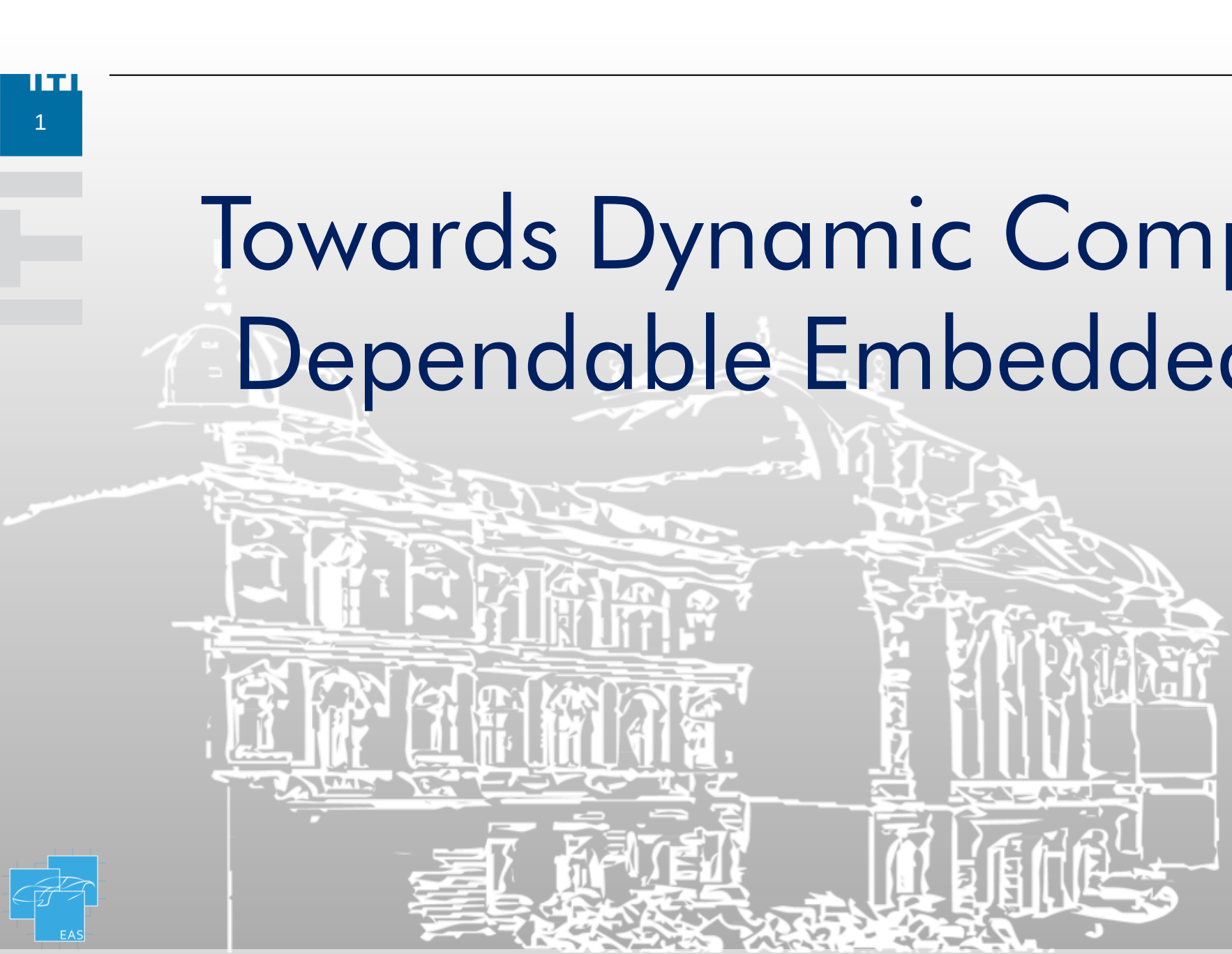
Towards Dynamic Composition of Dependable Embedded Systems



Leandro Batista Ribeiro
lbatistaribeiro@tugraz.at

Institute of Technical Informatics
Embedded Automotive Systems Group
Graz University of Technology

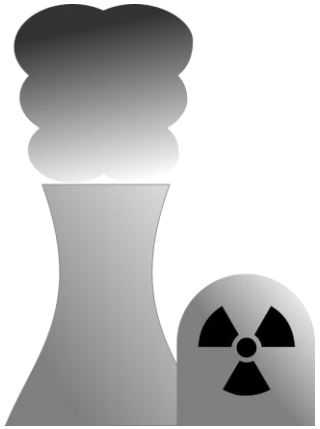
Towards Dynamic Composition of Dependable Embedded Systems

Towards Dynamic Composition of Dependable Embedded Systems

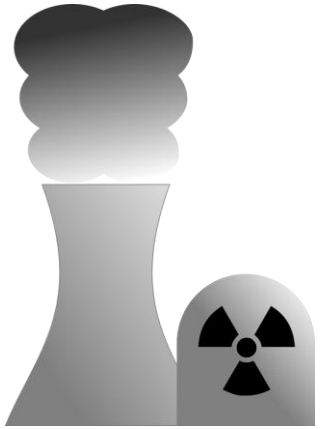
Towards Dynamic Composition of Dependable Embedded Systems

Towards Dynamic Composition of Dependable Embedded Systems

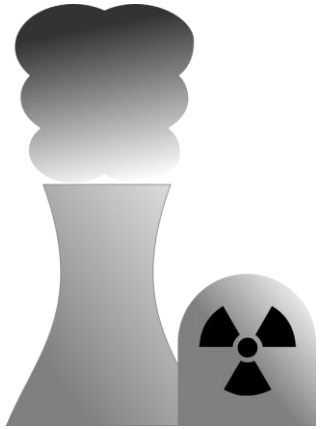
Towards Dynamic Composition of Dependable Embedded Systems



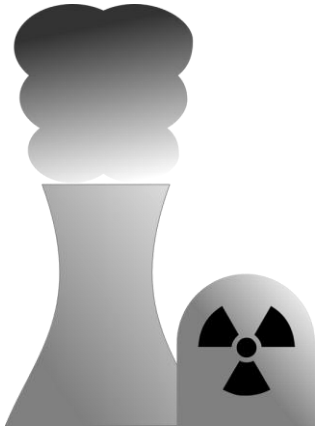
Towards Dynamic Composition of Dependable Embedded Systems



