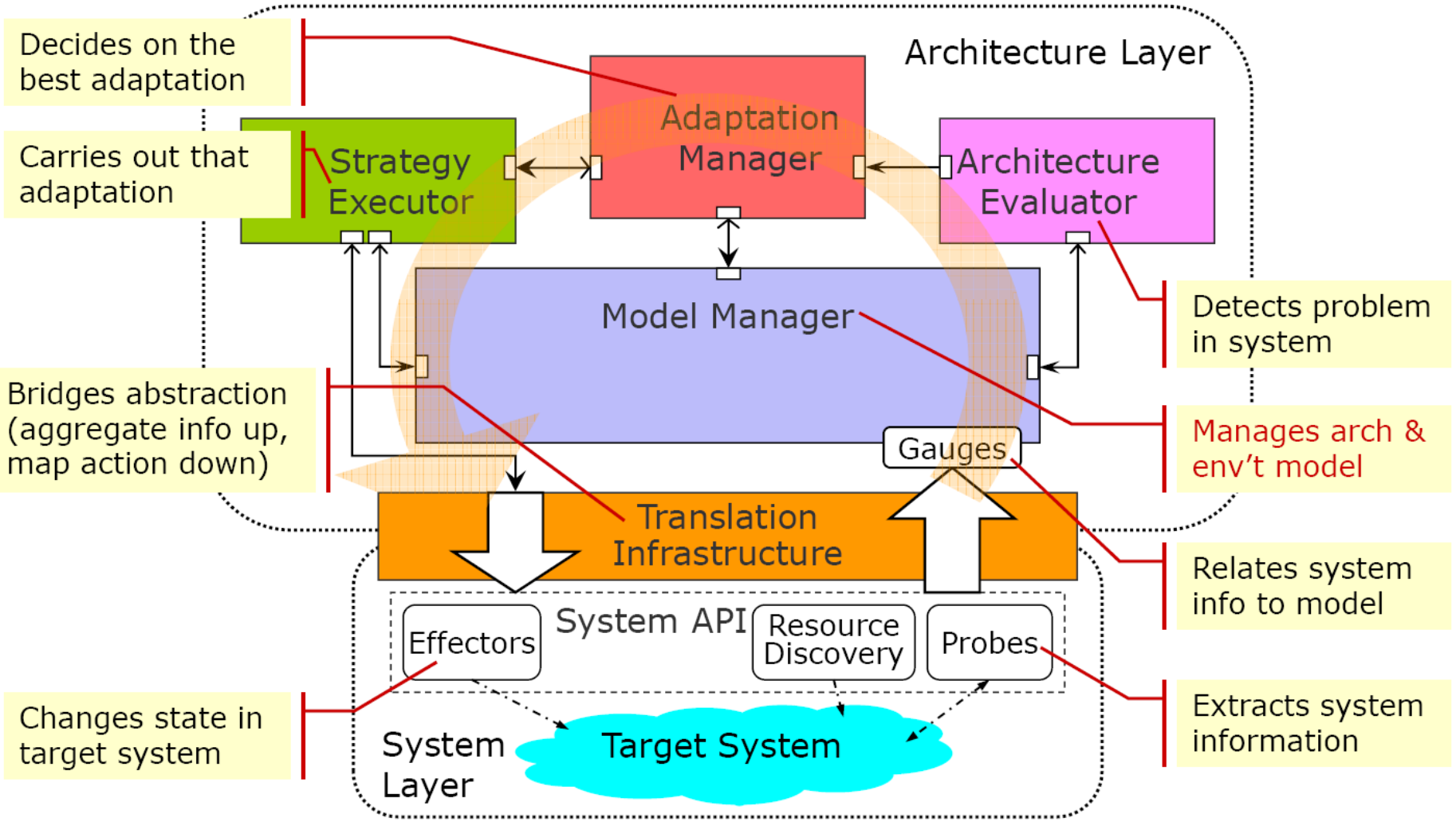
Challenges: engineer + find reasonable f + assurance

Assumption: stable context and requirements

Benefits:

- Covers more possible changes of the software and context
- Possible changes of the software and context can be analyzed at runtime (by f)
- Changes of the software could also be triggered by changes of the requirements

