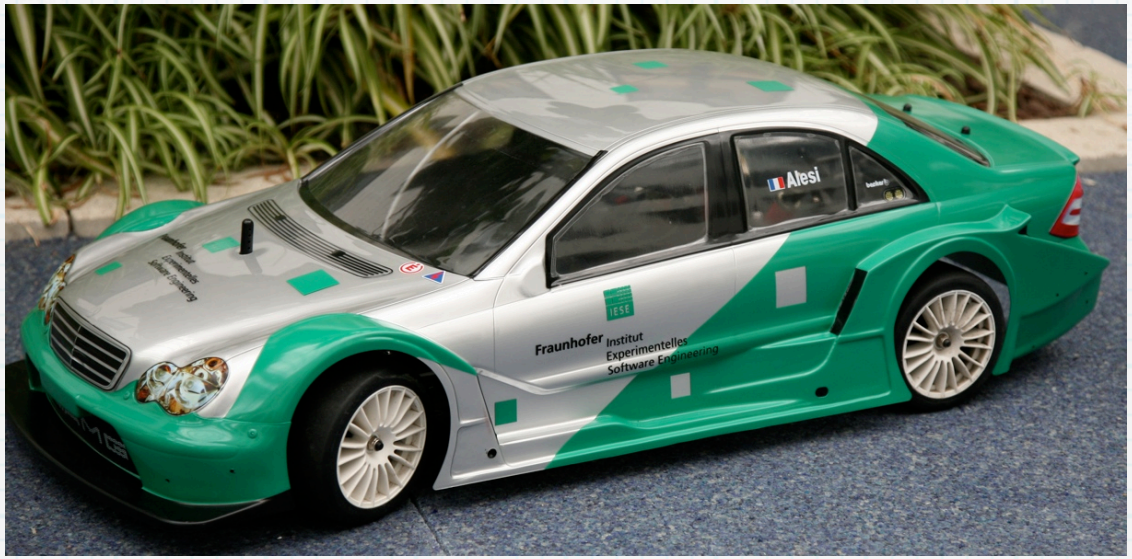
From Model-Based Design to Formal Verification of Adaptive Embedded Systems

Rasmus Adler, Ina Schaefer, Tobias Schüle, Eric Vecchié
Fraunhofer IESE and TU Kaiserslautern
Germany

9th Intl. Conference on Formal Engineering Methods (ICFEM 2007)
14 November 2007, Boca Raton, FL

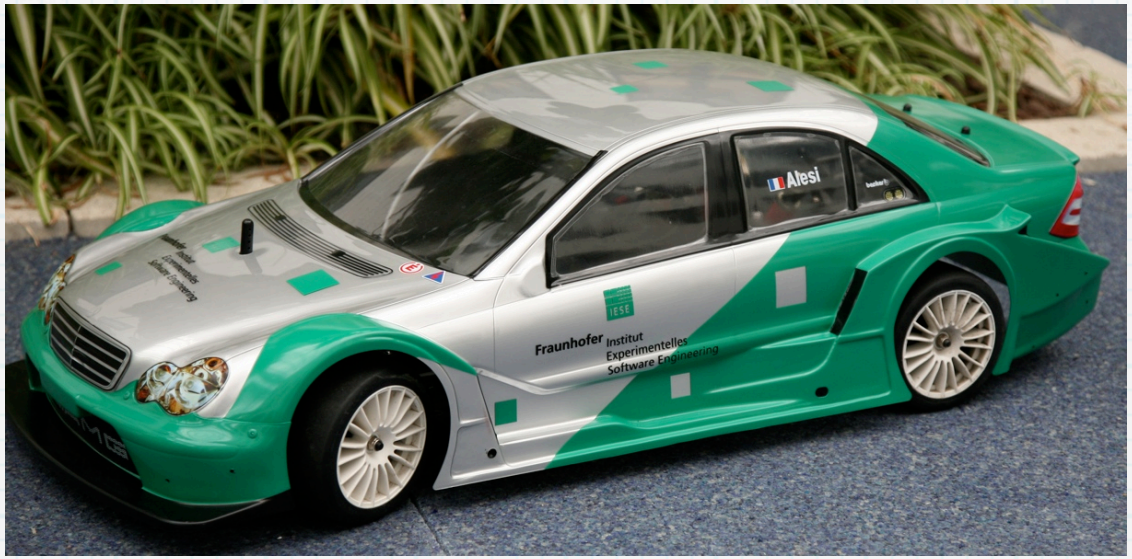
Technische Universität Kaiserslautern
SoFTeX Arbeitsgruppe Softwaretechnik
DASMOD Dependable Adaptive Systems and Mathematical Modeling

Self-Adaptive Systems

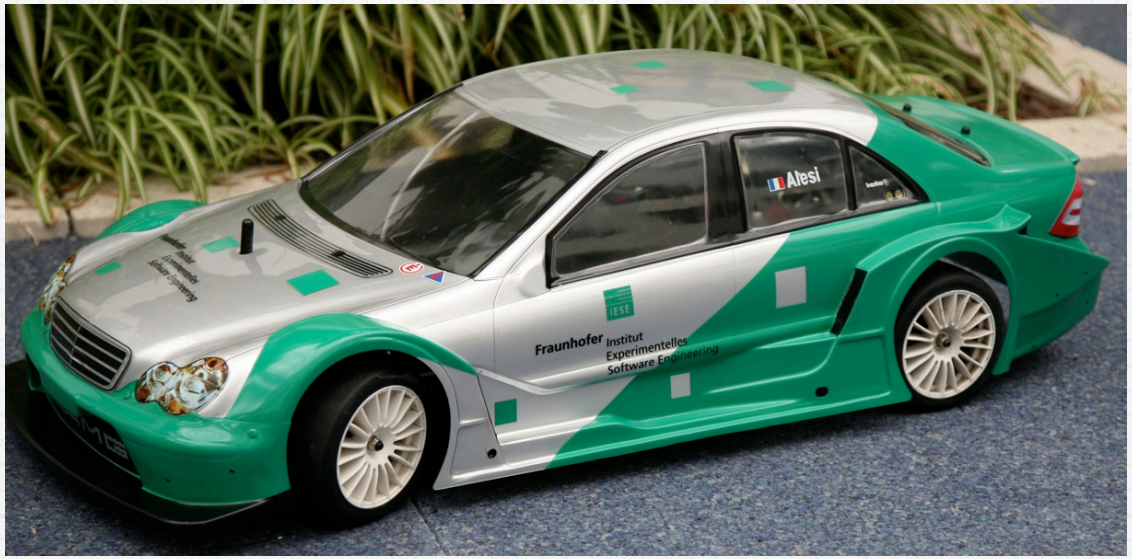


Self-Adaptive Systems

HW - Fault
SW - Fault



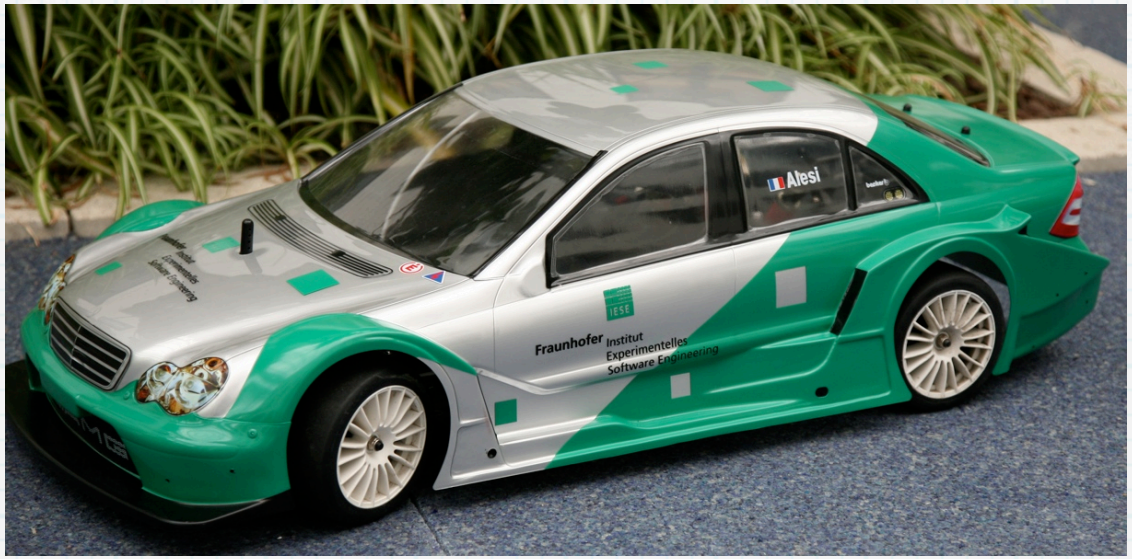
Self-Adaptive Systems



HW - Fault
SW - Fault

Runtime adaptation

Self-Adaptive Systems

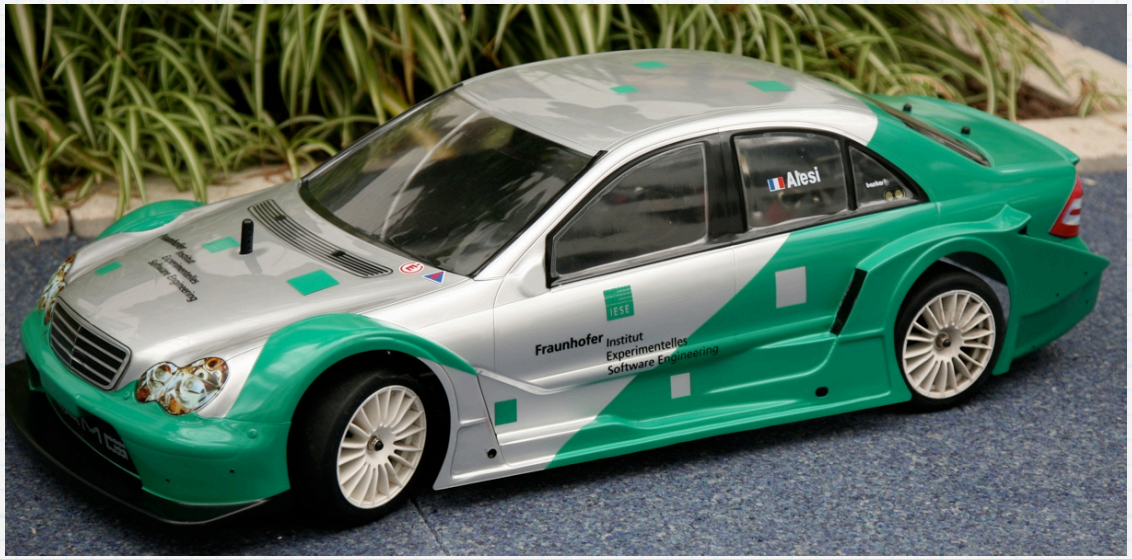


HW - Fault
SW - Fault

Runtime adaptation

Degrade functionality to remain operational by switching to different predetermined configuration

Self-Adaptive Systems



HW - Fault
SW - Fault

Safety
Survivability

Runtime adaptation

Degrade functionality to remain operational by switching to different predetermined configuration

Adaptive Systems Development

Adaptation increases design complexity, since

- **decentralized adaptation mechanism**
- **reconfiguration triggers reconfiguration**
- **complex interdependencies**

Adaptive Systems Development

Adaptation increases design complexity, since

- **decentralized adaptation mechanism**
- **reconfiguration triggers reconfiguration**
- **complex interdependencies**

Solution:

- **Model-Based Design of Adaptive Systems**
- **Integrated with Formal Verification of Adaptation Behaviour**