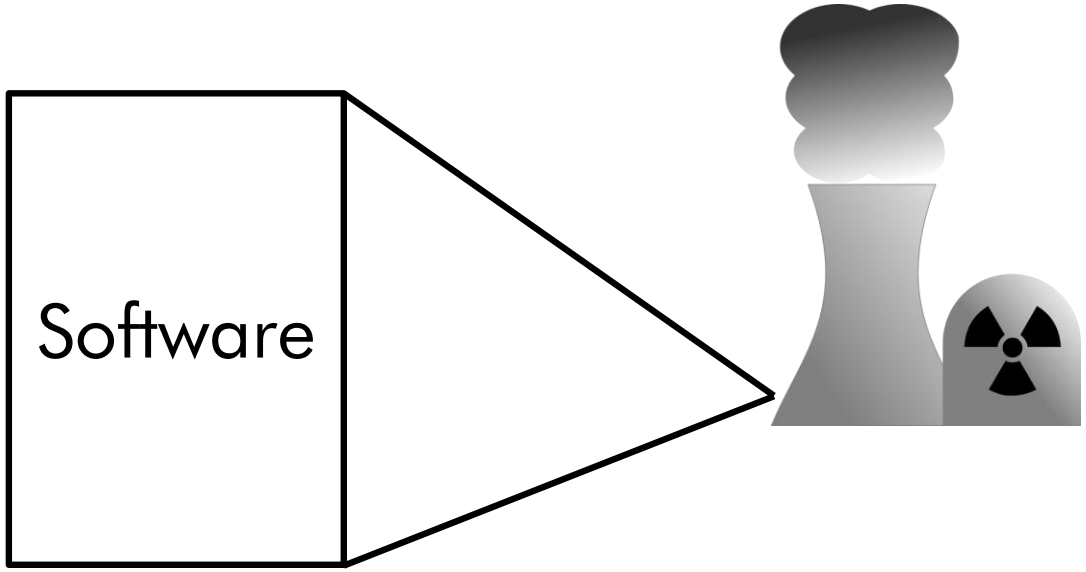
Towards Dynamic Composition of Dependable Embedded Systems



Towards Dynamic Composition of Dependable Embedded Systems

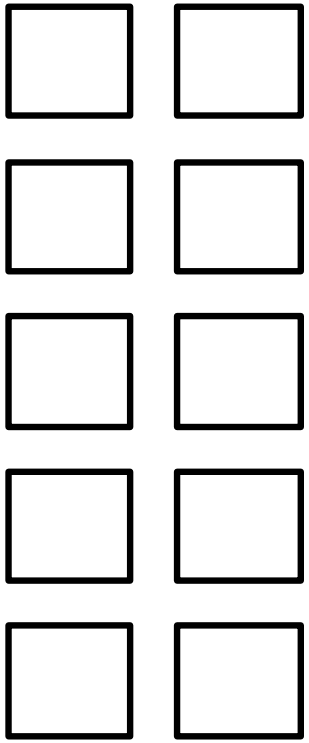


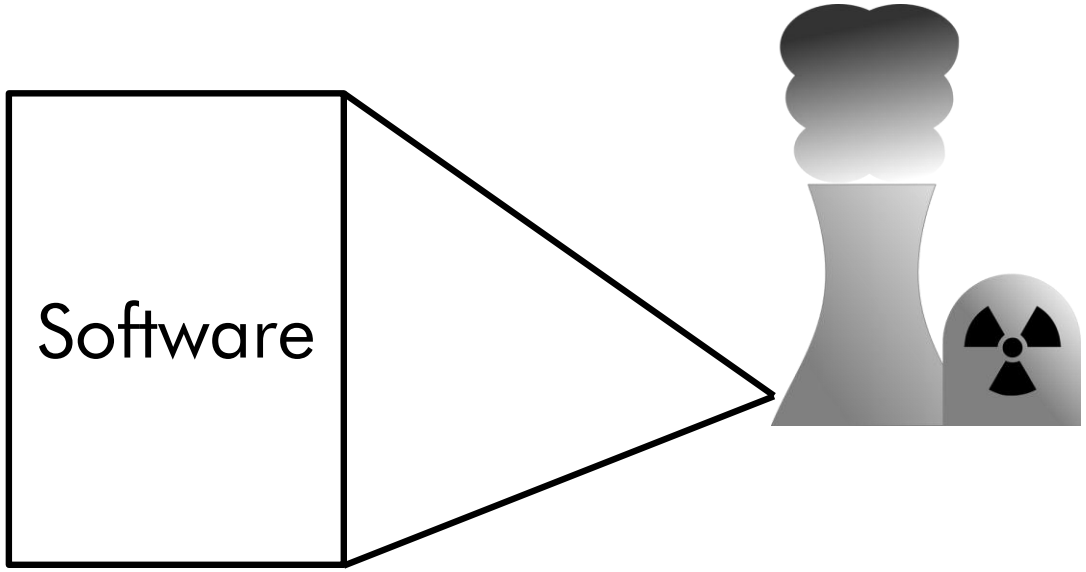
Towards Dynamic Composition of Dependable Embedded Systems



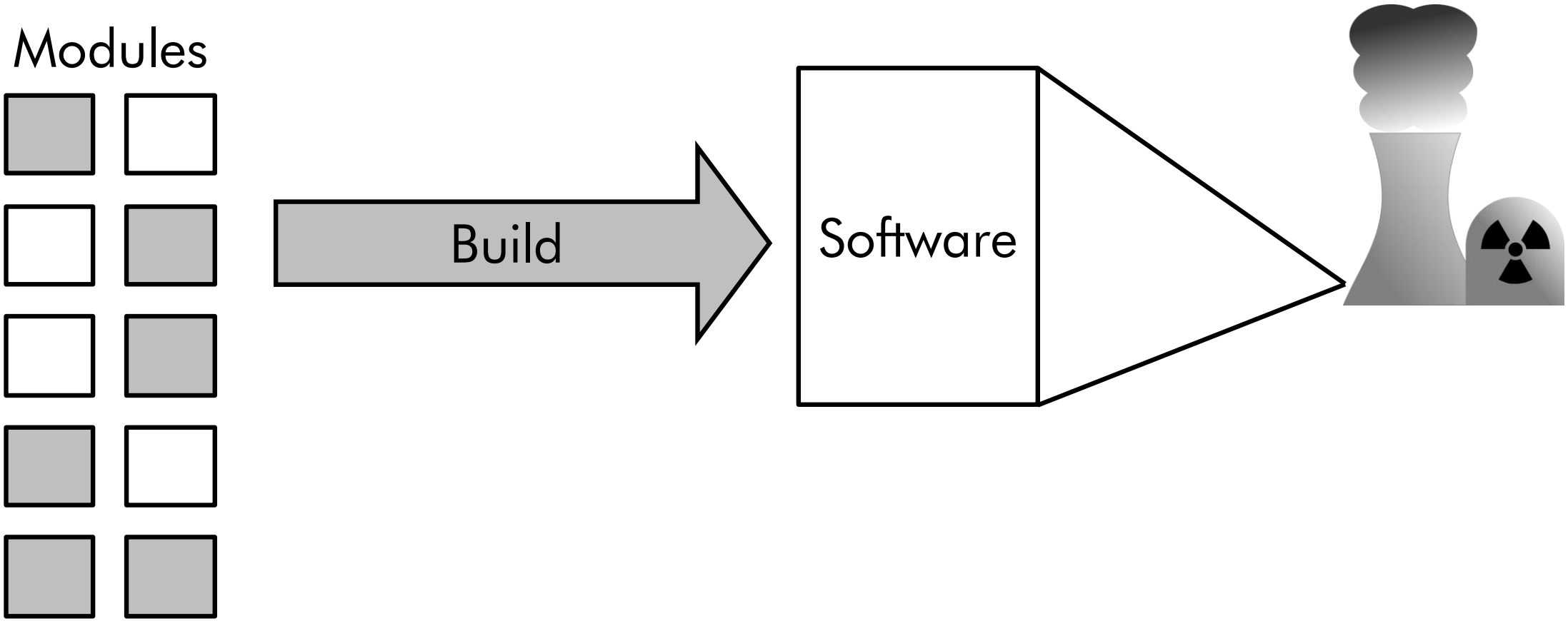
Towards Dynamic Composition of Dependable Embedded Systems