Framework: Reuse Adaptation Infrastructure

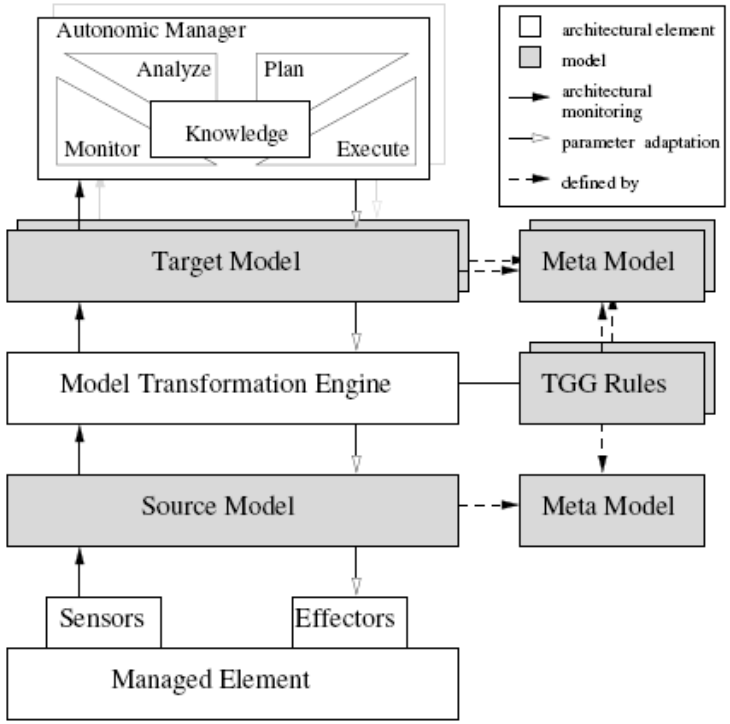


Rainbow

David Garlan, Bradley Schmerl, Shang-Wen Cheng. Architecture-based Self-adaptation. IBM 3rd Proactive Problem Prediction Conference. April 27, 2005

Requirements and Models at Runtime

- Requirements are monitored at runtime
- Software is represented by a runtime model (synchronization)
- Generate system specific parts for collect, analyze, decide and act



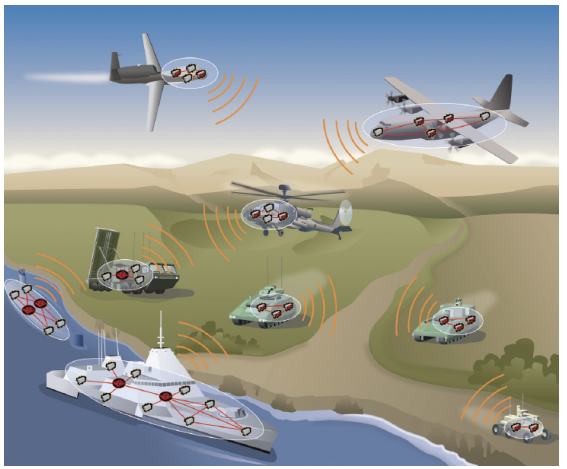
Vogel, T., Neumann, S., Hildebrandt, S., Giese, H., Becker, B.: Model-Driven Architectural Monitoring and Adaptation for Autonomic Systems. In: Proc. of the 6th International Conference on Autonomic Computing and Communications (ICAC'09), Barcelona, Spain, ACM (15-19 June 2009) accepted paper.

Beyond Monolithic Systems ...

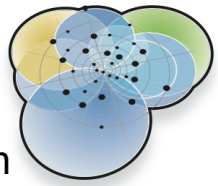
Prognoses:

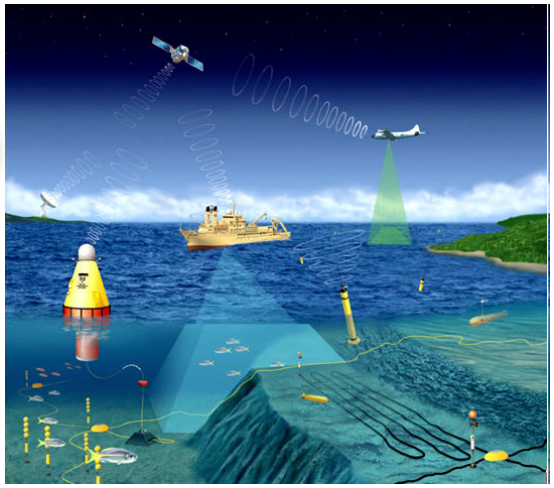
"In the near future, **software-intensive systems** will exhibit **adaptive** and **anticipatory behavior**; they will process knowledge and not only data, and **change their structure dynamically**. Software-intensive systems will act as global computers **in highly dynamic environments** and will be based on and **integrated** with service-oriented and pervasive computing."

M. Wirsing, ed., Report on the EU/NSF Strategic Workshop on Engineering Software-Intensive Systems *"Challenges, Visions and Research Issues for Software-Intensive Systems„ at ICSE 2004.* Edinburgh, UK, May 2004.



"The sheer scale of **ultra large scale 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.