Integration - Outline

Integration - Outline

Modelling Concepts for Adaptive Systems

Integration - Outline

Modelling Concepts for Adaptive Systems

Verification Tools

System Properties

Integration - Outline

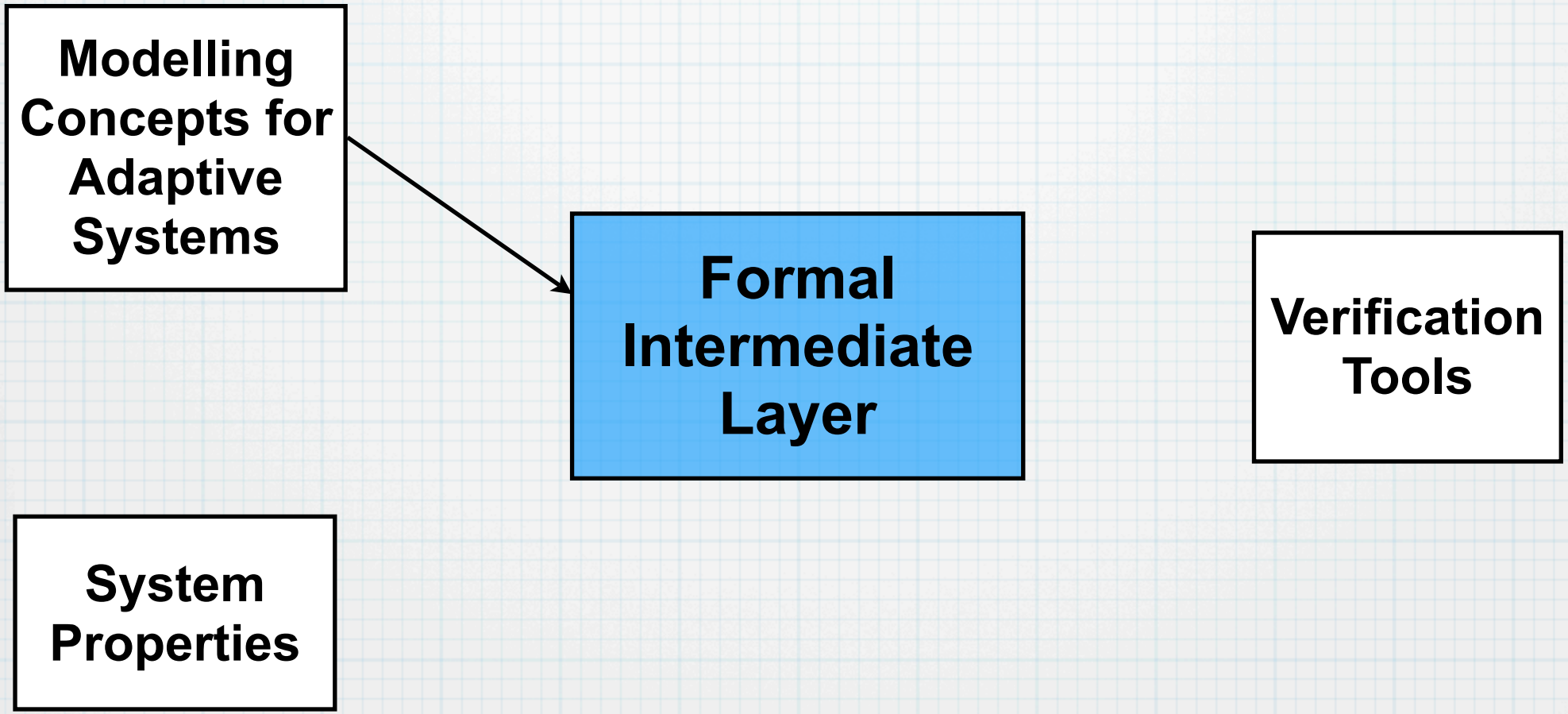
Modelling Concepts for Adaptive Systems

Formal Intermediate Layer

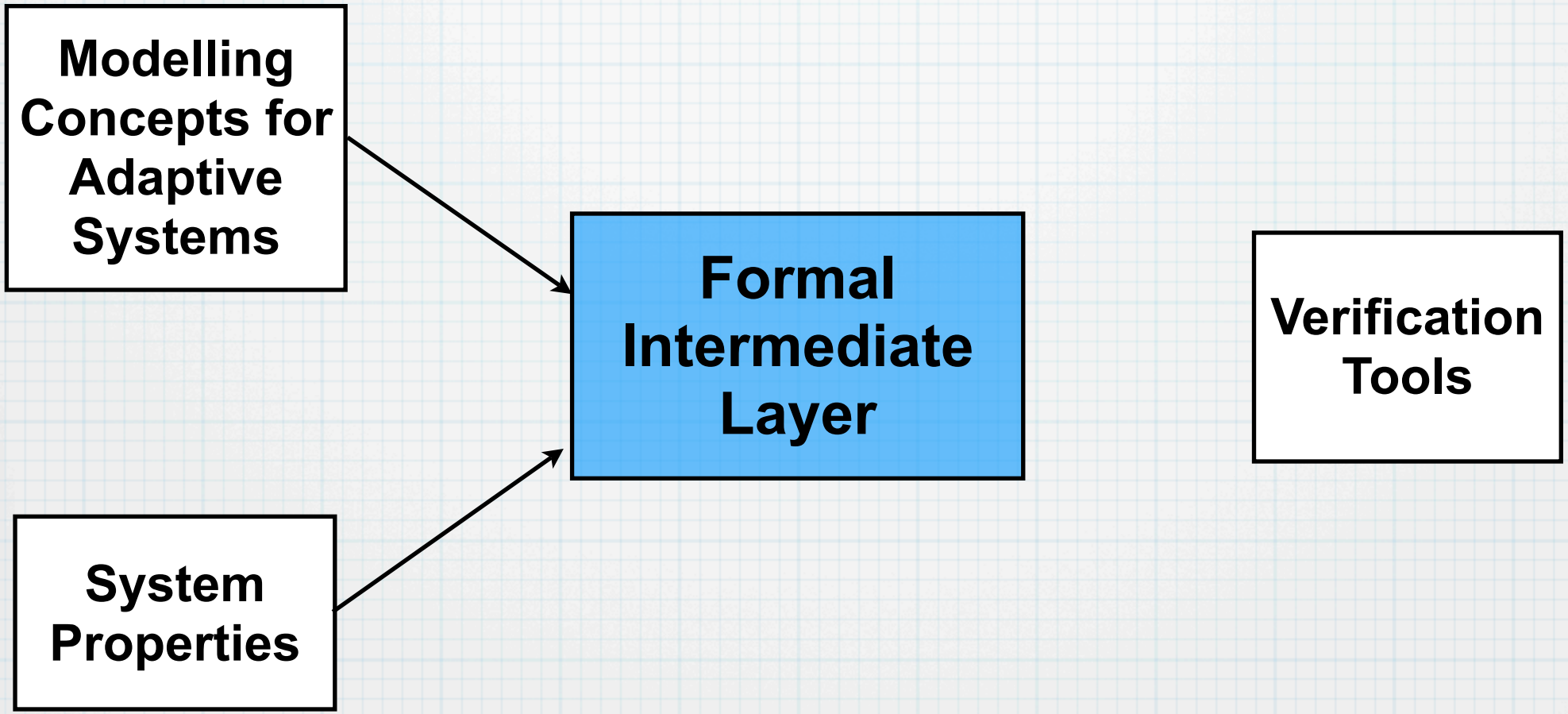
Verification Tools

System Properties

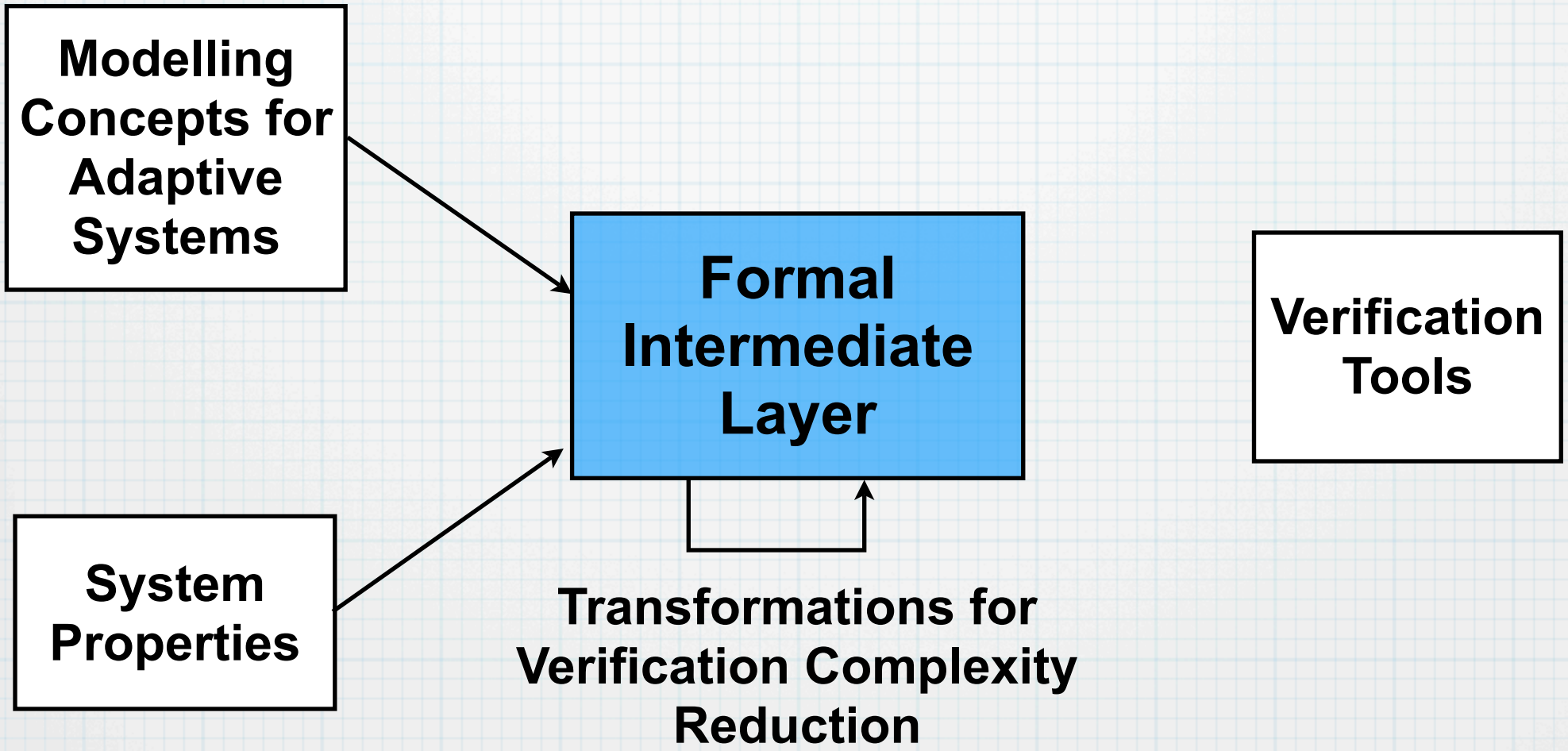
Integration - Outline



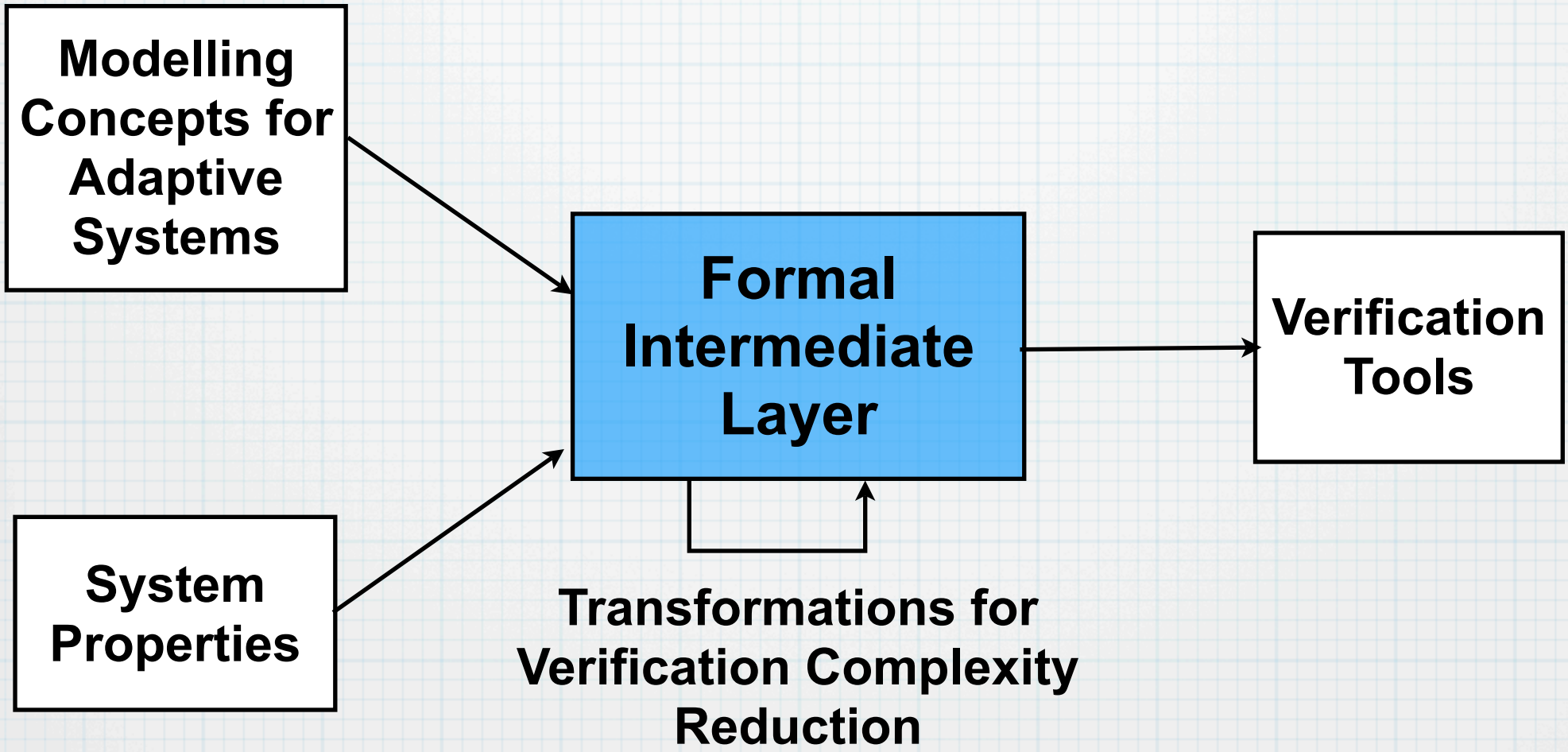
Integration - Outline



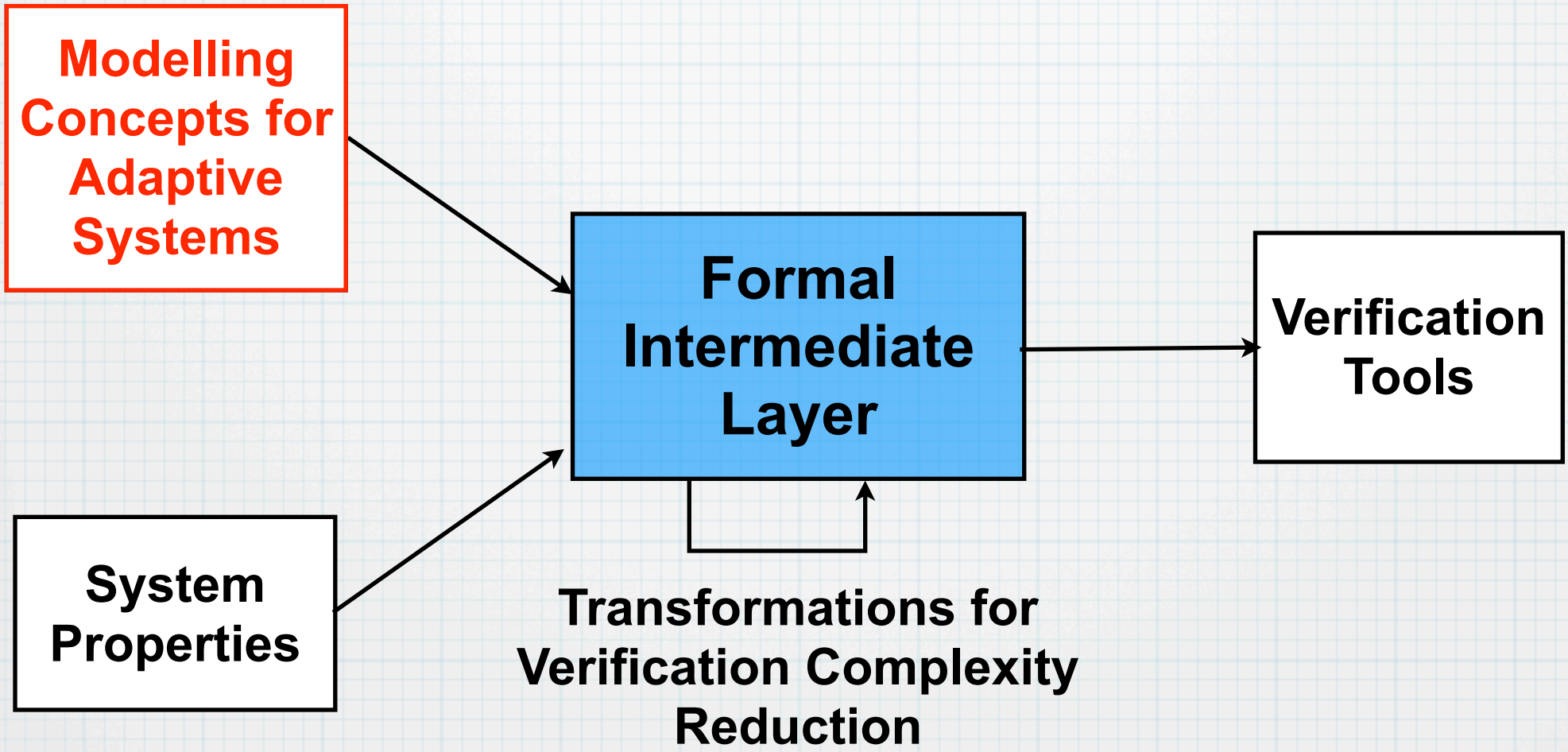
Integration - Outline



Integration - Outline



Integration - Outline

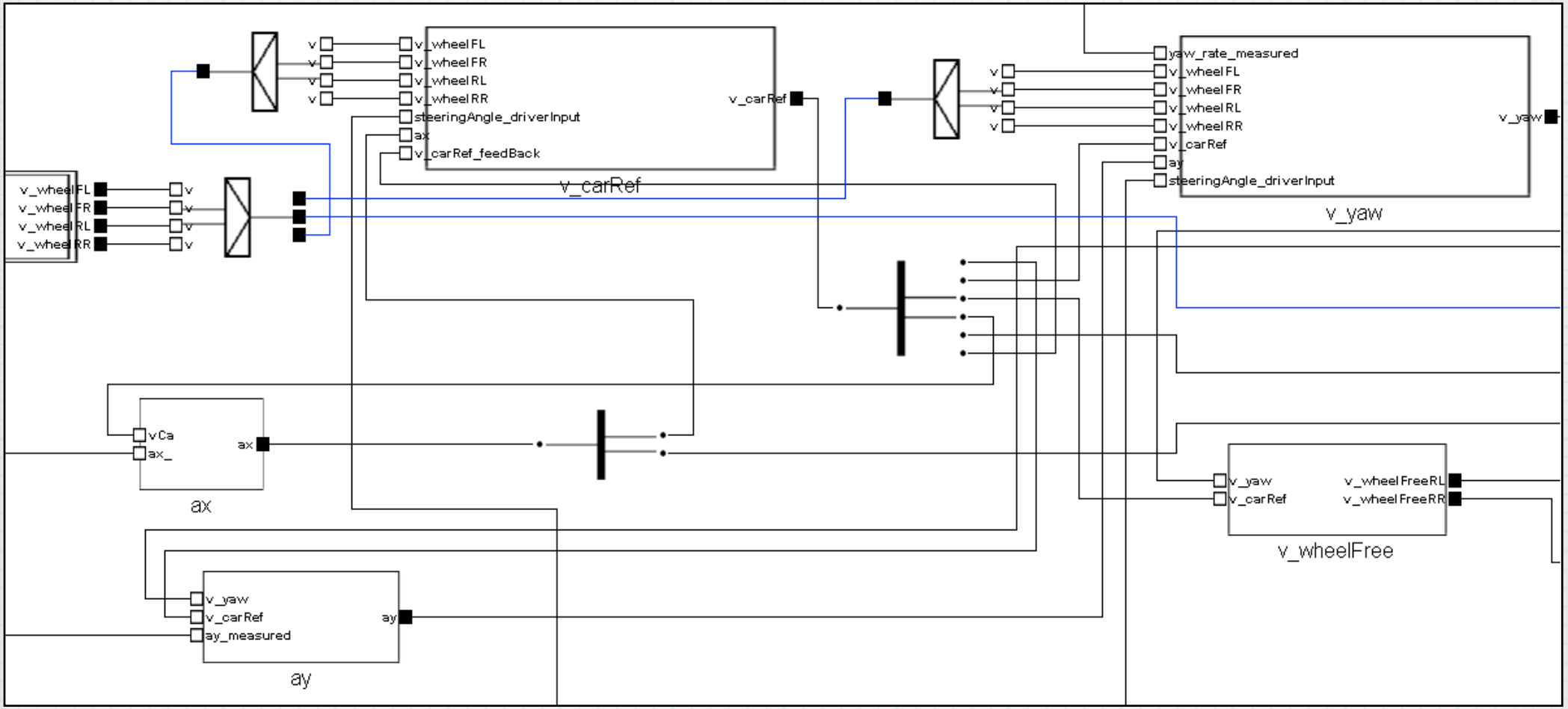


Adaptive System Modelling (MARS)

Main Concepts reducing Design Complexity:

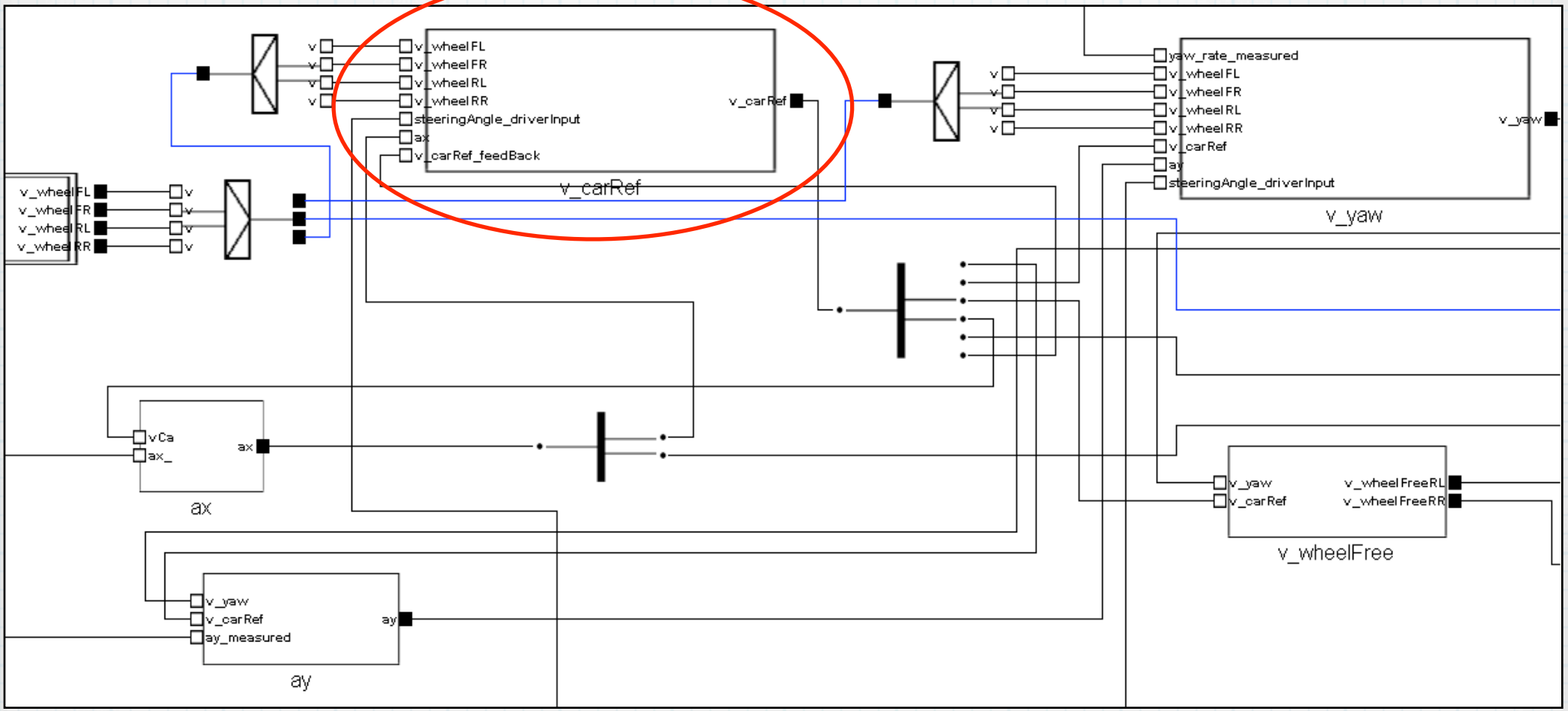
- **Modular System Structure**
- **Separation of Adaptation and Functionality in Modules**
- **Propagation of Adaptation by Qualities attached to Data**