Modules

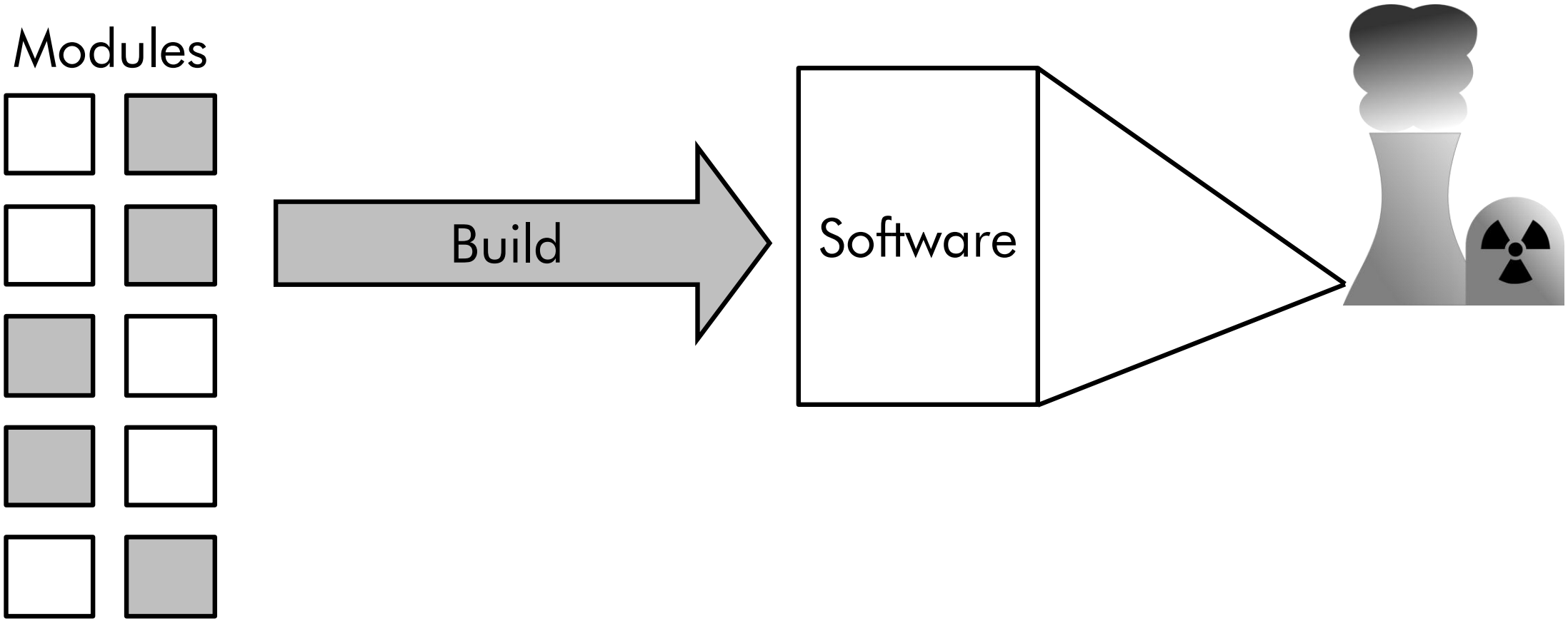




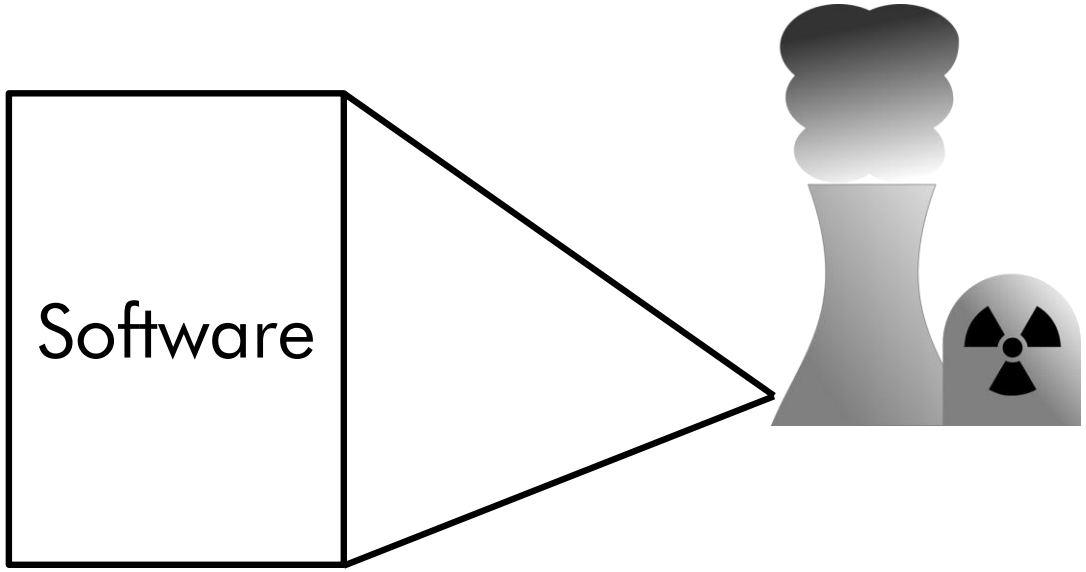
Towards Dynamic Composition of Dependable Embedded Systems