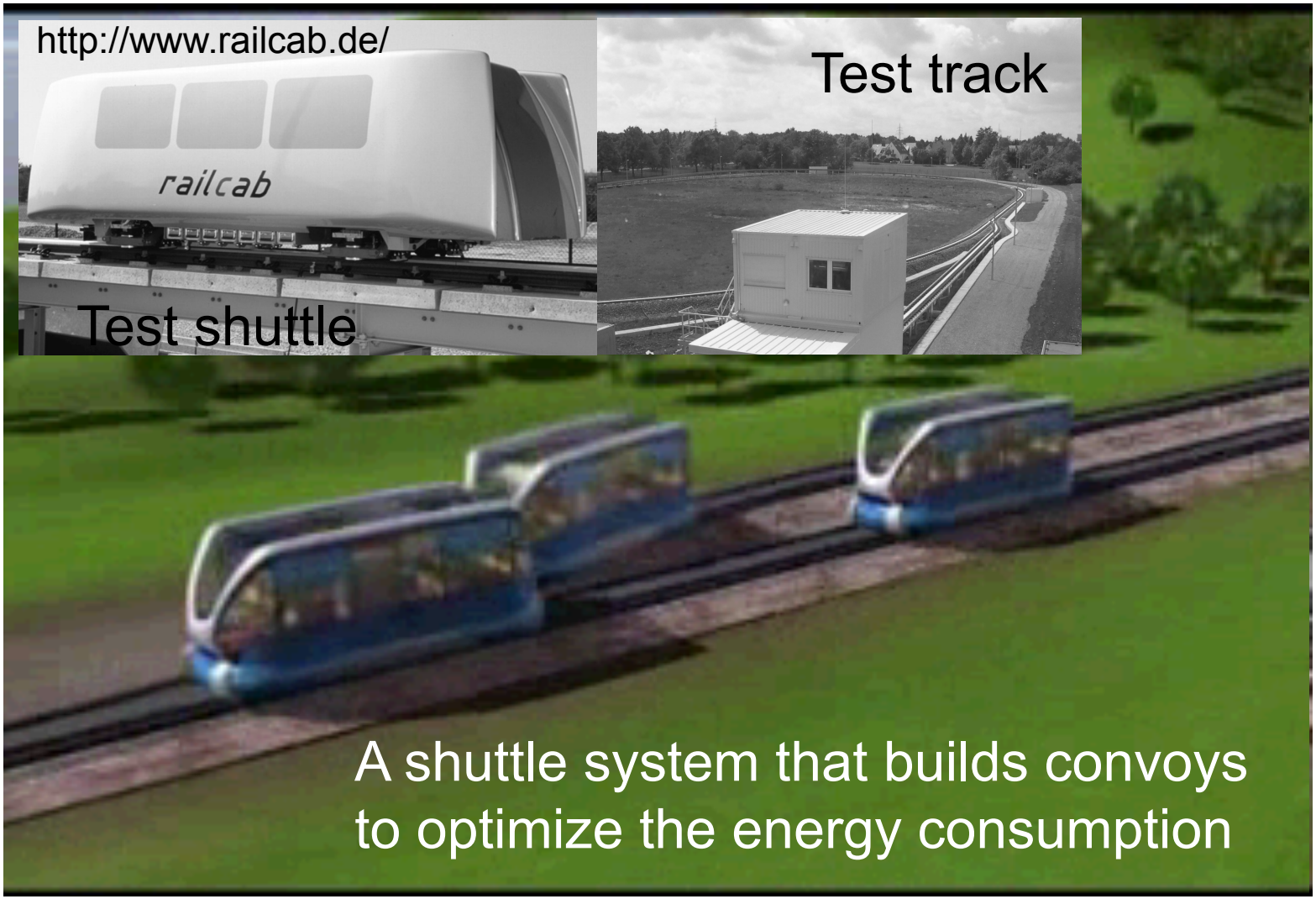




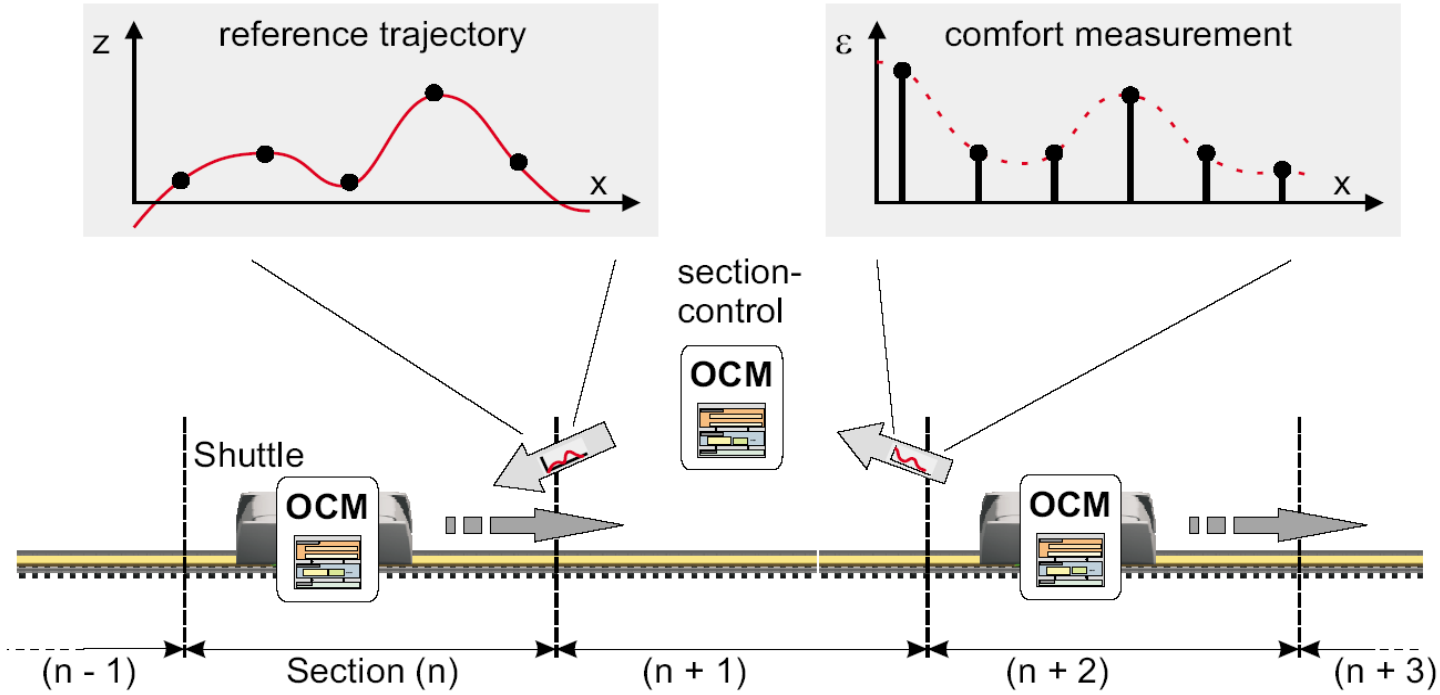
Adaptation is needed to compensate for changes in the mission requirements (…) and operating environments (..)