Modular System Structure



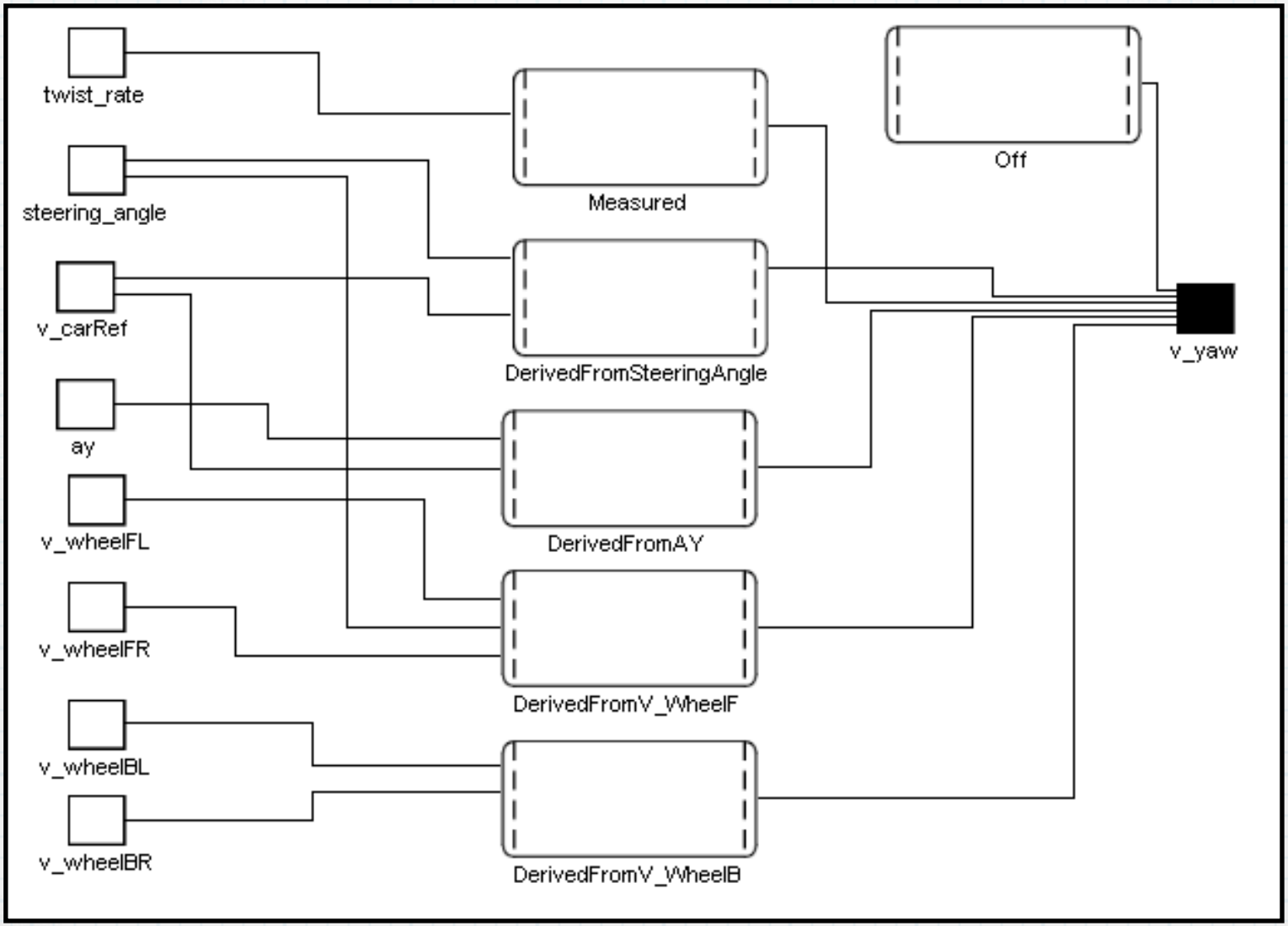
Modular System Structure

Module



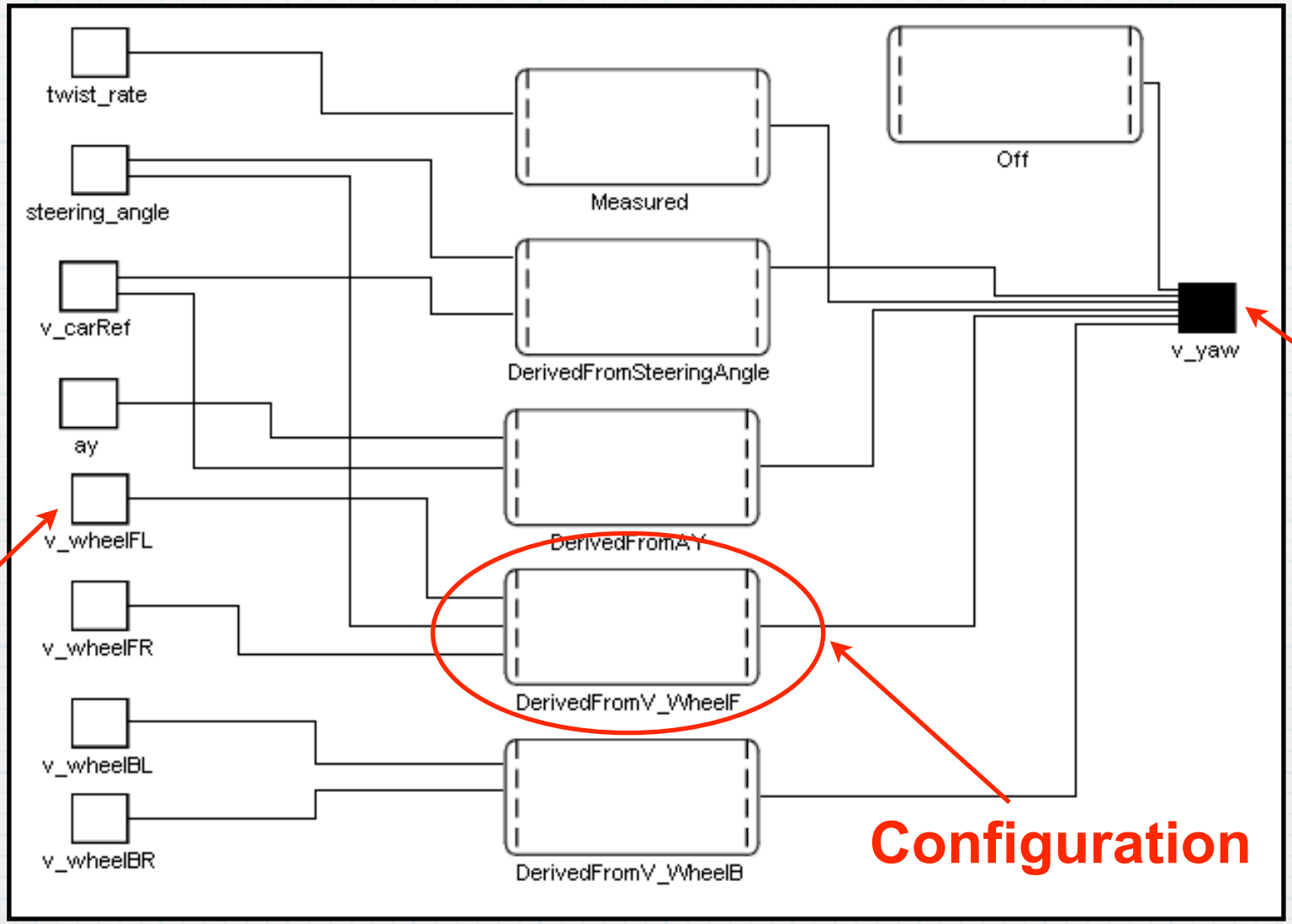
Adaptive MARS Modules

Example: Yaw Rate Module

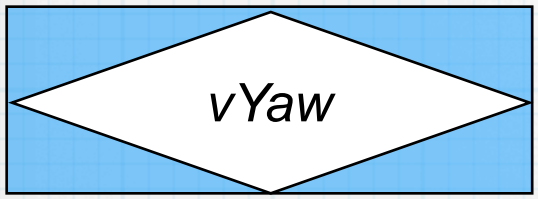


Adaptive MARS Modules

Example: Yaw Rate Module

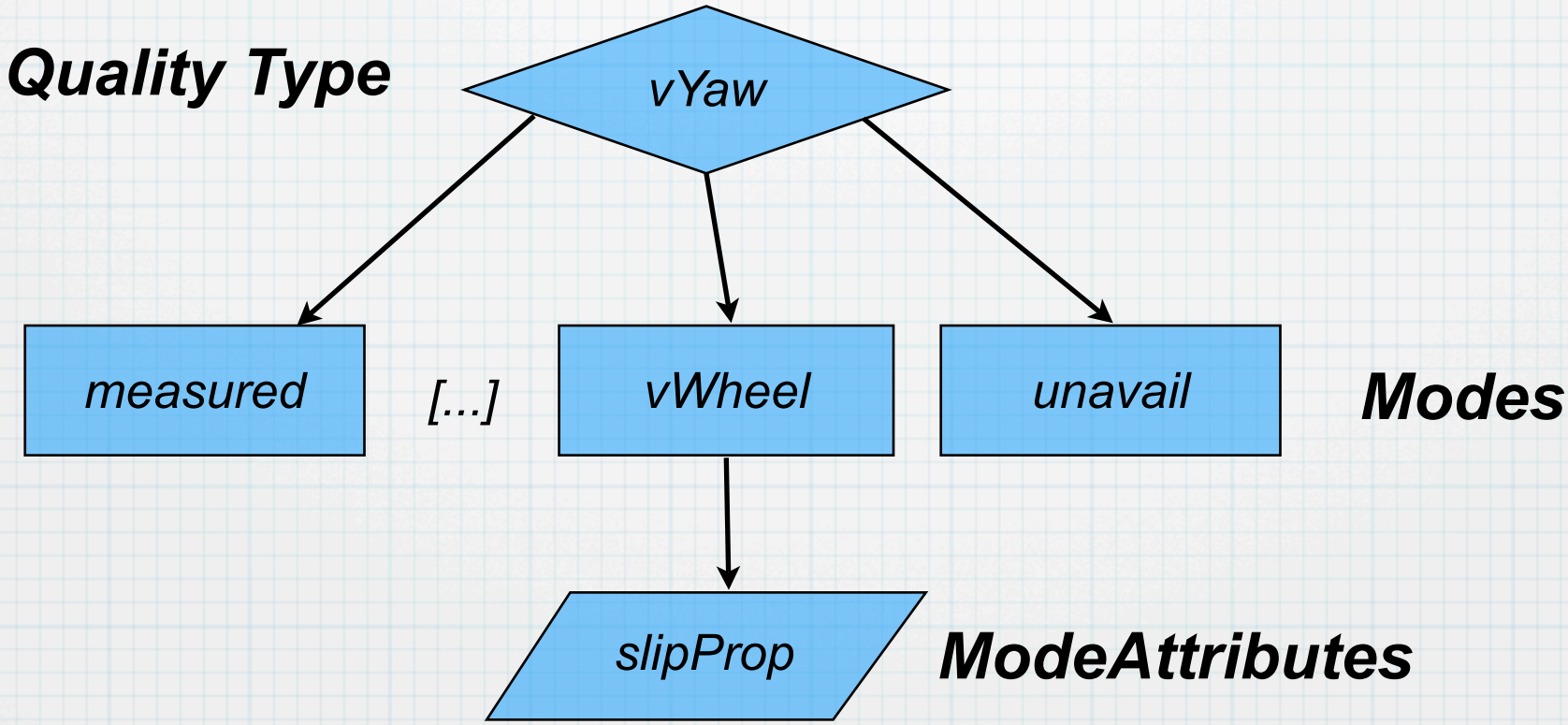


Adaptation Behaviour - Qualities



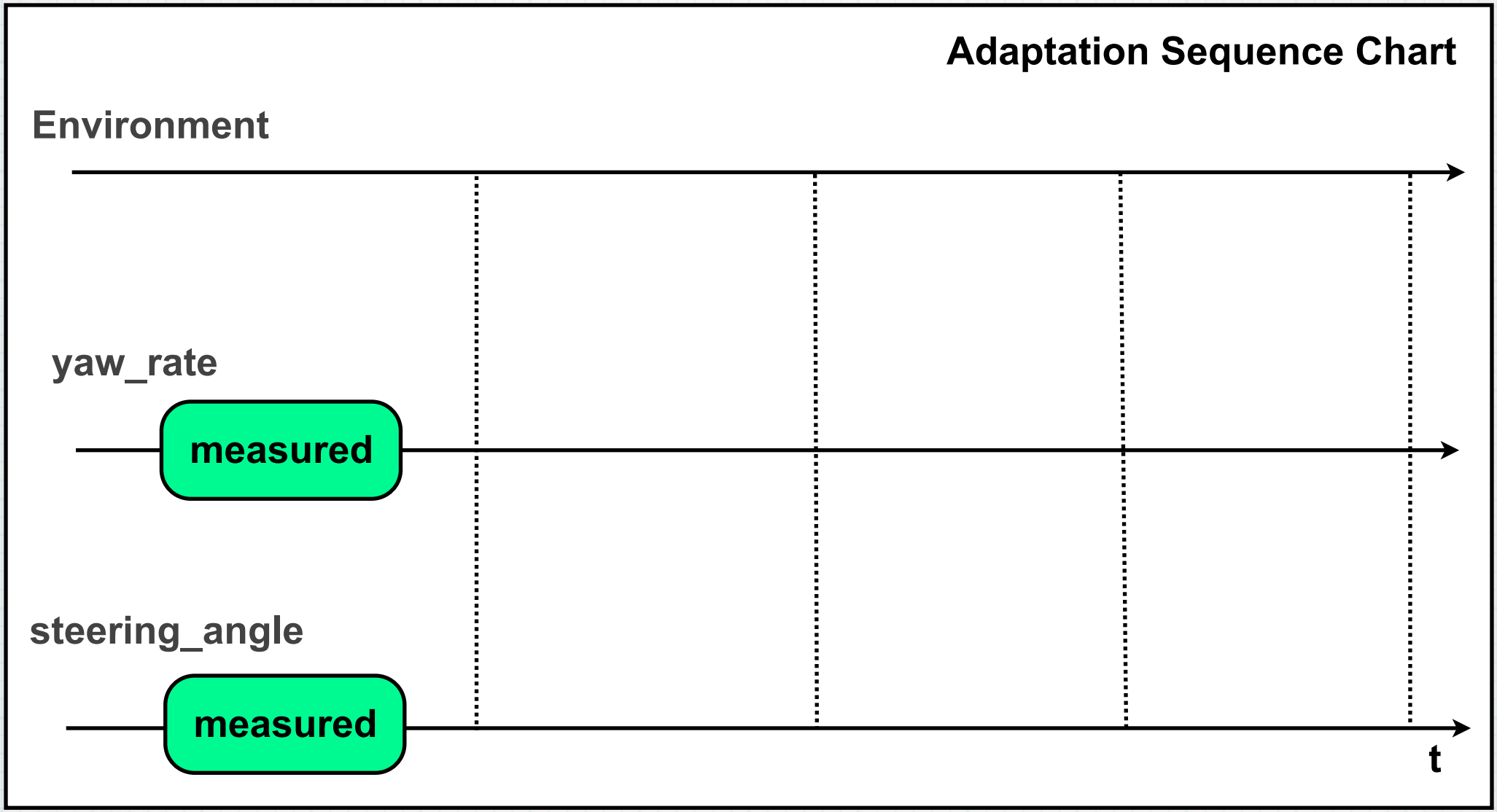
Quality-Extended Datatypes for Adaptive Systems

Dative = [data type, quality type]