Towards Dynamic Composition of Dependable Embedded Systems

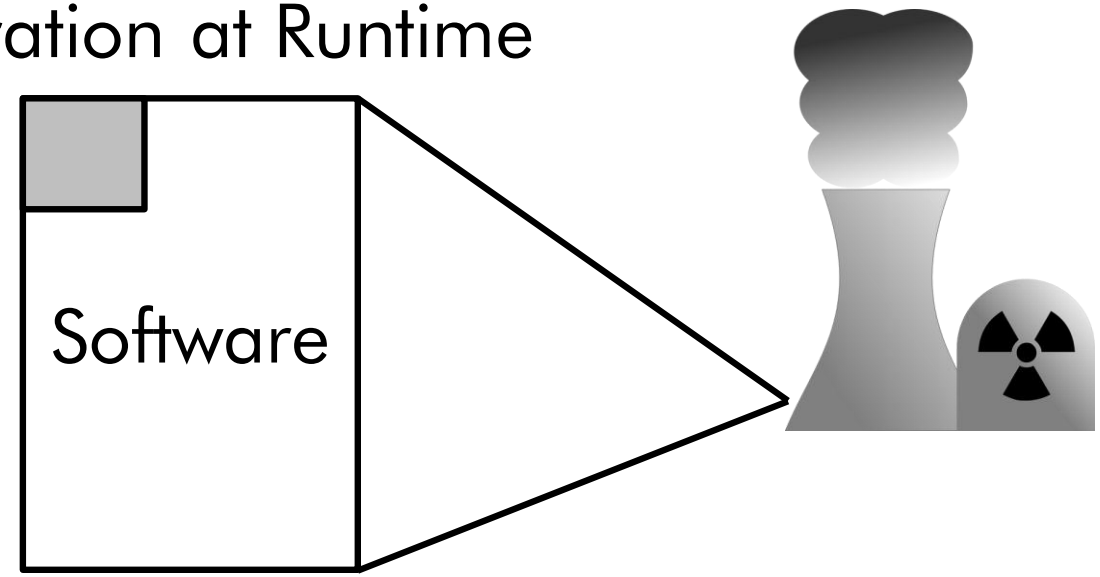


Towards Dynamic Composition of Dependable Embedded Systems

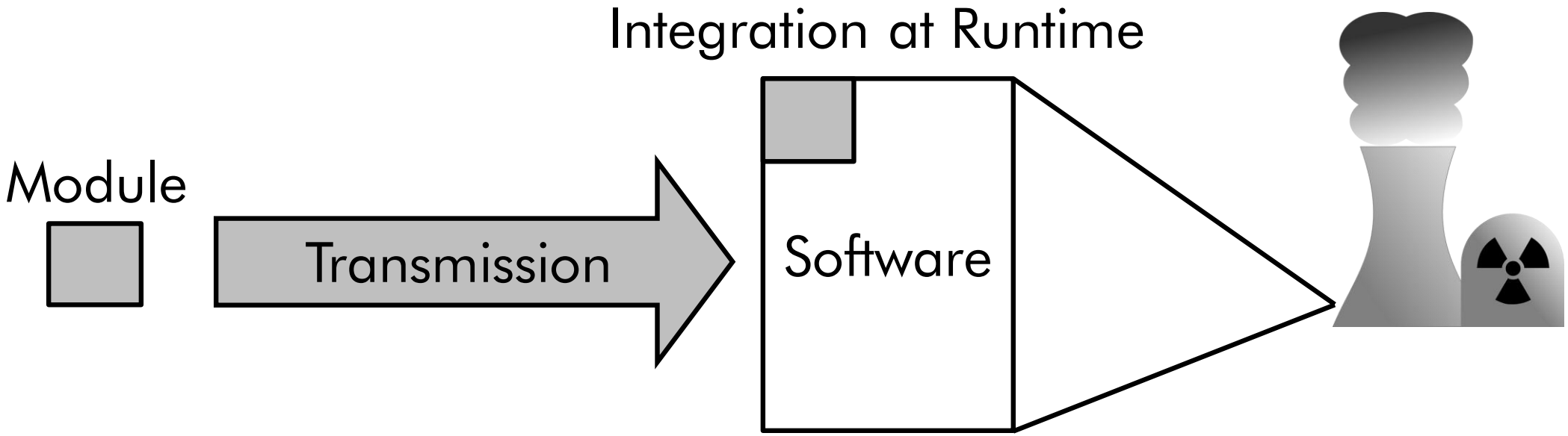


Towards Dynamic Composition of Dependable Embedded Systems

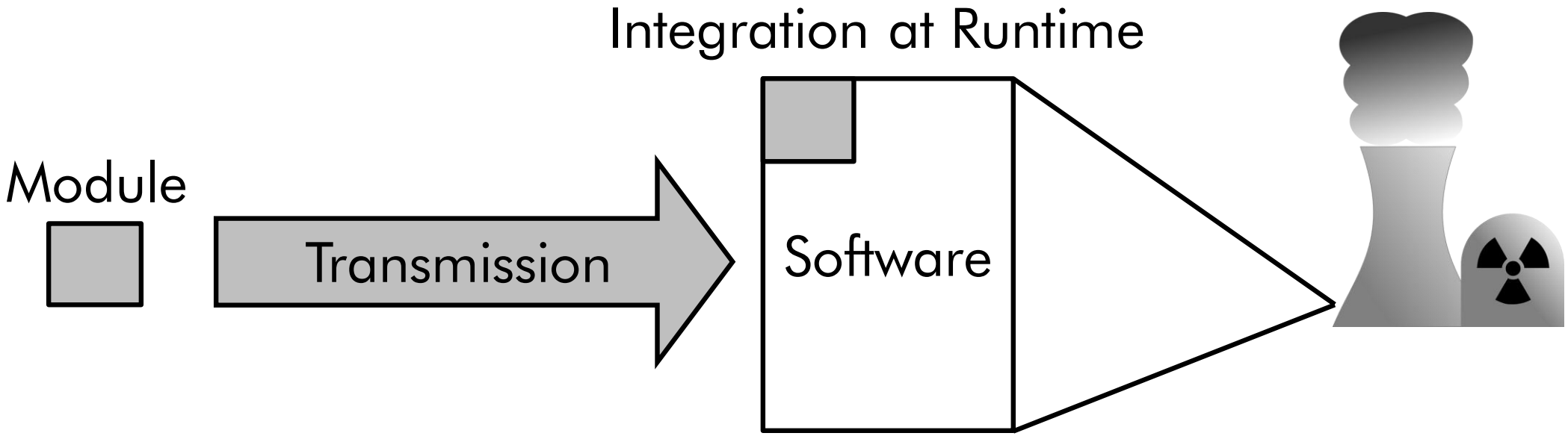
Integration at Runtime



Towards Dynamic Composition of Dependable Embedded Systems



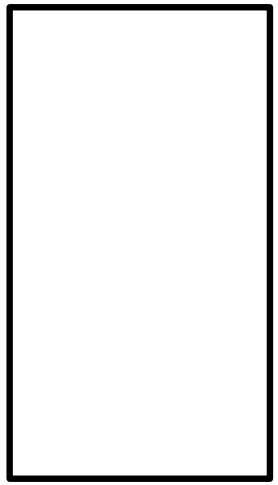
Towards Dynamic Composition of Dependable Embedded Systems