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

Example: Shuttle System



Example for Distributed Self-Consciousness/Adaptation

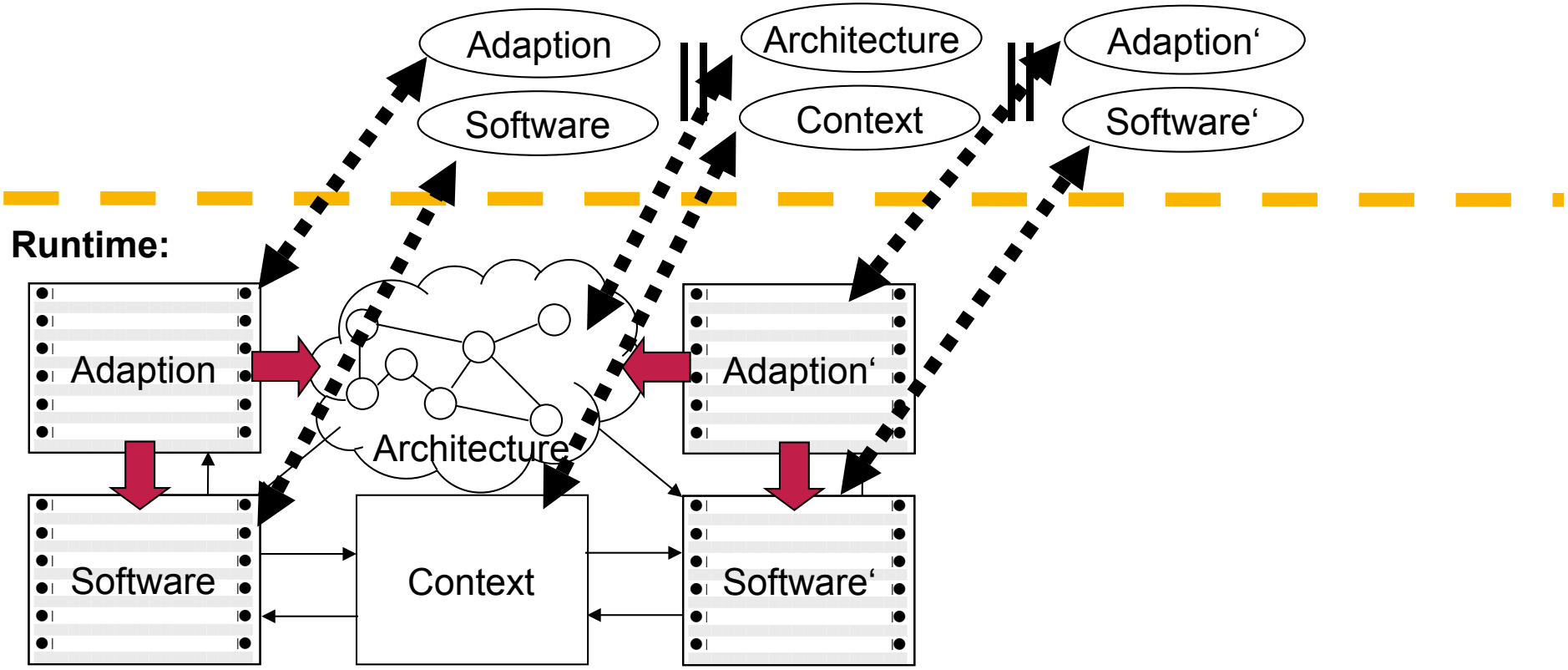


- Distributed learning of a model of the track (context)
- Local learning of a model of the shuttle (self!)
- Planning an adaptation in form of an optimal trajectory
- Trajectory synthesis establishes required assurance

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

Engineer Distributed Self-Consciousness

Development time:



Open Problems:

(1) Multiple possibly conflicting adaptation loops

(2) Decentralized organization required => self-organization

Proposed Priorities for the Call

- **Cost-Effective Engineering of Systems with Self-Awareness & Self-Consciousness**
  - Forward Engineering via frameworks and models at runtime
  - Reengineering with black-box and white-box models
  - Co-existence of different levels (online application management ➔ self-awareness ➔ self-consciousness)
  - Assurance
- **Engineering of Systems of Systems with Self-Awareness & Self-Consciousness**
  - Decentralized self-consciousness & adaptation
  - Overcoming the heterogeneity
  - Decentralized assurance