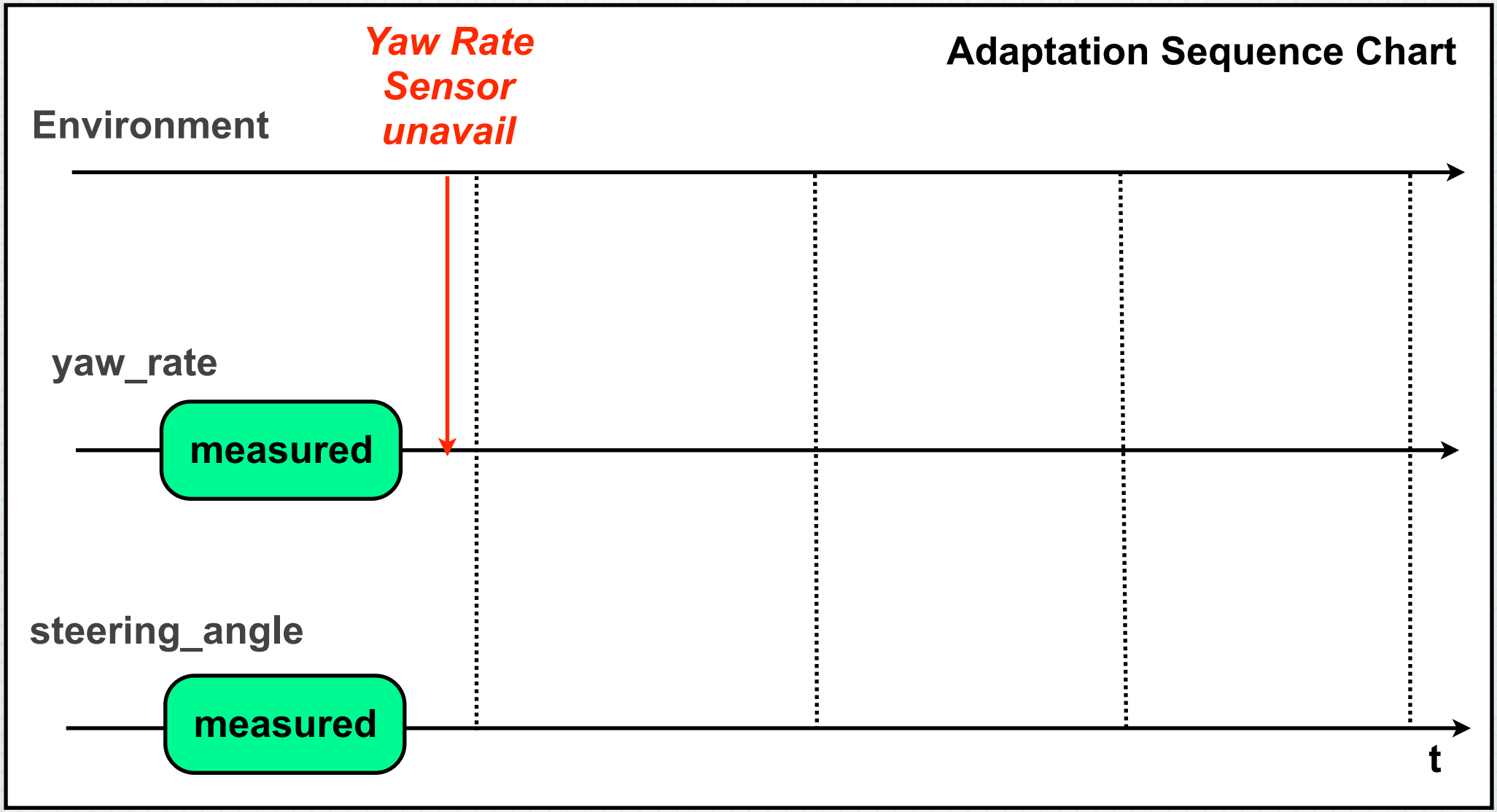


Modelling Adaptation

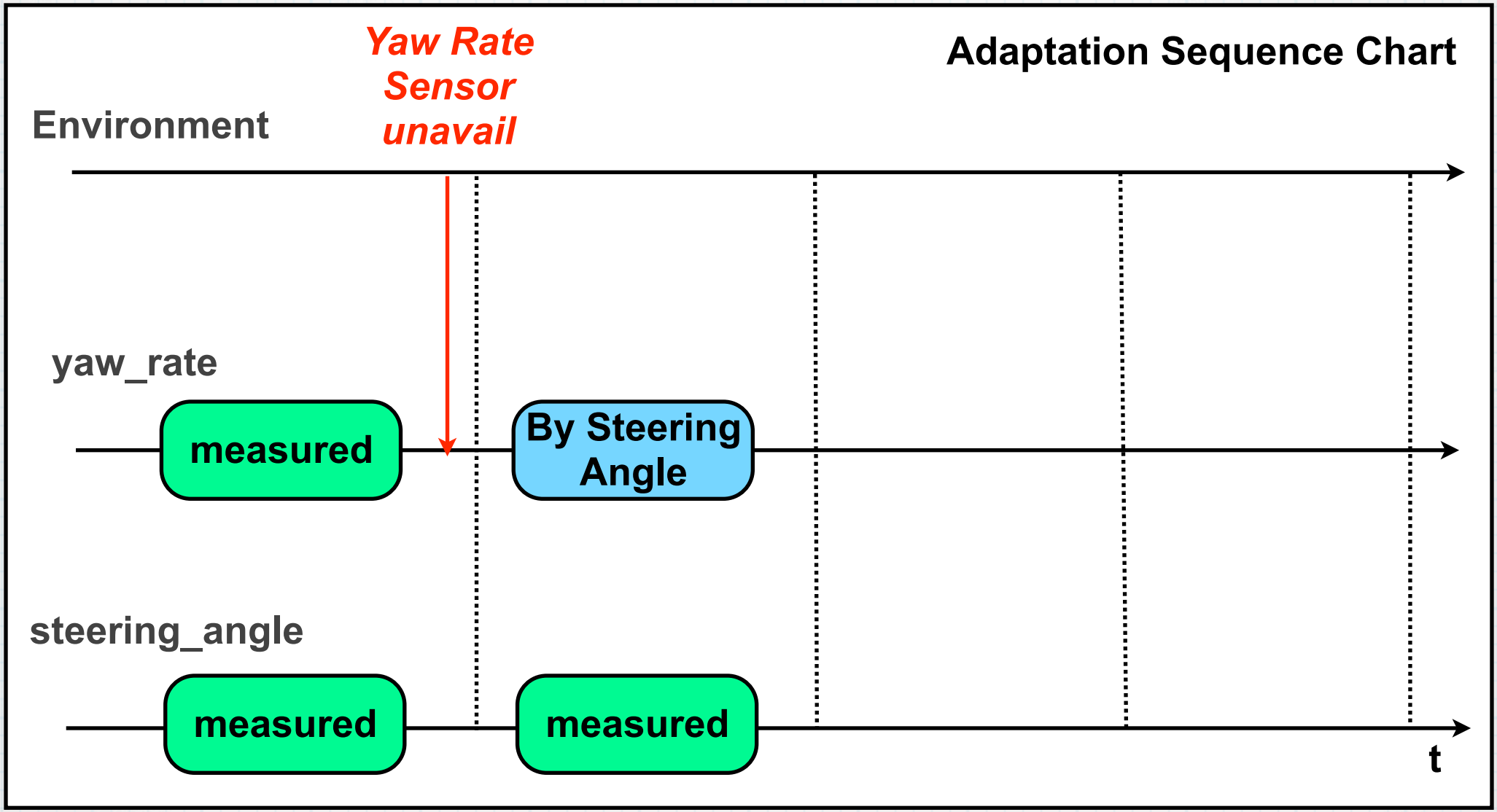
Modelling Adaptation



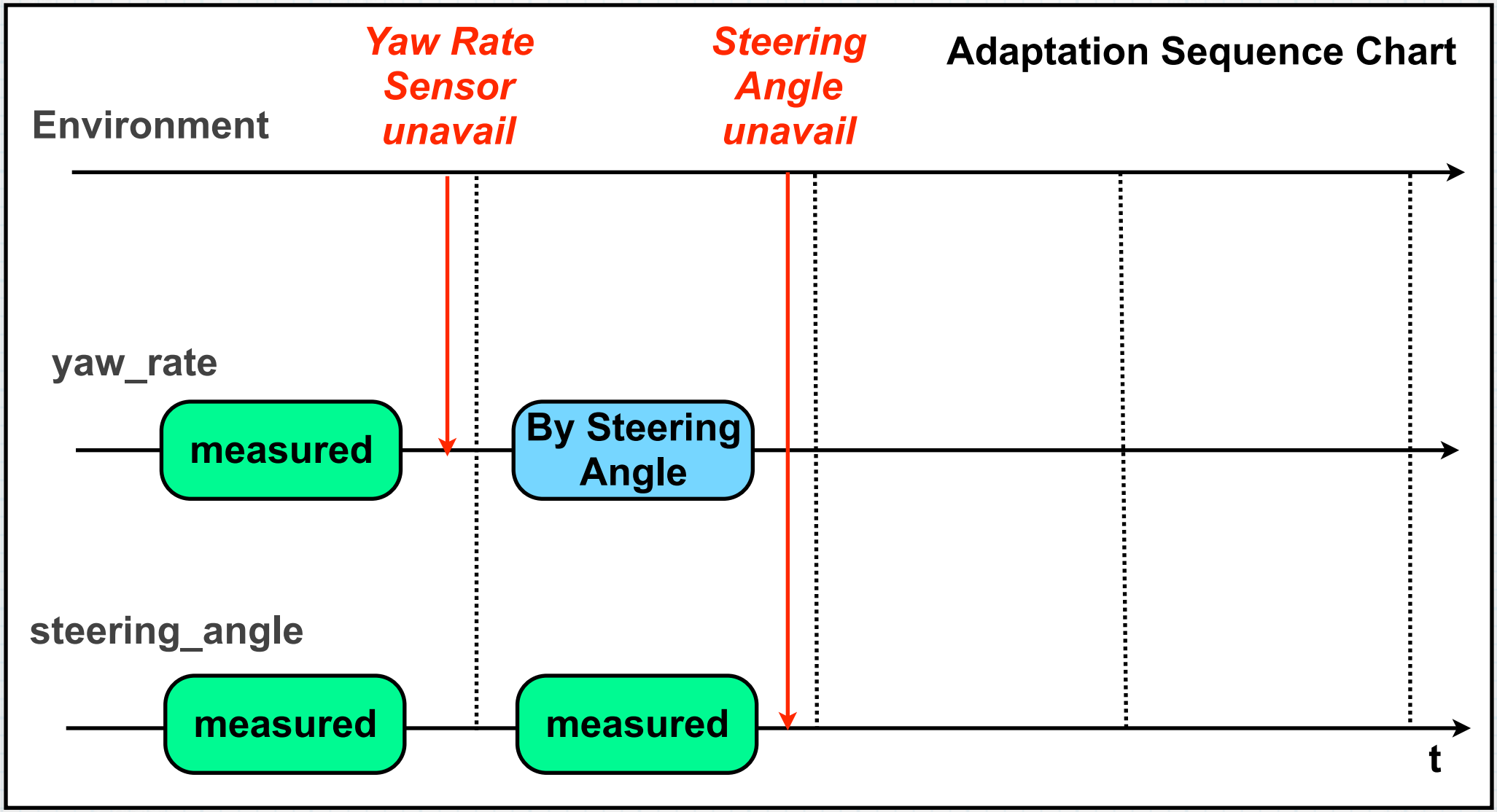
Modelling Adaptation



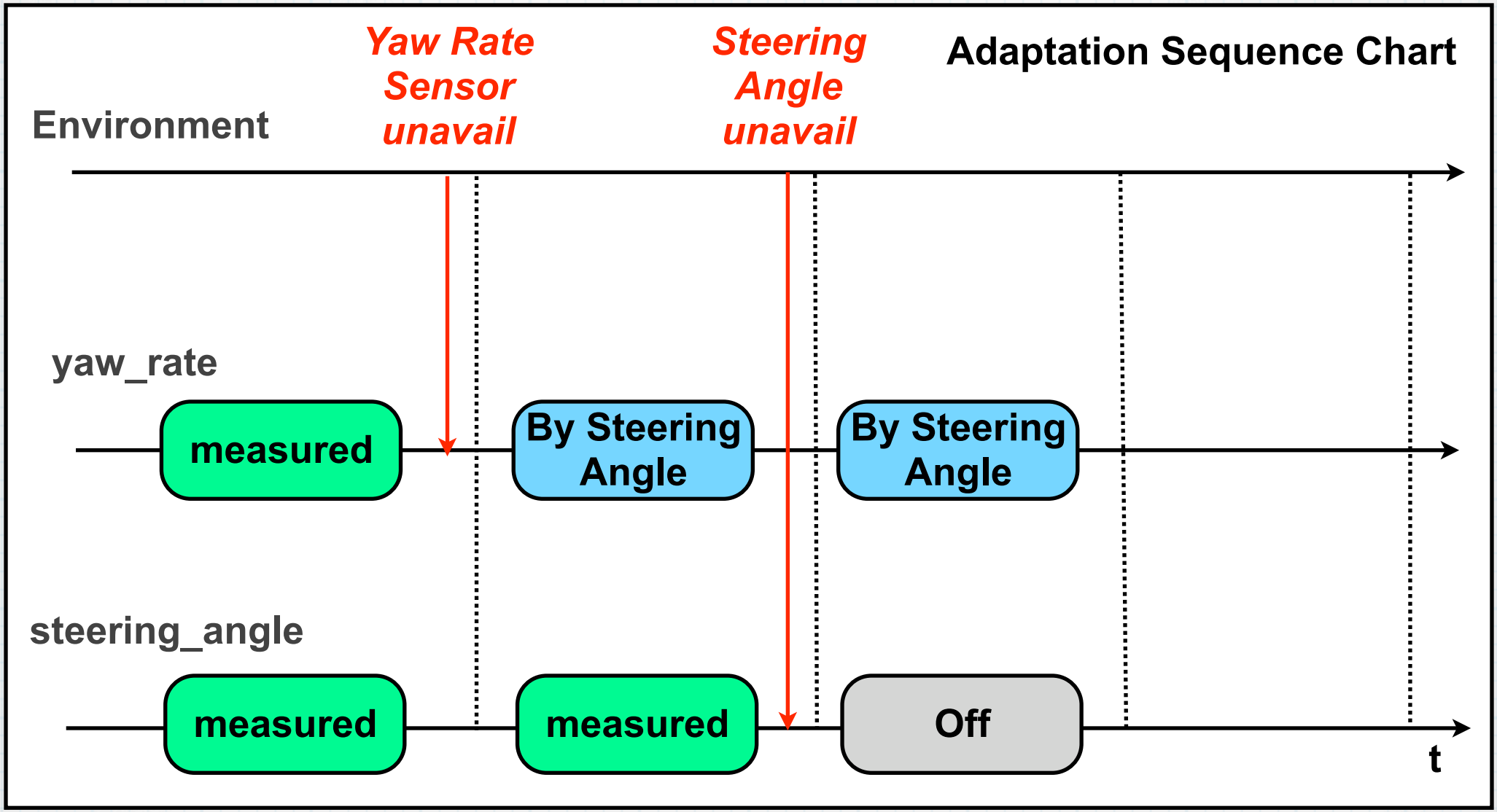
Modelling Adaptation



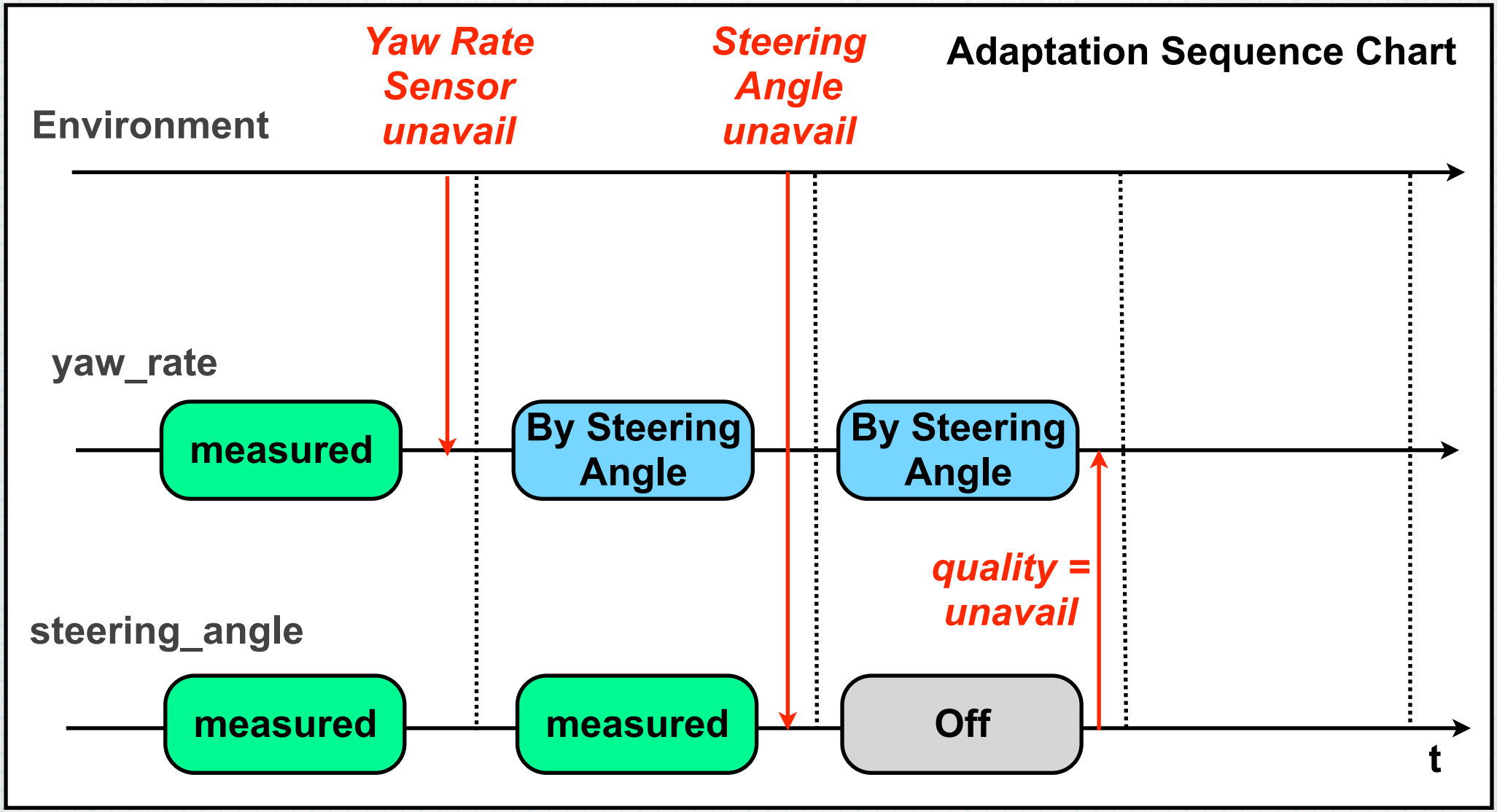
Modelling Adaptation



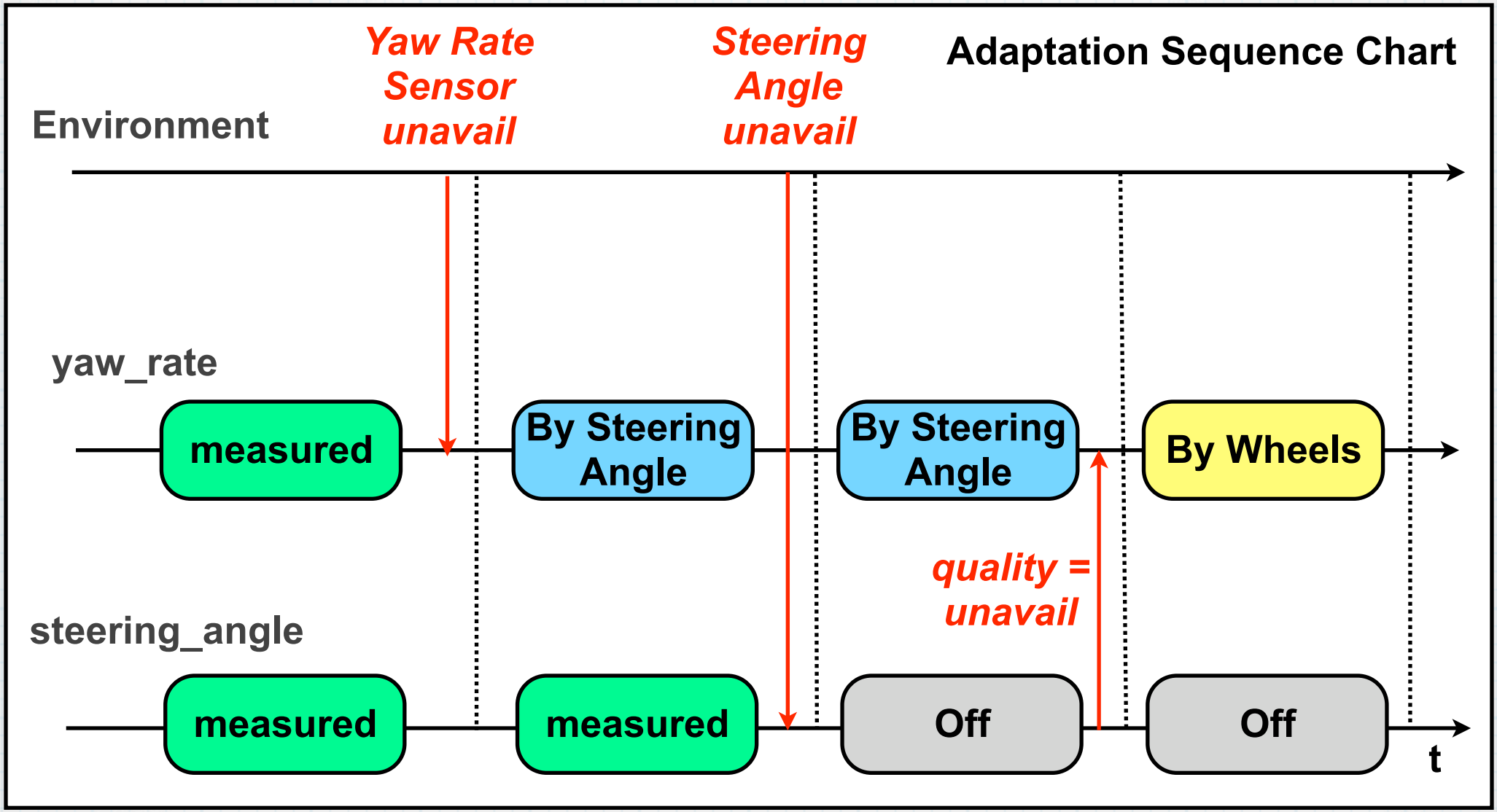
Modelling Adaptation