Inherent Problems

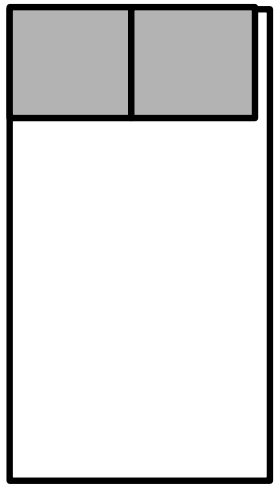
Inherent Problems

Memory



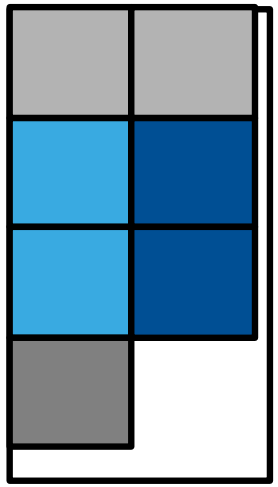
Inherent Problems

Memory



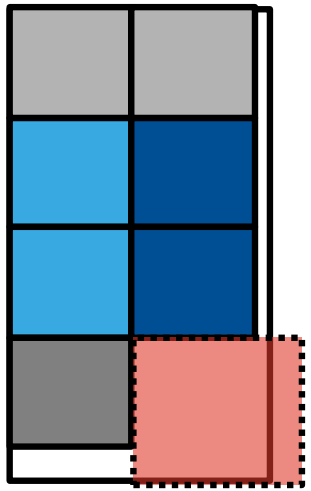
Inherent Problems

Memory



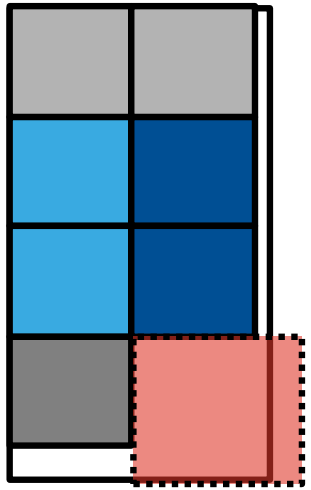
Inherent Problems

Memory



Inherent Problems

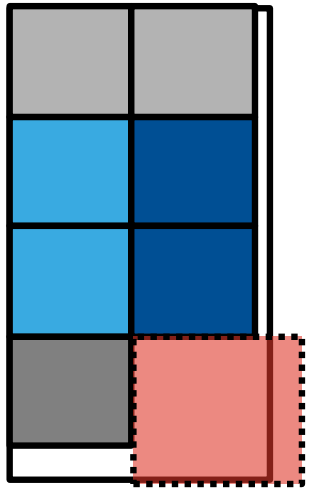
Memory



Dependencies

Inherent Problems

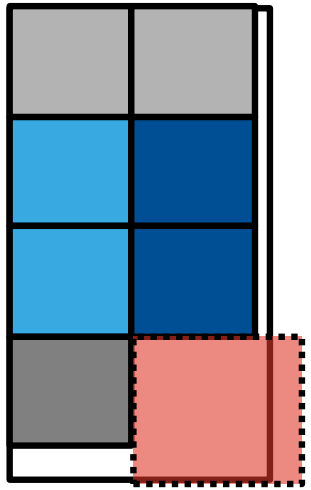
Memory



Dependencies

Inherent Problems

Memory

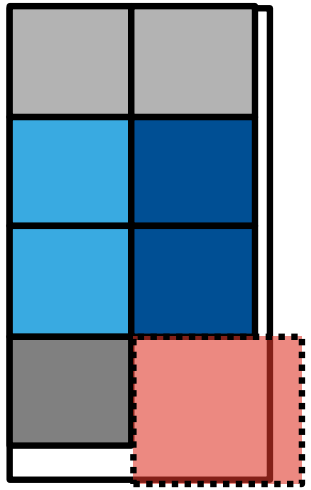


Dependencies

Behavior

Inherent Problems

Memory



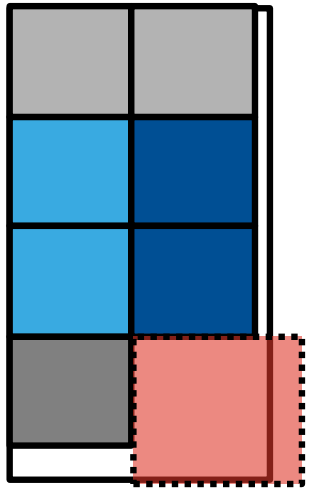
Dependencies

Behavior

Functional Requirements

Inherent Problems

Memory



Dependencies

Behavior

(Non-)Functional Requirements

Research Question

How to partially update embedded systems and guarantee the correctness of the resulting software composition?

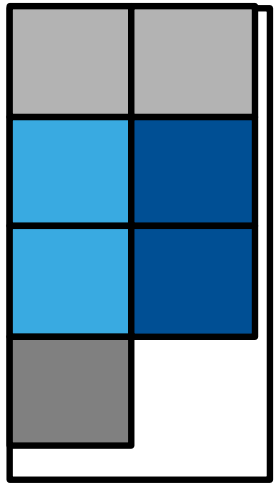
Research Question

How to partially update embedded systems and guarantee the correctness of the resulting software composition?

Partial Updates

SmartOS and Update Protocol

Memory



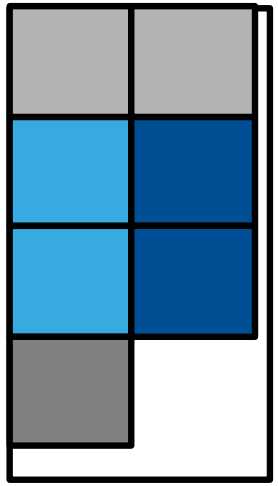
Dependencies

Partial Updates

SmartOS and Update Protocol

EWSN'20

Memory



Dependencies

SmartOS

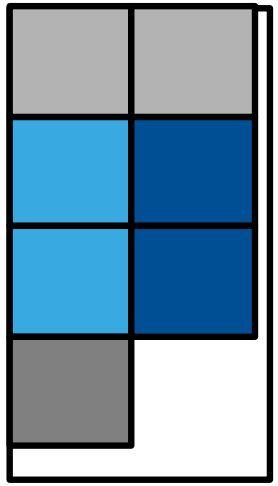
Sustainable modular adaptive real-time Operating System

Partial Updates

SmartOS and Update Protocol

EWSN'20

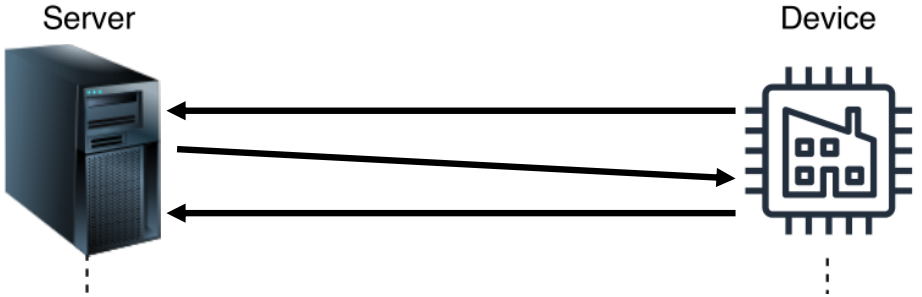
Memory



Dependencies

SmartOS

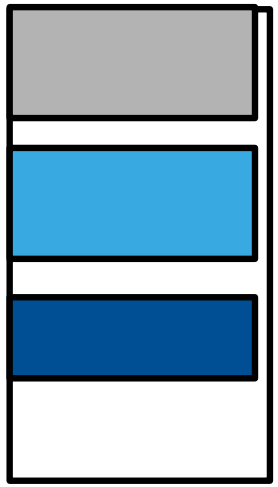
Sustainable modular adaptive real-time Operating System



Partial Updates

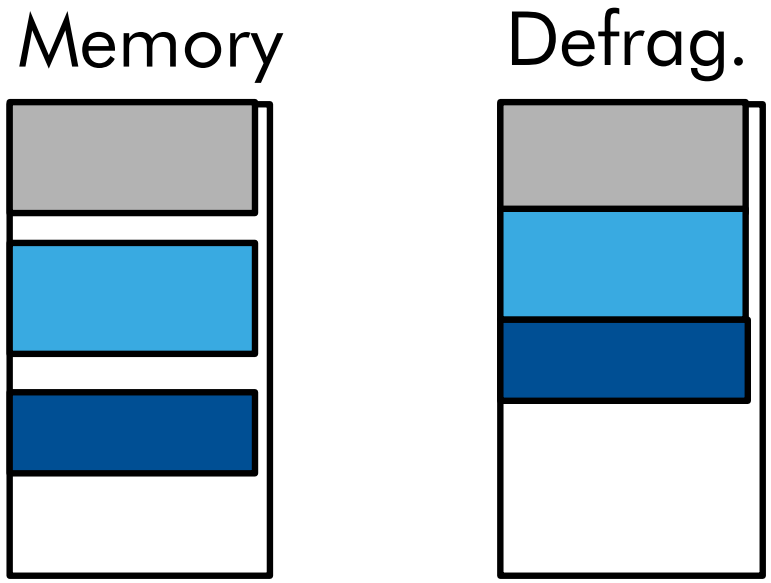
Maintainability

Memory



Partial Updates

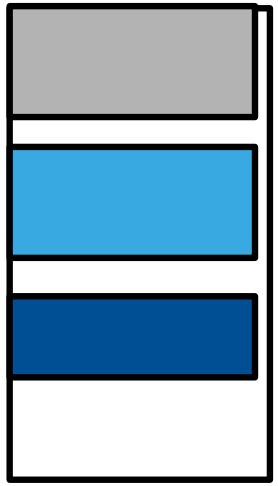
Maintainability



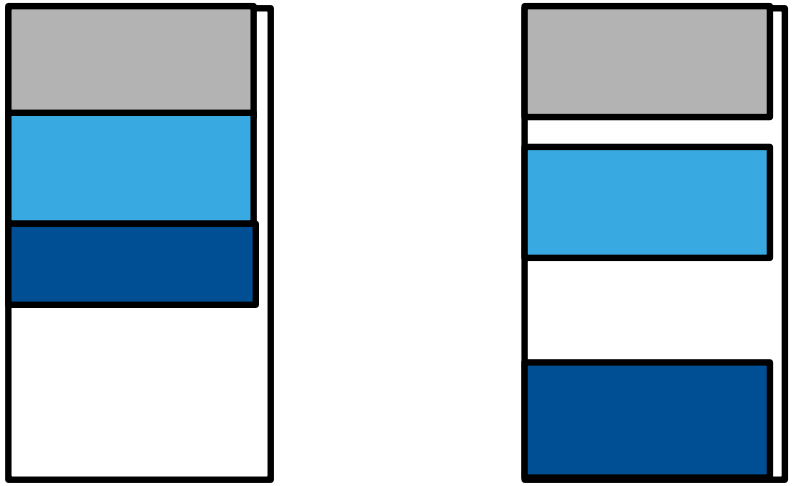
Partial Updates

Maintainability

Memory

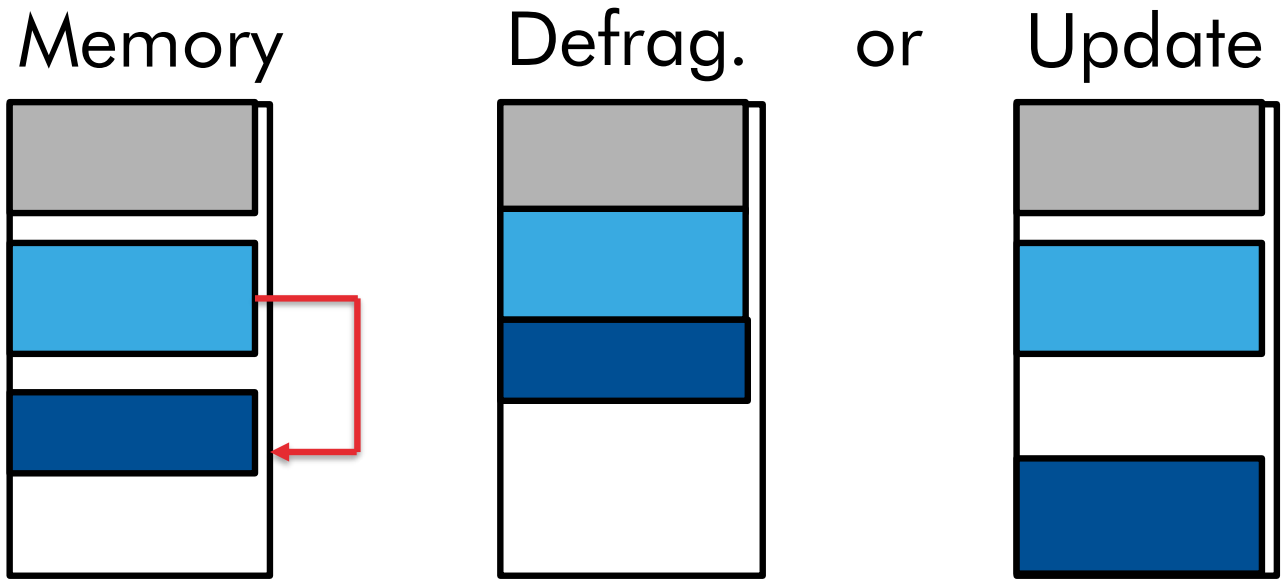


Defrag. or Update



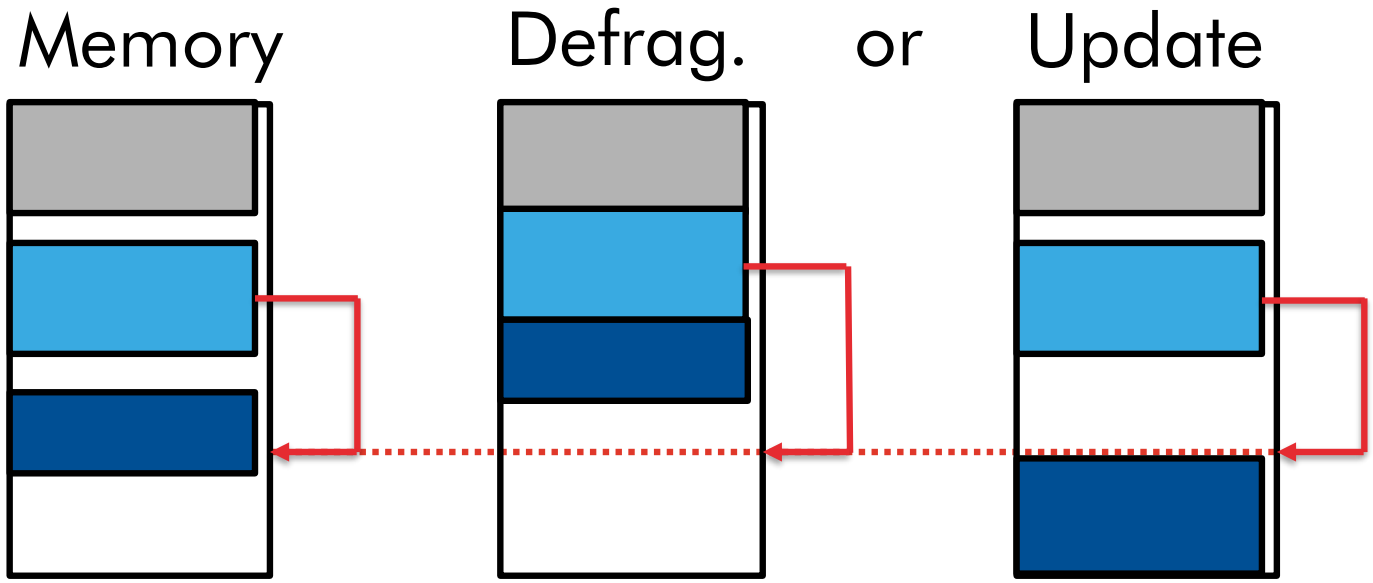
Partial Updates

Maintainability



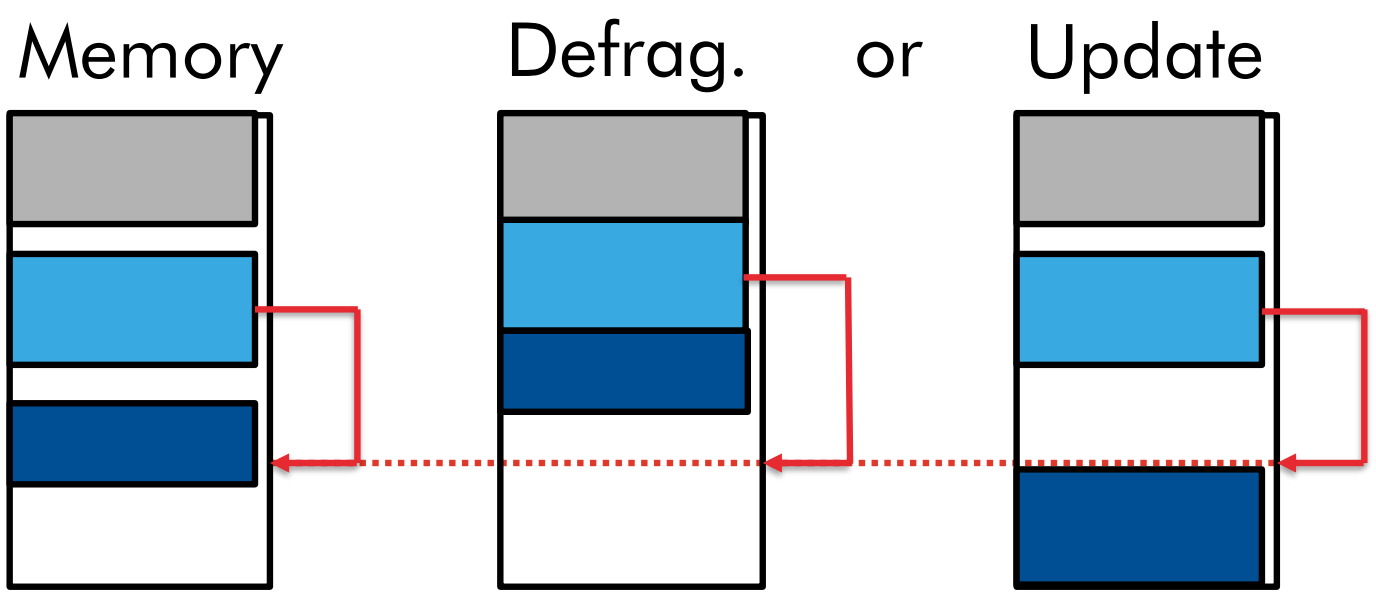
Partial Updates

Maintainability



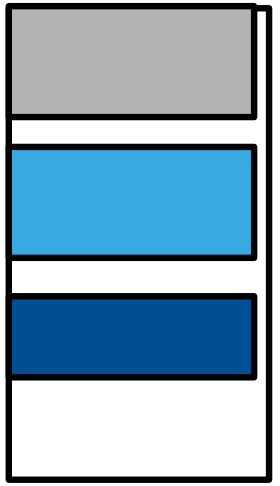
Partial Updates

Maintainability



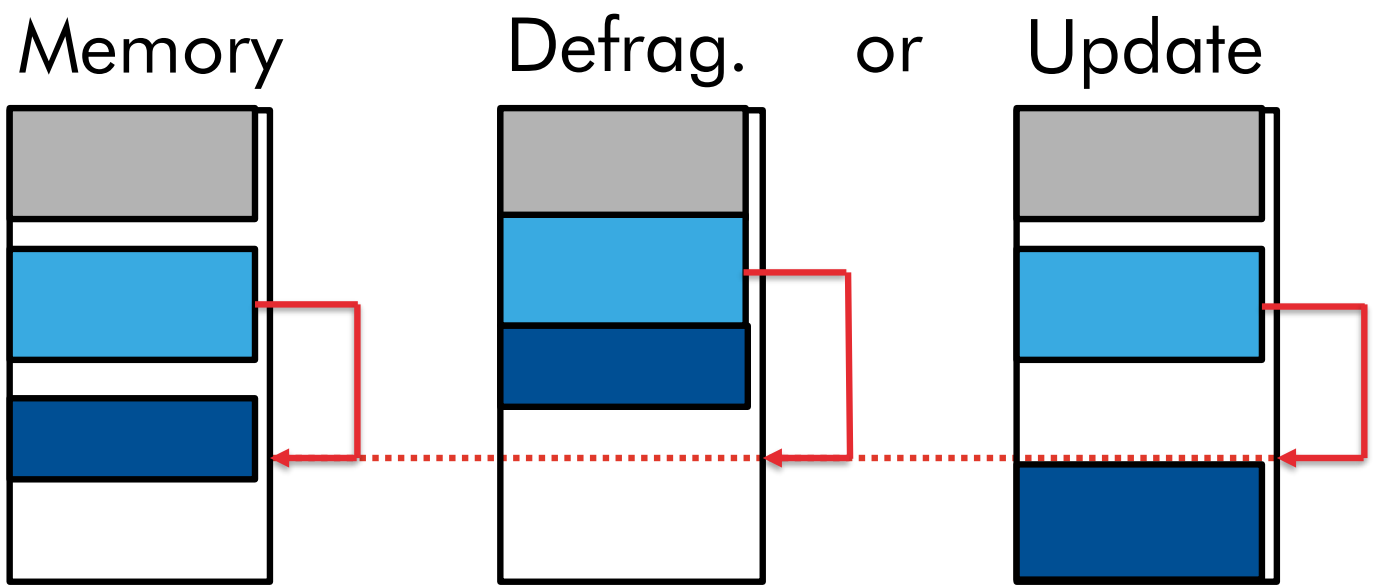
CODES'21

HW Support



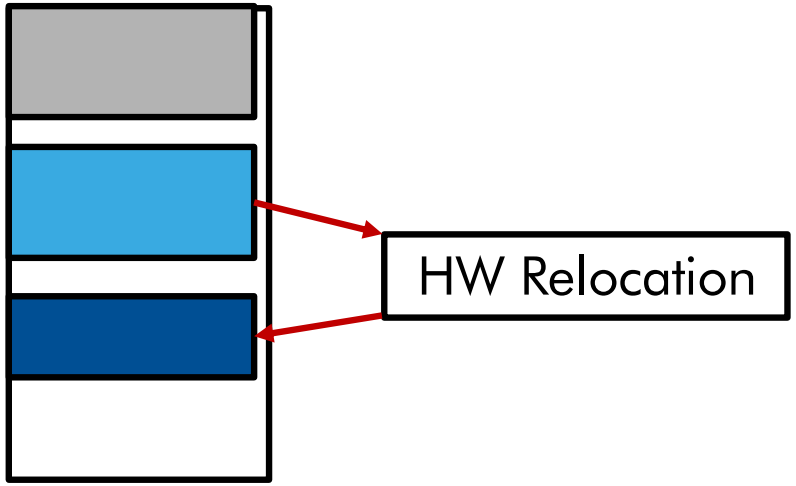
Partial Updates