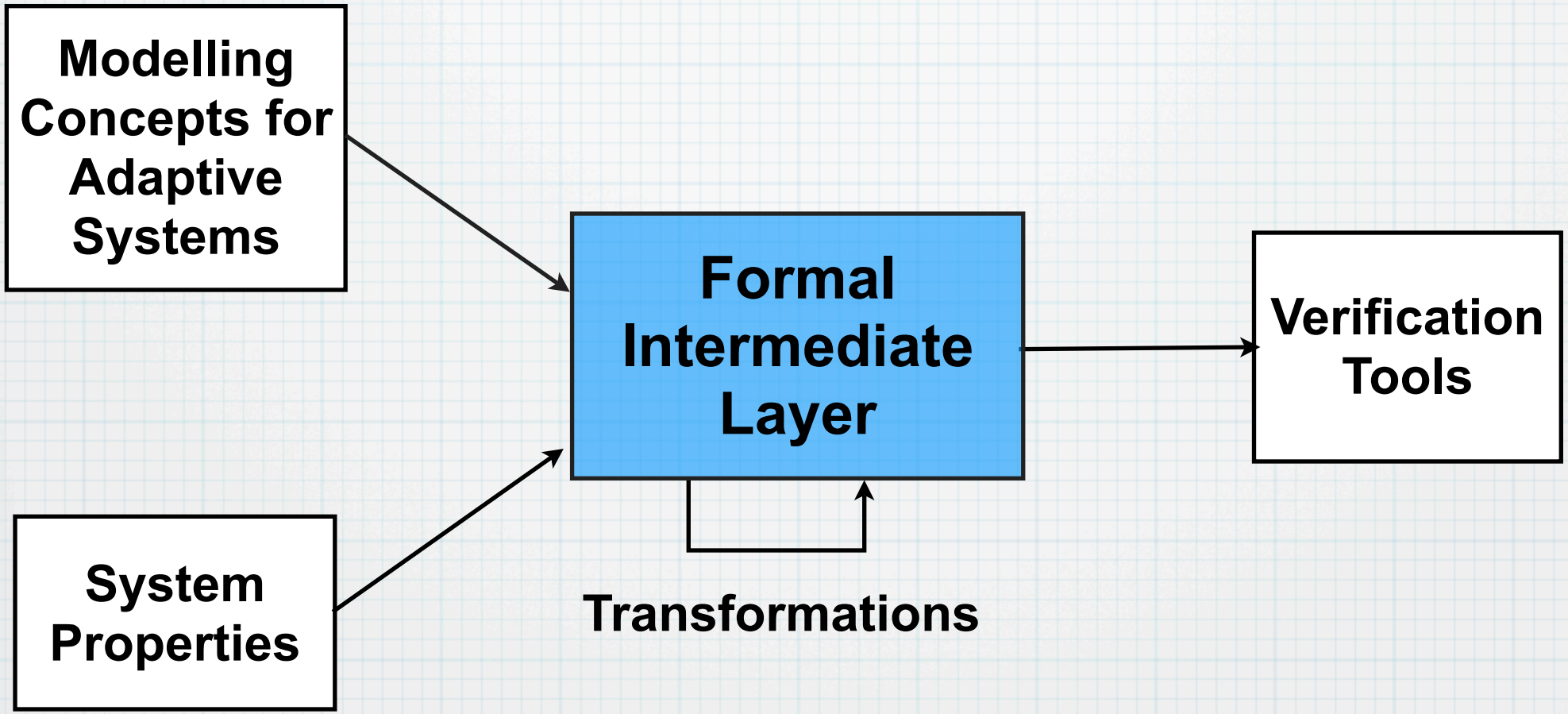
Modelling Adaptation



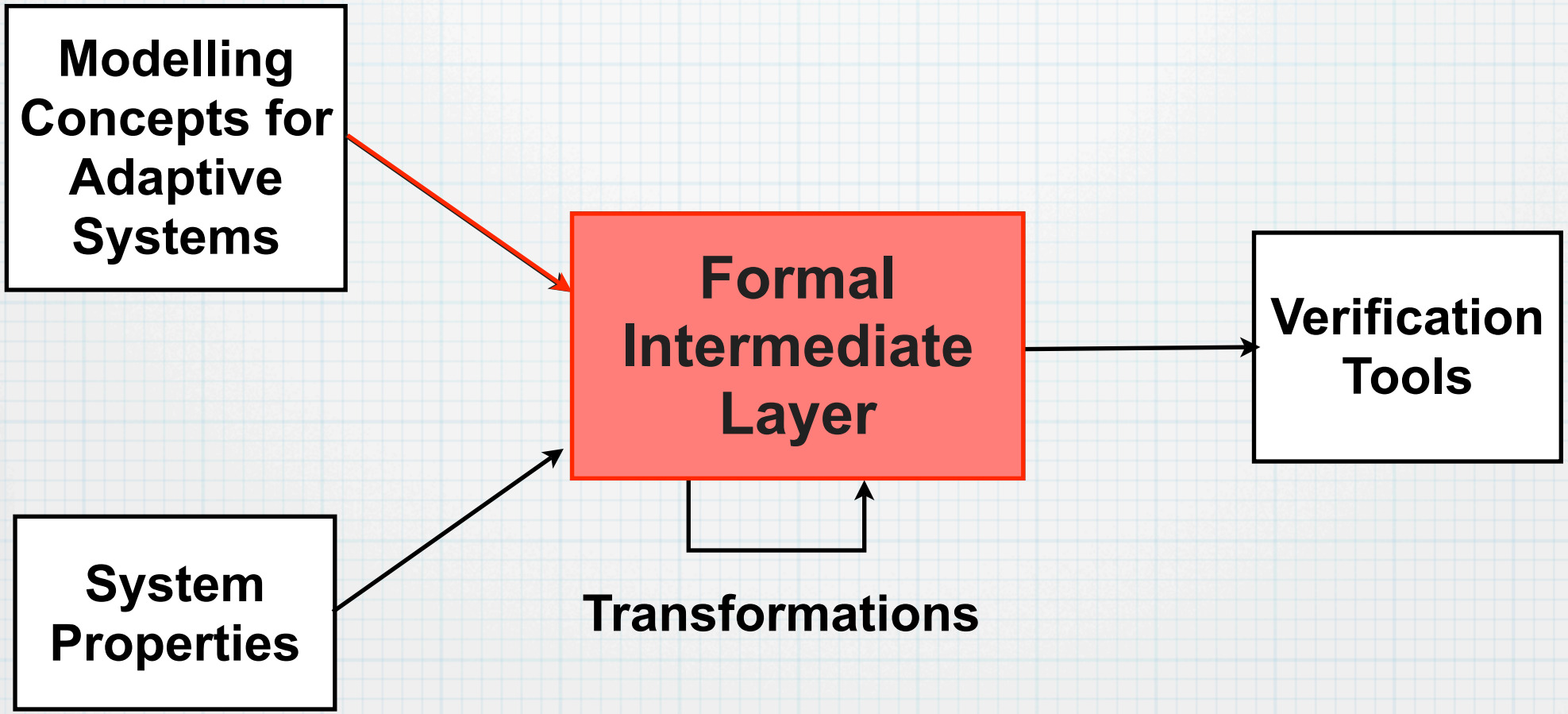
Modelling Adaptation



Integration - Outline



Integration - Outline



Synchronous Adaptive Systems

Design Goals:

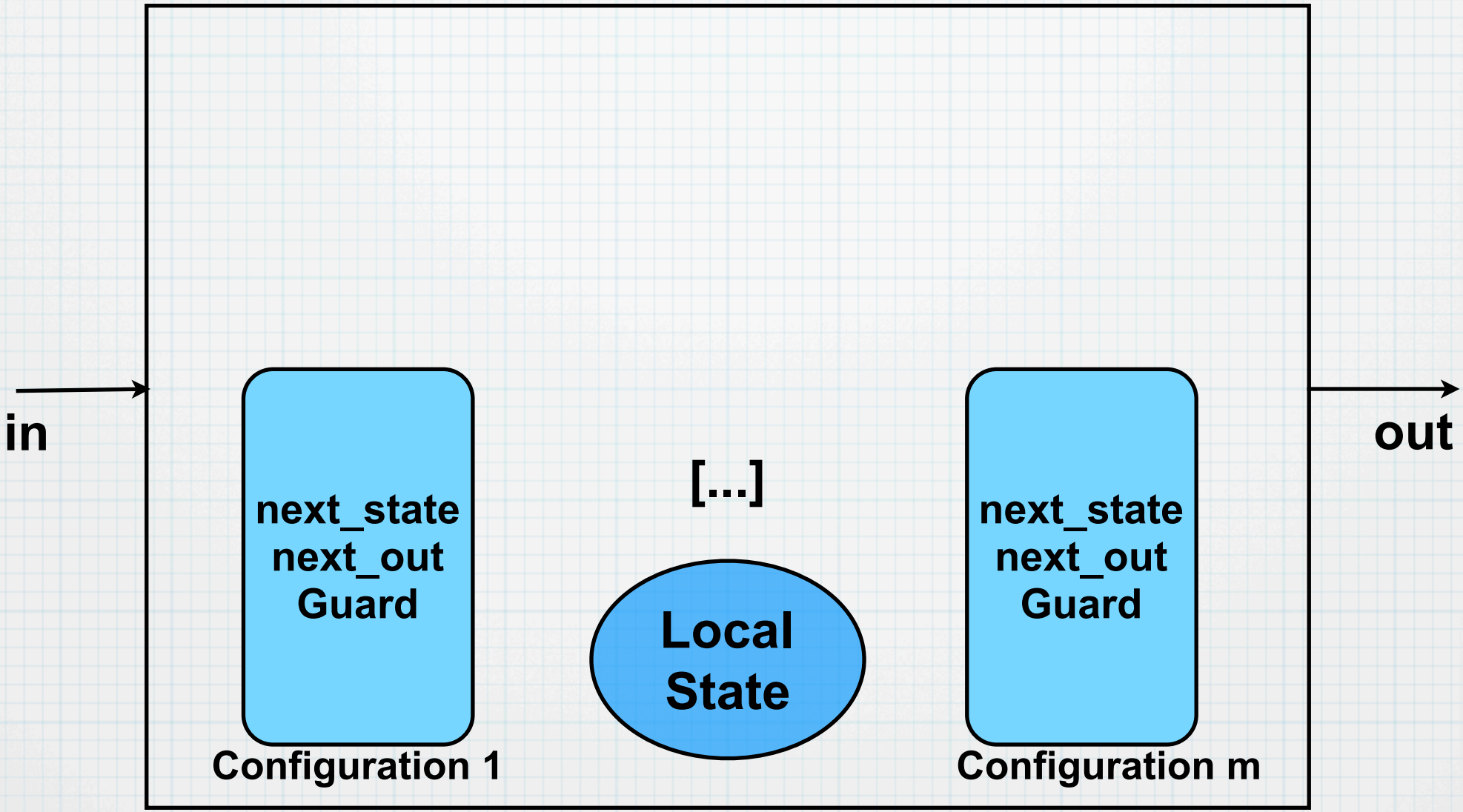
- Modular System Structure
- Explicit and Separated Modelling of Adaptation and Functionality
- Formal Mathematical Model

Main Concepts:

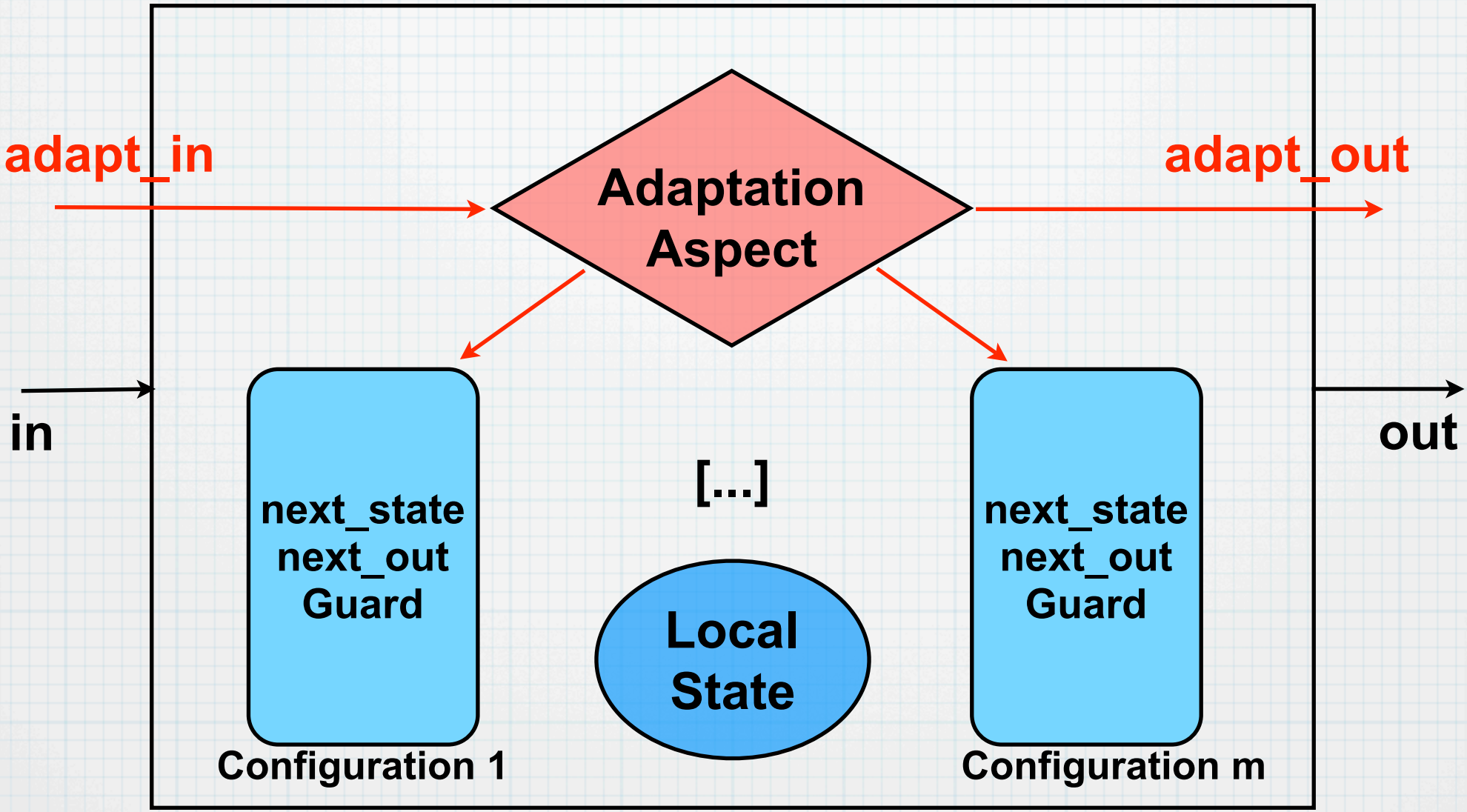
- Based on State-Transition Systems
- Uses Adaptation Aspect

SAS Module

SAS Module

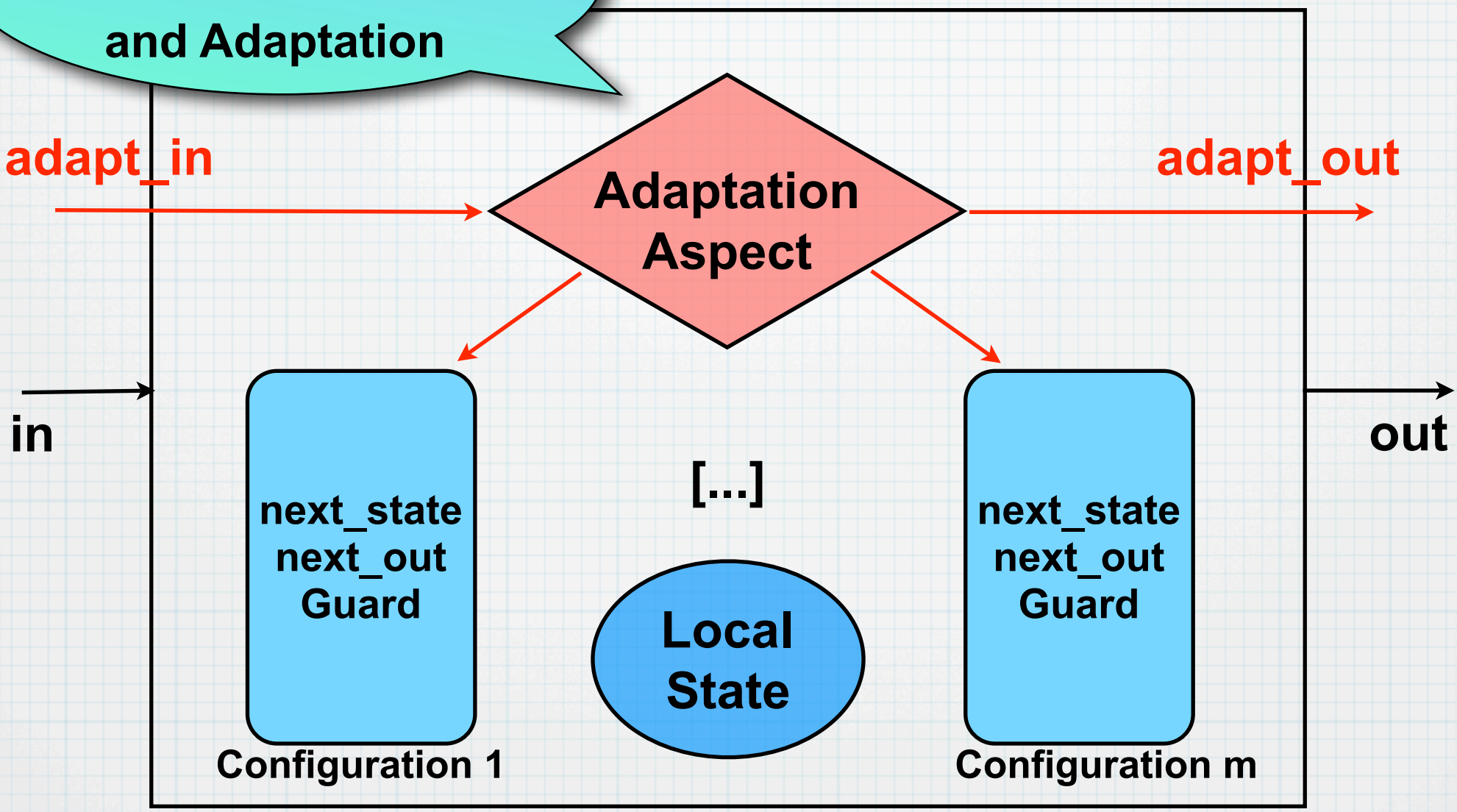

in
next_state
next_out
Guard
Configuration 1
[...]
Local
State
next_state
next_out
Guard
Configuration m
out

SAS Module


adapt_in
Adaptation Aspect
adapt_out
in
out
next_state
next_out
Guard
[...]
Local State
next_state
next_out
Guard
Configuration 1
Configuration m