Maintainability



CODES'21

HW Support



Correctness

Our Focus

Behavior

(Non-)Functional Requirements

Correctness

Our Focus

- Real-Time

Behavior

(Non-)Functional Requirements

Correctness

Our Focus

- Real-Time
- Liveness

Behavior

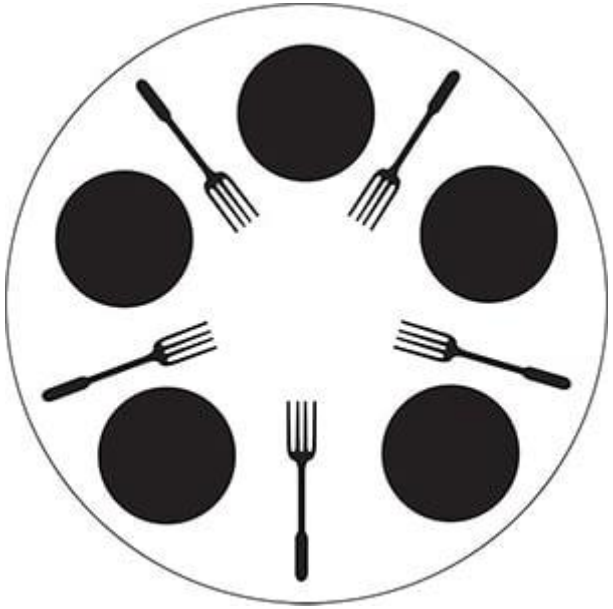
(Non-)Functional Requirements

Correctness

Our Focus

- Real-Time
- Liveness

Dining Philosophers



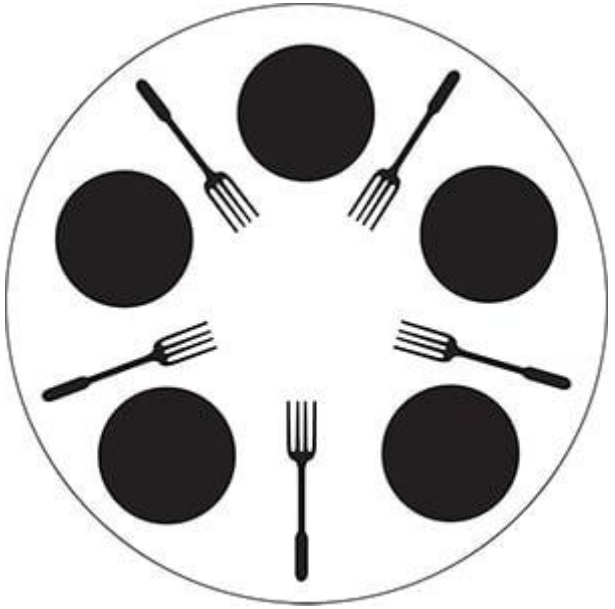
[DP]

Correctness

Our Focus

- Real-Time
- Liveness
  - Freedom of Starvation

Dining Philosophers



[DP]

Correctness

Our Focus

- Real-Time
- Liveness
  - Freedom of Starvation

Dining Philosophers

[DP]

Correctness

Our Focus