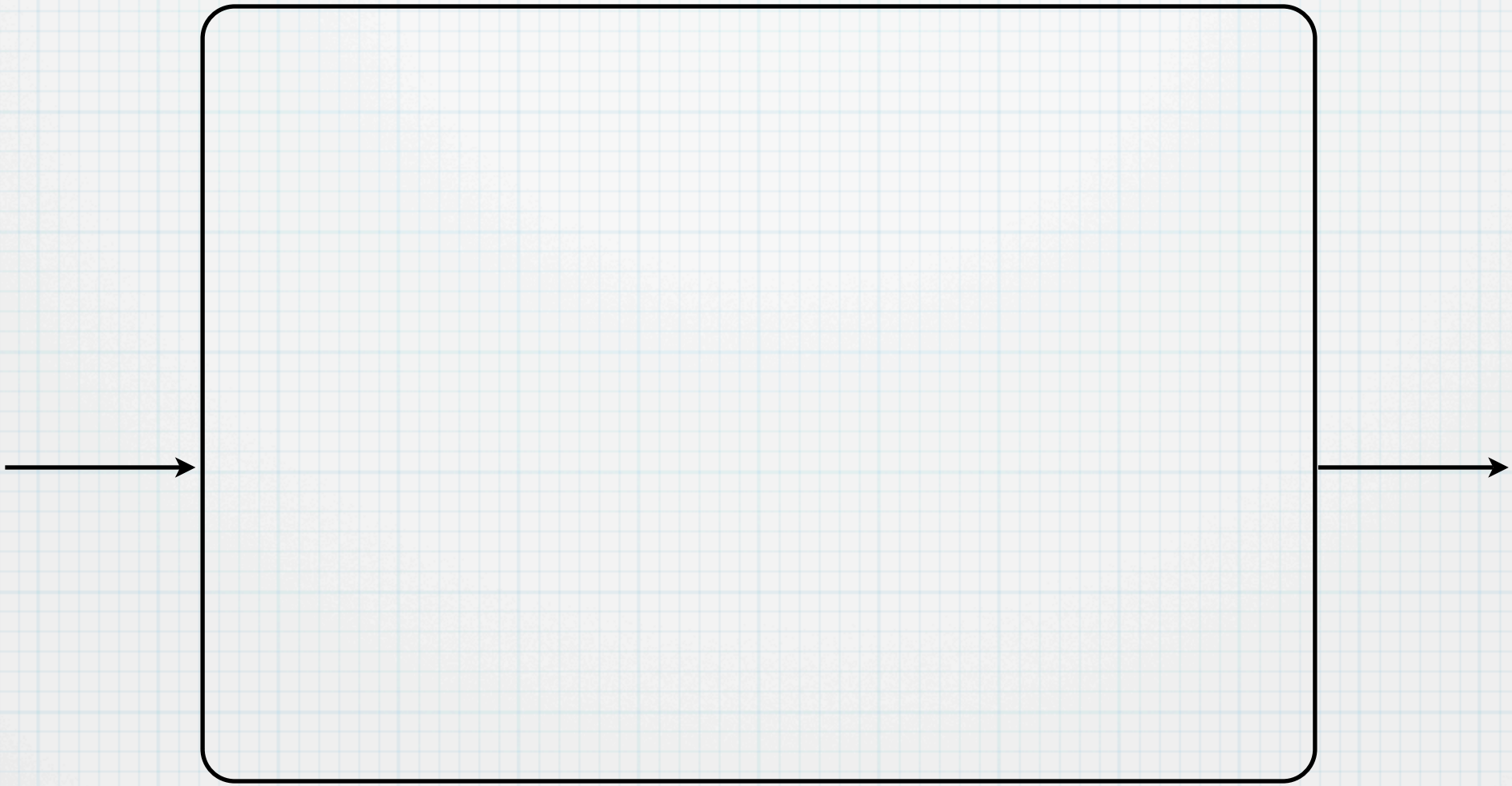
AS Module

Independent Modelling of Functionality and Adaptation



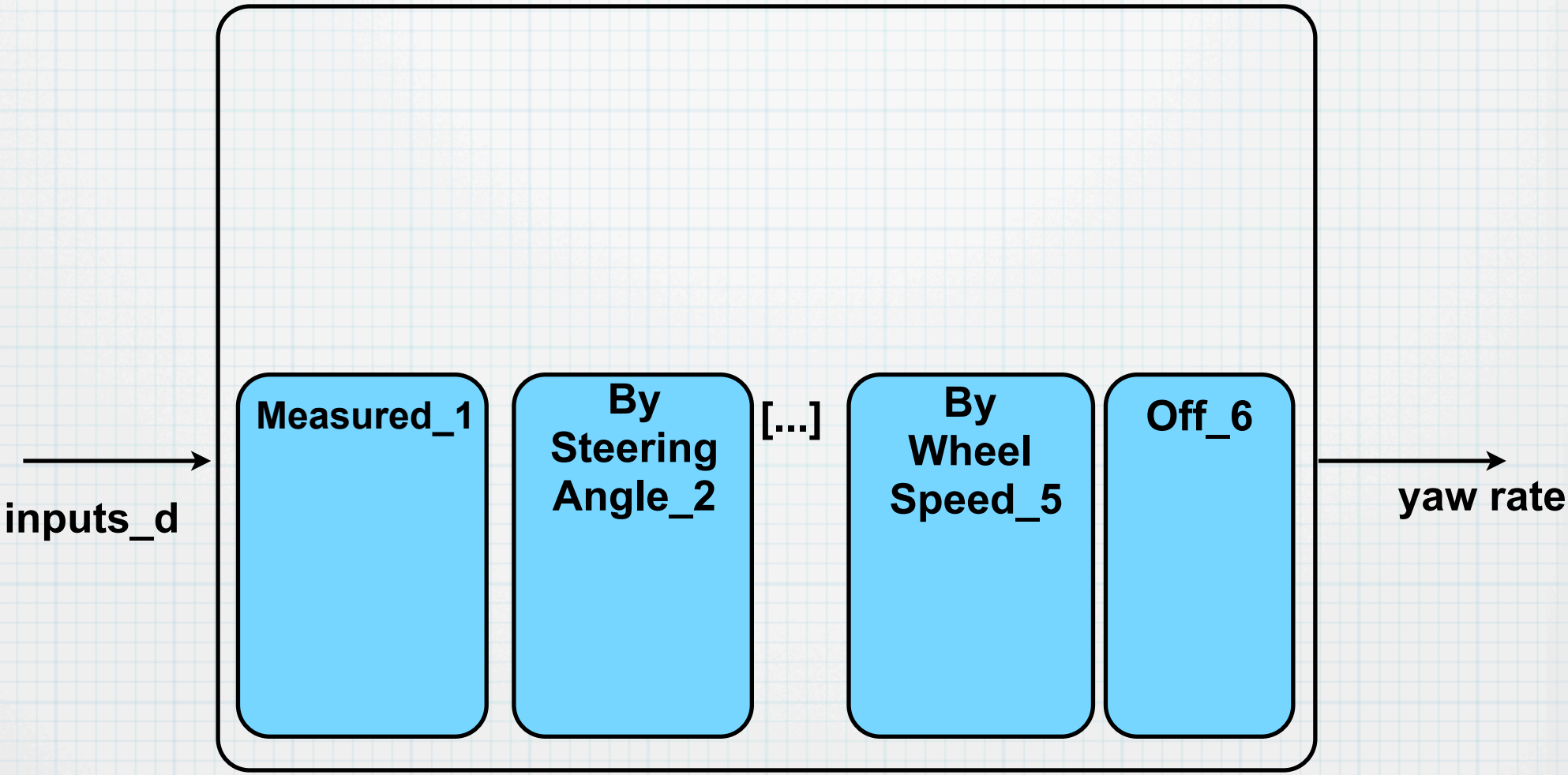
Traction Control System

Example: Yaw Rate Module



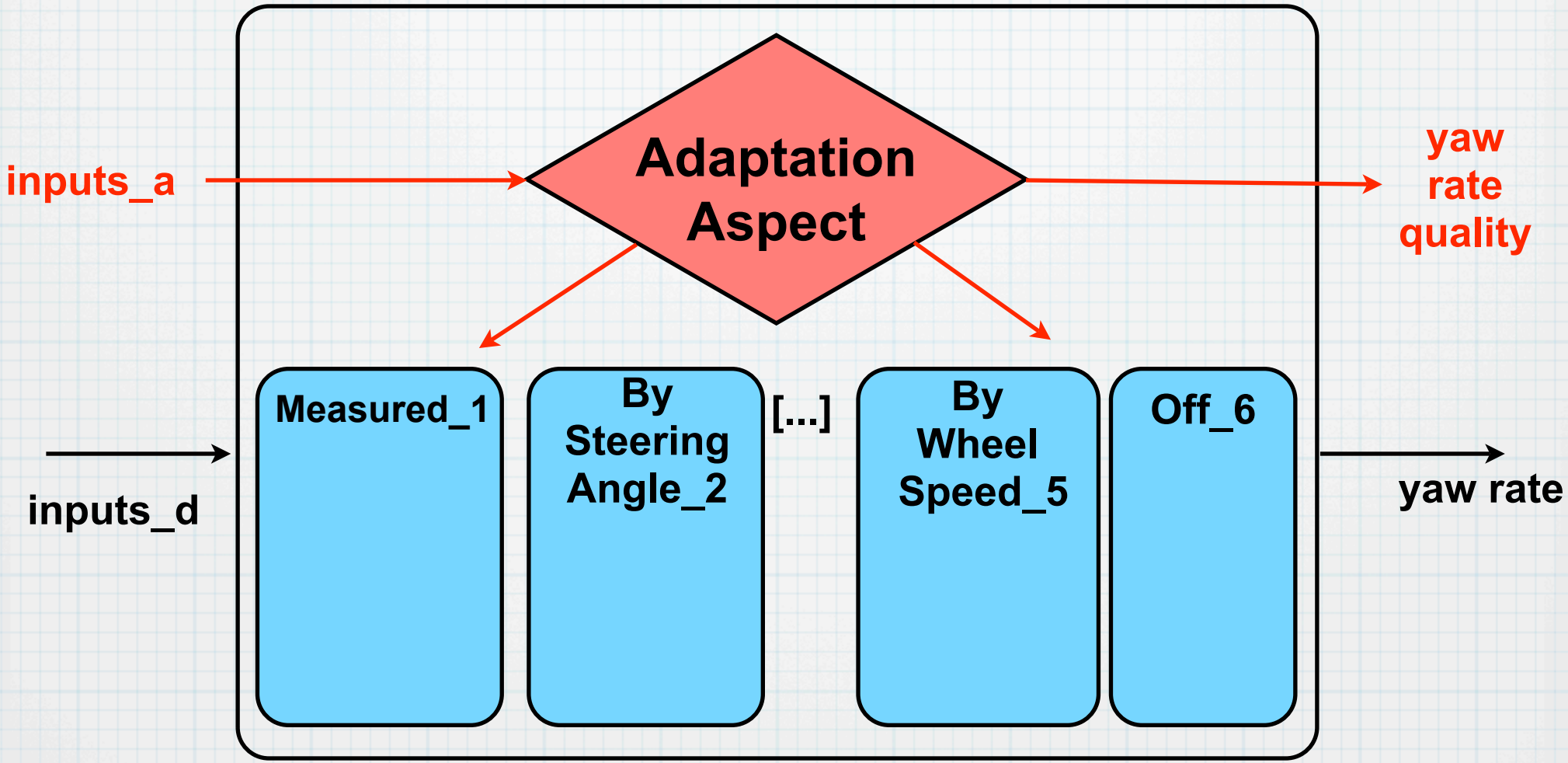
Traction Control System

Example: Yaw Rate Module

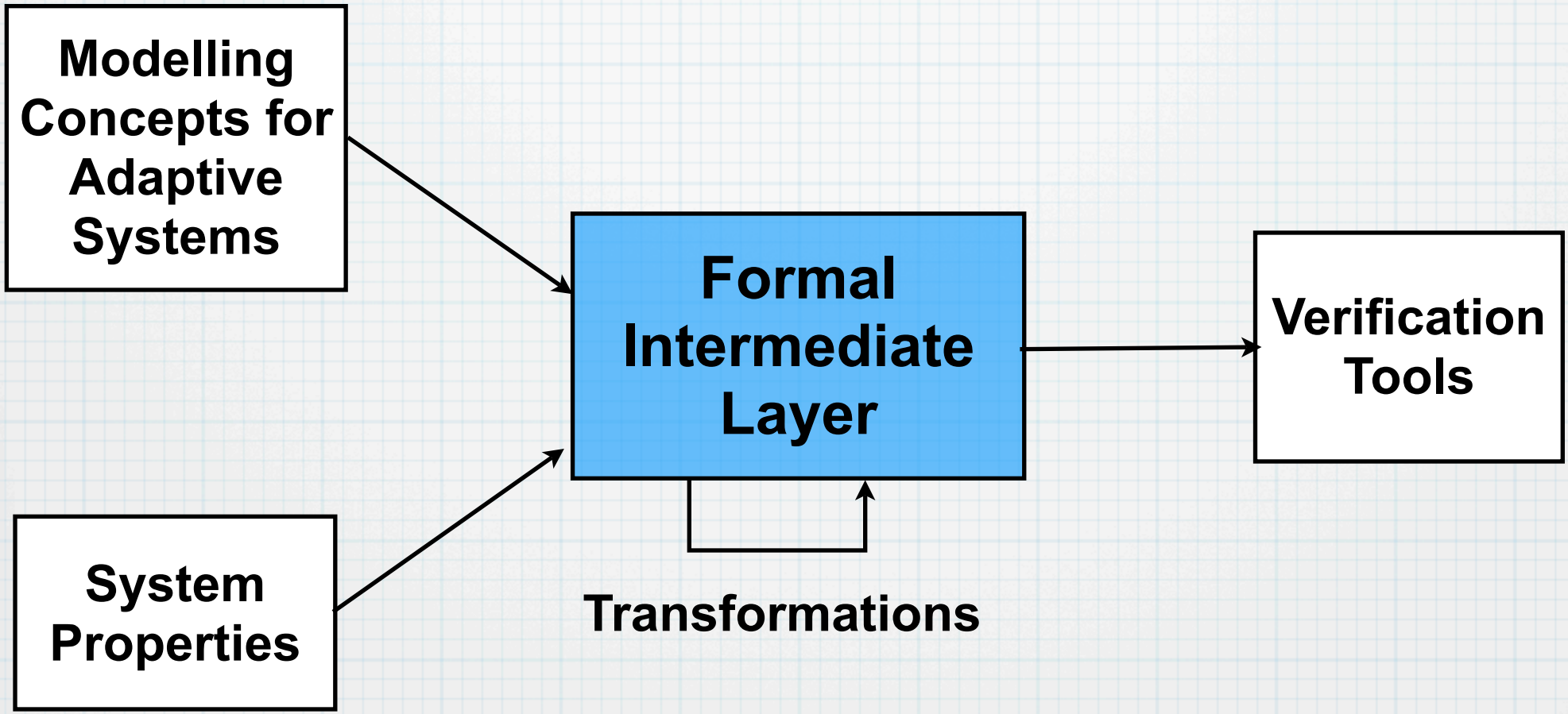


Traction Control System

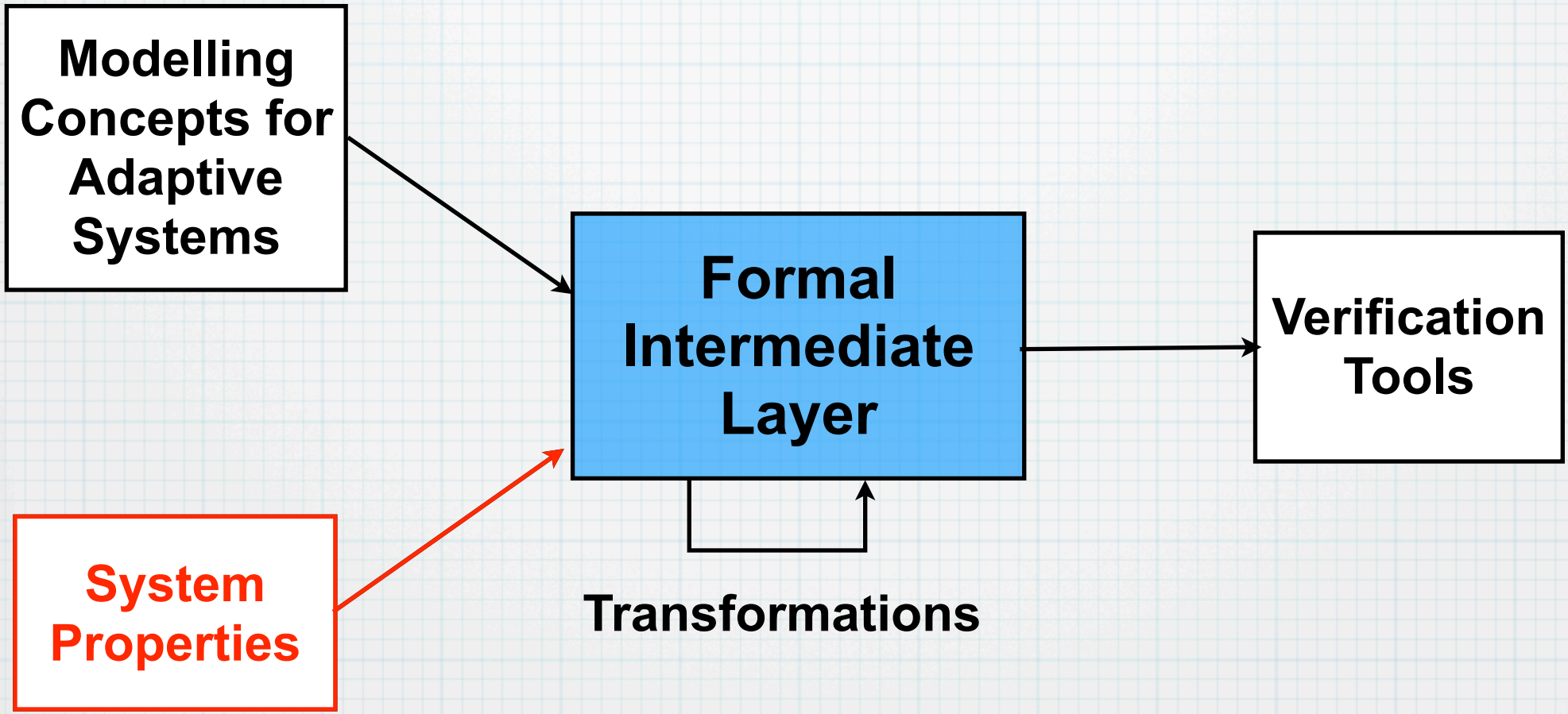
Example: Yaw Rate Module



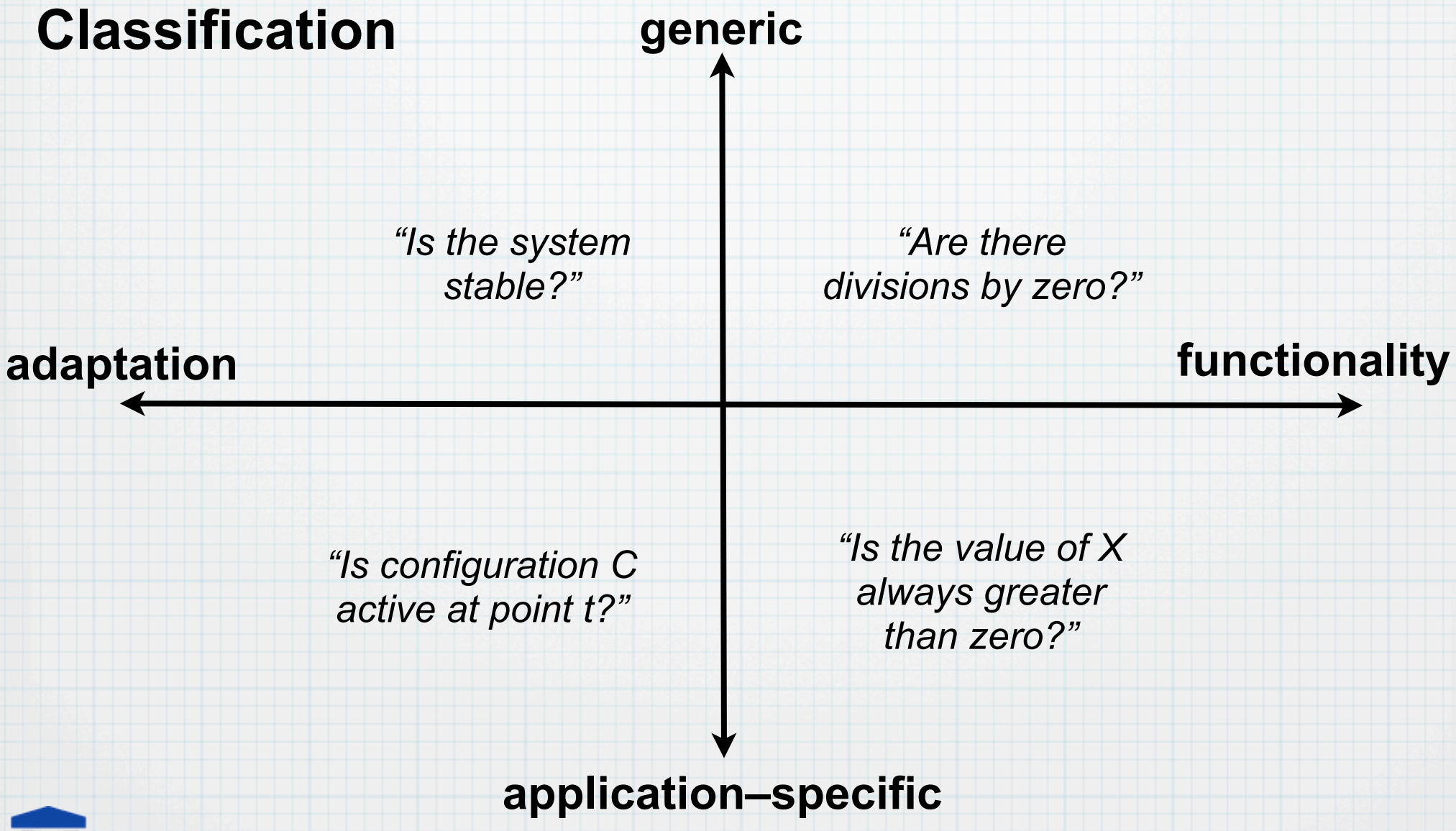
Integration - Outline



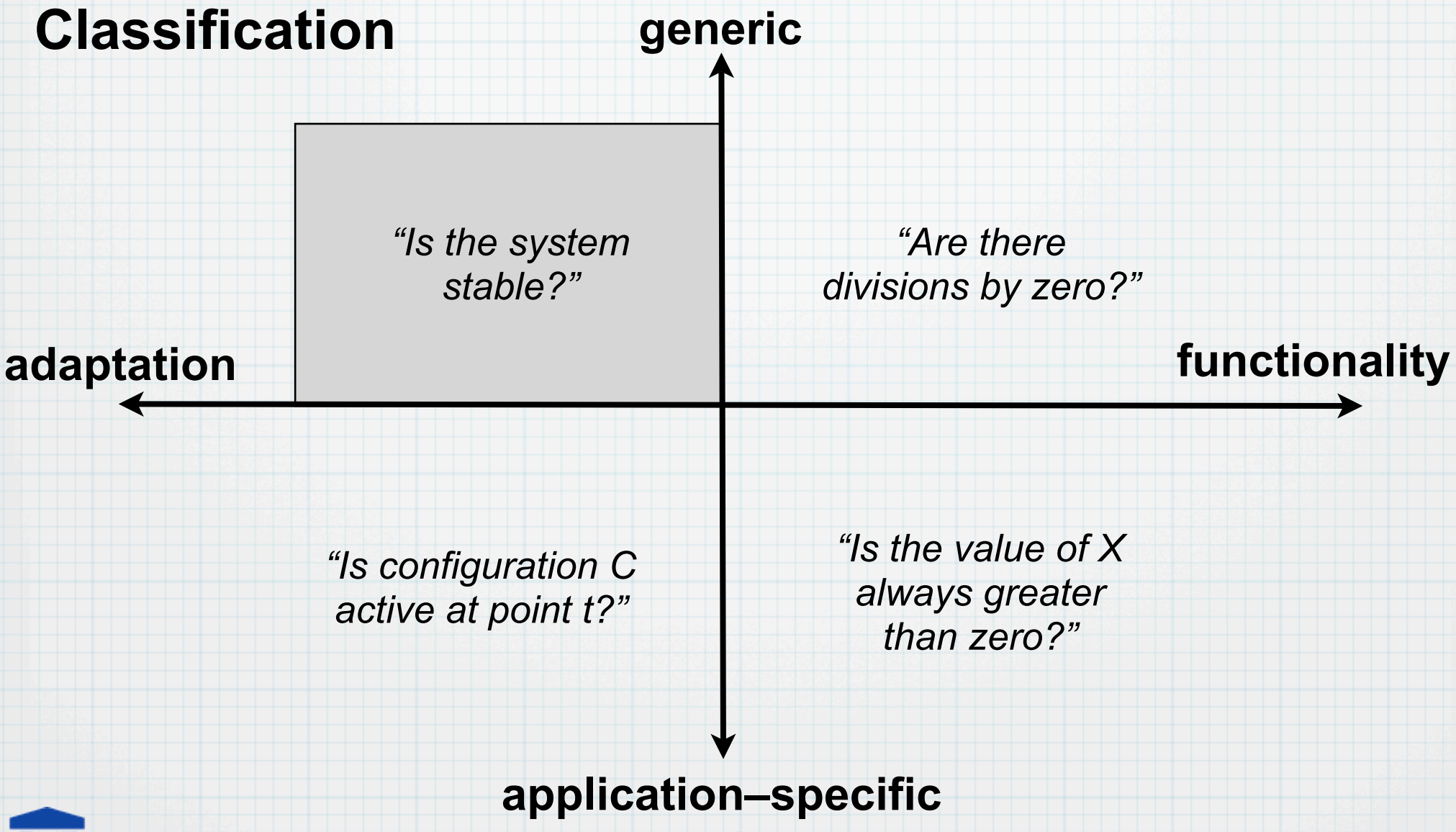
Integration - Outline



System Properties



System Properties



Adaptive Generic Properties

- **No module gets stuck in the default configuration ‘off ’:**

$$\mathsf{AG}(c = \text{off} \rightarrow \mathsf{EF} c \neq \text{off})$$

- **Every module can reach all configurations at all times:**

$$\mathsf{AG}(\bigwedge_{i=1}^{n} \mathsf{EF}\ c = \text{config}_i)$$

- **No inconsistent states can be reached:**

$$\mathsf{AG}(\bigvee_{i=1}^{n} c = \text{config}_i)$$

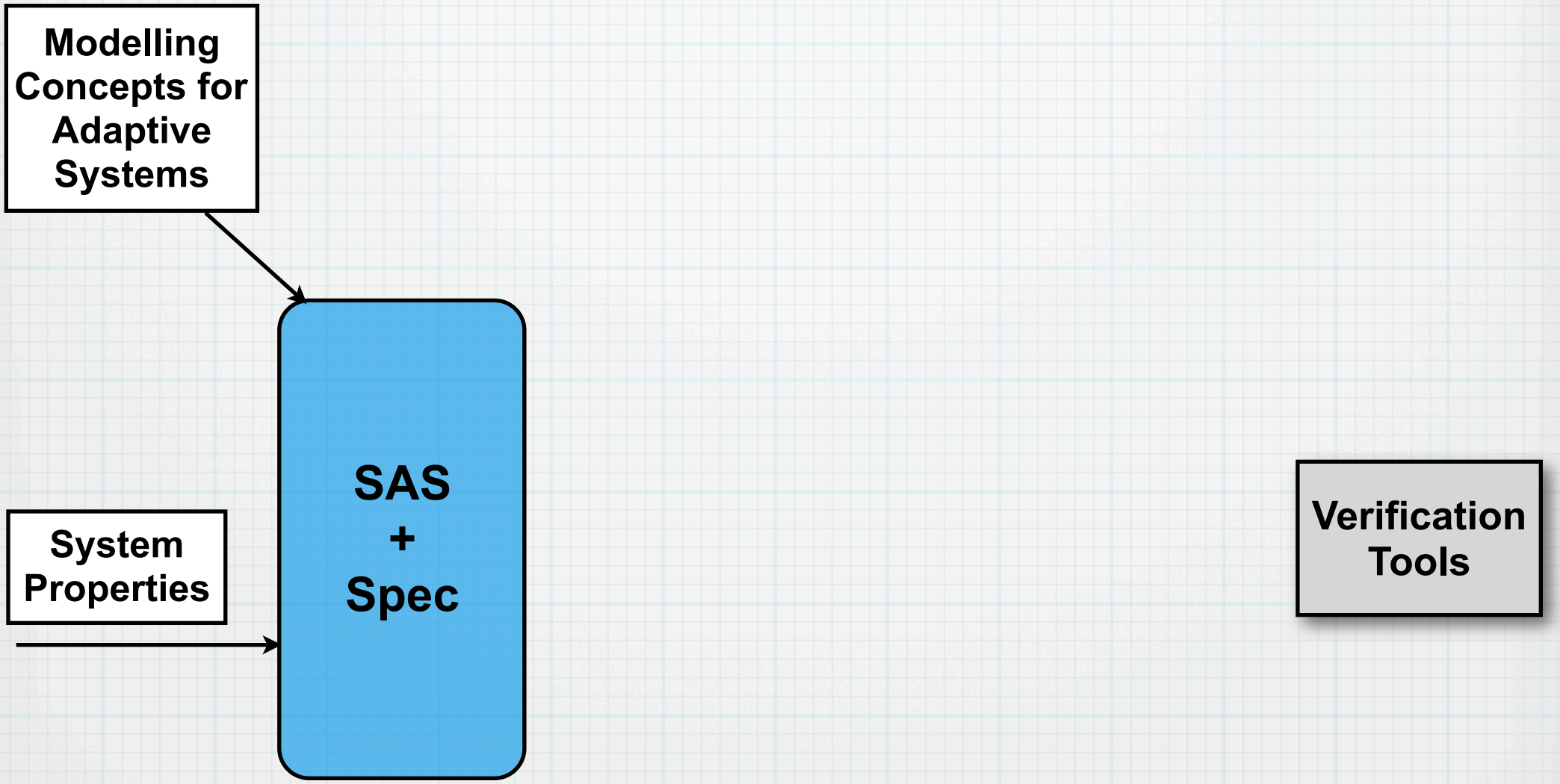
- **No configuration is always only transient:**

$$\bigwedge_{i=1}^{n} \mathsf{EFEG}\ c = \text{config}_i$$

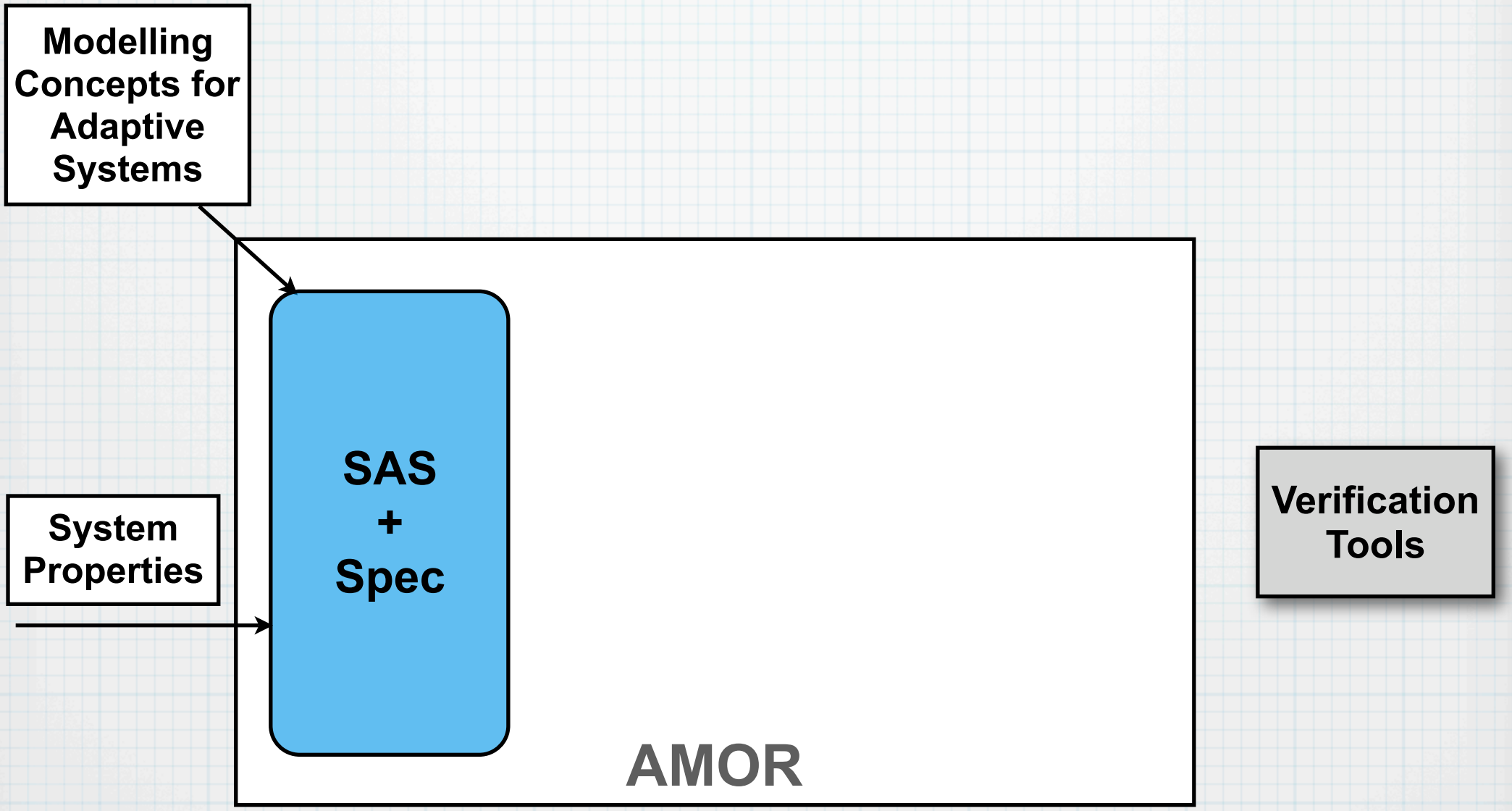
Integration Framework

Modelling Concepts for Adaptive Systems

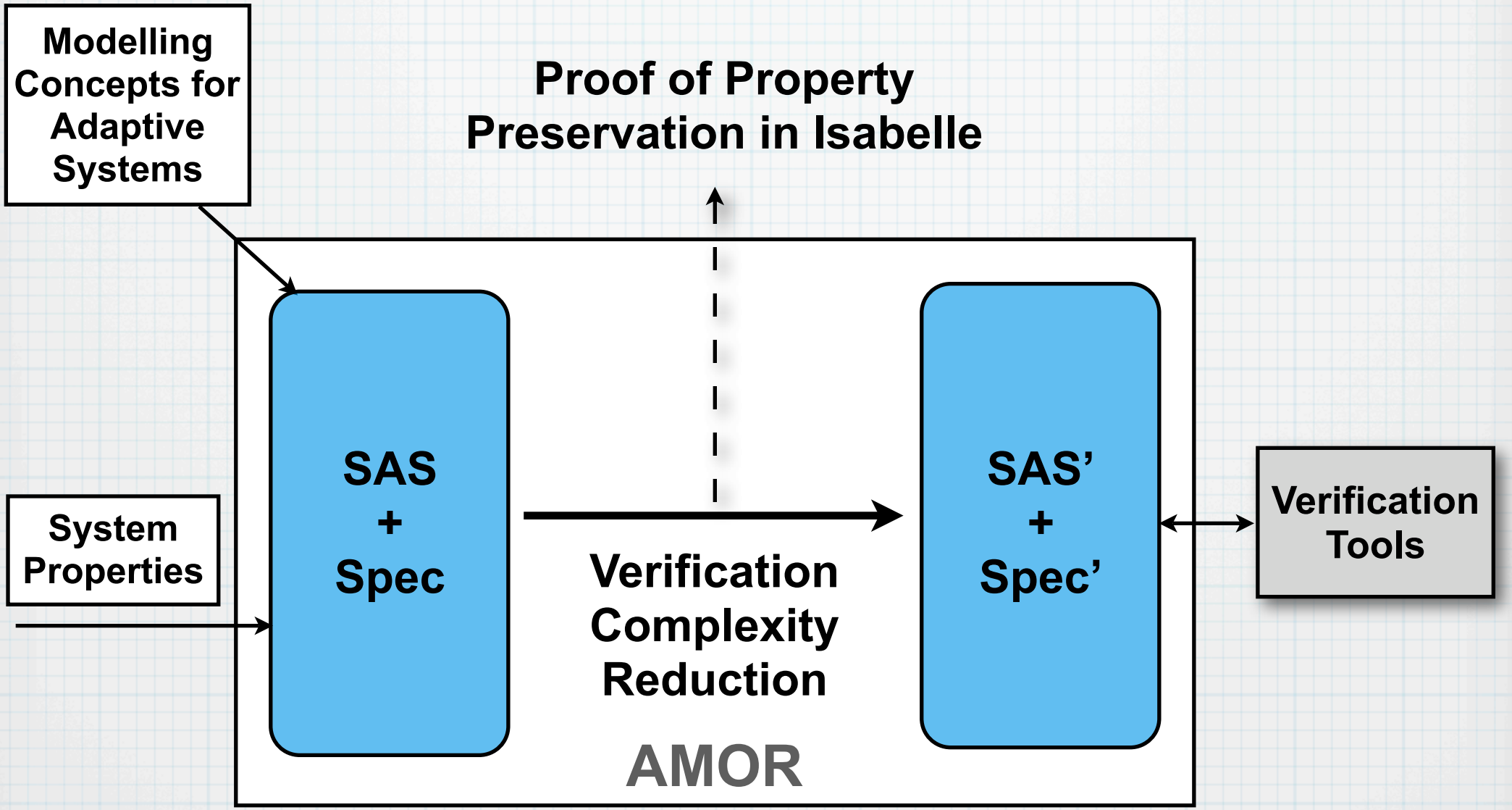
Integration Framework



Integration Framework



Integration Framework



Verification Complexity Reductions

- Reductions by Slicing
- Data Domain Abstractions
- Compositional Reasoning

Stabilisation of Adaptation

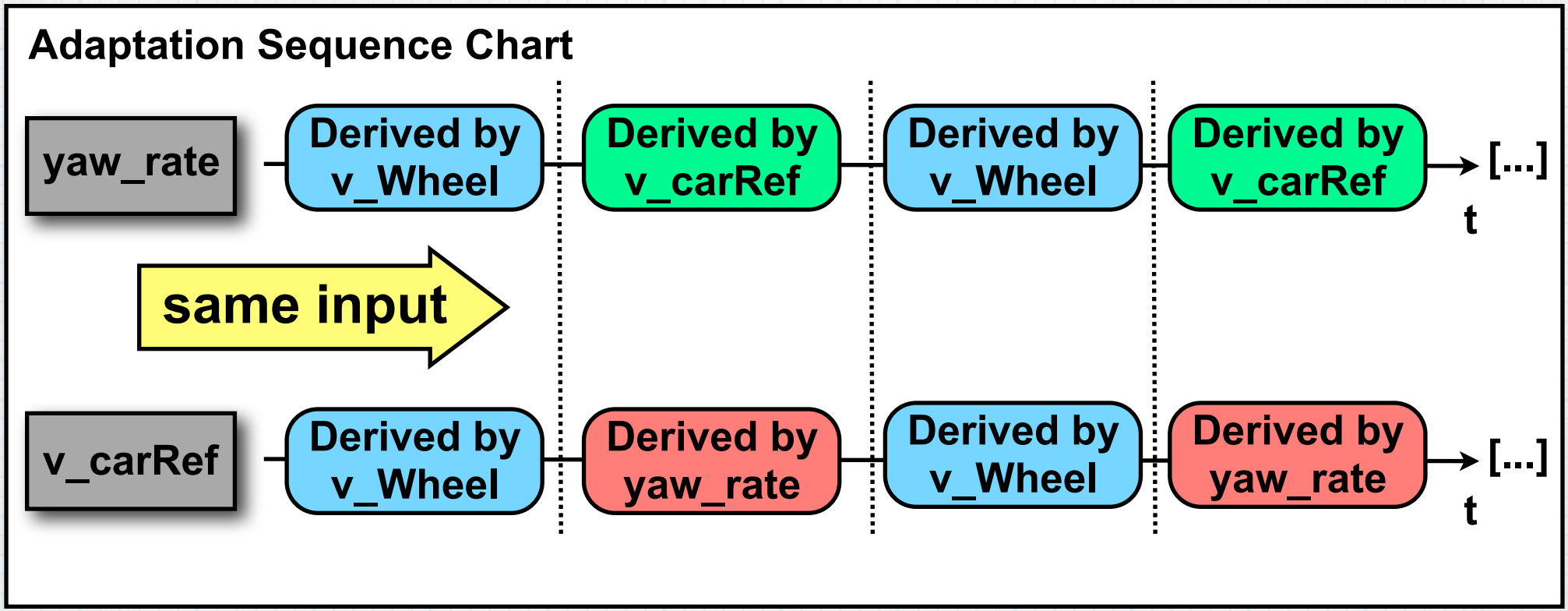
If input is unchanged, configurations will stabilise.

AG(**G** same_inputs => **F G** same_configurations)

Stabilisation of Adaptation

If input is unchanged, configurations will stabilise.

AG(**G** same_inputs => **F G** same_configurations)



Verification of Stability

AG(**G** same_inputs => **F G** same_configurations)

- **can be checked by standard LTL model checking**
- **involves two alternating fixpoints ⇒ not very efficient**

Verification of Stability

AG(**G** same_inputs => **F G** same_configurations)

- **can be checked by standard LTL model checking**
- **involves two alternating fixpoints ⇒ not very efficient**

- **Alternative Procedure:**

  Analyse two systems with one independent fixpoint each

  1. **Original System ➢ set of reachable states**
  2. **System restricted to input steady paths without self-loops ➢ set of states with an infinite path (outputs not steady)**

Verification of Stability

AG(**G** same_inputs => **F G** same_configurations)

- **can be checked by standard LTL model checking**
- **involves two alternating fixpoints ⇒ not very efficient**

- **Alternative Procedure:**

  Analyse two systems with one independent fixpoint each

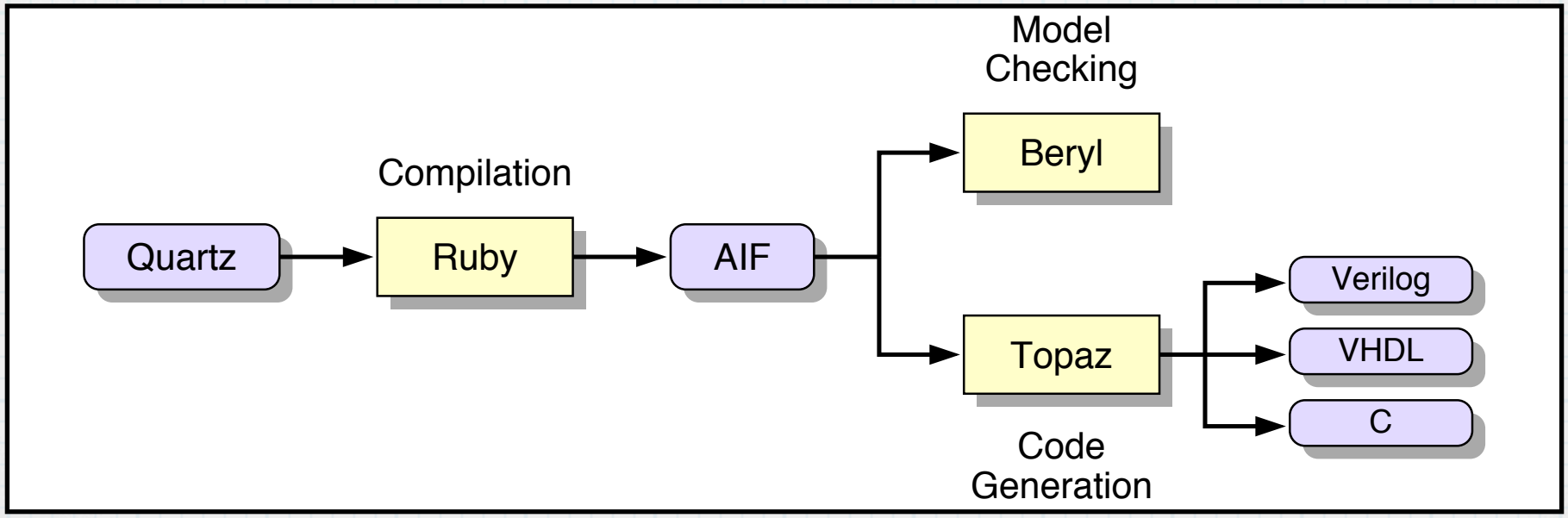
  1. **Original System ➢ set of reachable states**
  2. **System restricted to input steady paths without self-loops**

     ➢ set of states with an infinite path (outputs not steady)

Theorem: Original system is stable iff intersection is empty!

Verification with AVEREST

- Framework for the development of reactive systems
- Quartz: synchronous language for describing parallel, deterministic systems
- Process communication and synchronisation part of the language



Experimental Results

Traction Control System:

28 modules, 70 configurations, > 2000 lines of Quartz code
151 generic adaptation properties

Generic specifications of adaptation behaviour

event variables	min. [s]	avg. [s]	max. [s]	total [s]
no	< 1	5	222	782

Stability of adaptation behaviour

LTL model checking: > 1h	**Specialised procedure: ~0.48h**

Related Work

Modelling and Verification of Adaptive Systems

- [Bradbury et al.; 2004]:
  Survey on Self-Managed Software Architectures

- [Zhang, Cheng; 2005/06]:
  Modelling and Verification of Adaptation Behaviour based on Petri Nets

- [Schneider, Schüle, Trapp; 2006]:
  Direct Translation of Modelling Concepts to Verification Tools

Conclusion

Main Results:

- **Modelling Concepts for Adaptive Systems**
- **Framework for Integration of Model-based Design and Formal Verification using with High-Level Formal Model and Automatic Transformations for Complexity Reduction**
- **Standard and Specialised Verification Techniques for Adaptive Systems**

Future Work

- Extension of Modelling Concepts with Intuitive Property Specification at Modelling Level
- Further Development of Reduction and Verification Techniques
- Propagation of Verification Results to Modelling Level