- Real-Time
- Liveness
  - Freedom of Starvation

Dining Philosophers

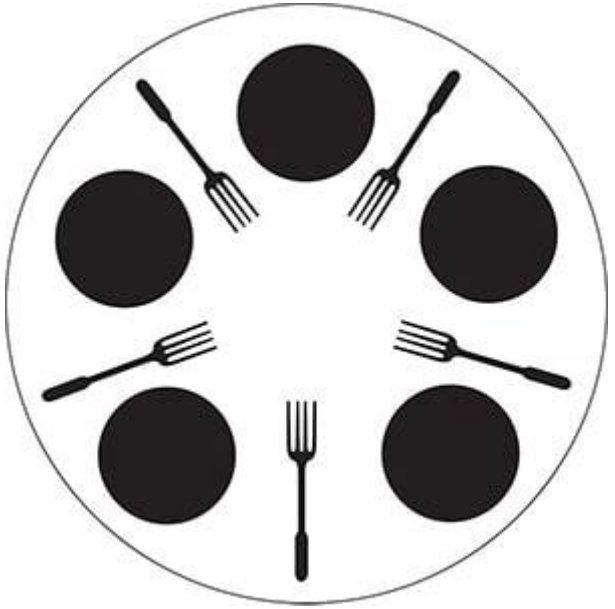
[DP]

Correctness

Our Focus

- Real-Time
- Liveness
  - Freedom of Starvation

Dining Philosophers



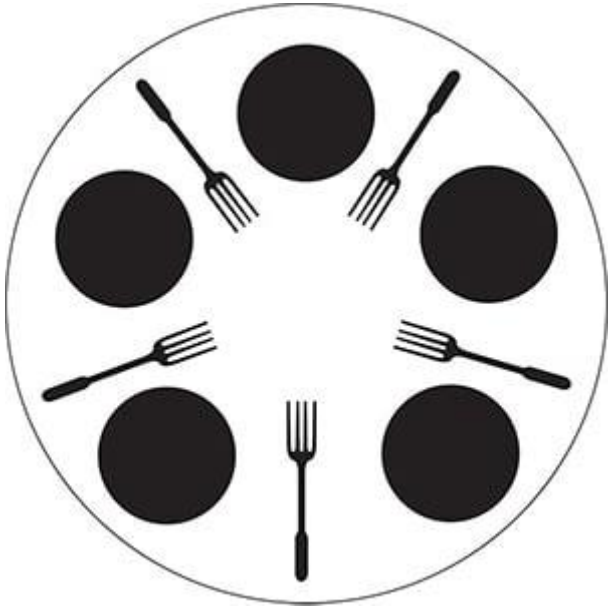
[DP]

Correctness

Our Focus

- Real-Time
- Liveness
  - Freedom of Starvation
  - Freedom of Deadlock

Dining Philosophers



[DP]

Correctness

Our Focus

- Real-Time
- Liveness
  - Freedom of Starvation
  - Freedom of Deadlock

Dining Philosophers

[DP]

Correctness

Formal Methods

- Model and Verify Software Behavior

Correctness

Formal Methods

- Model and Verify Software Behavior

Correctness

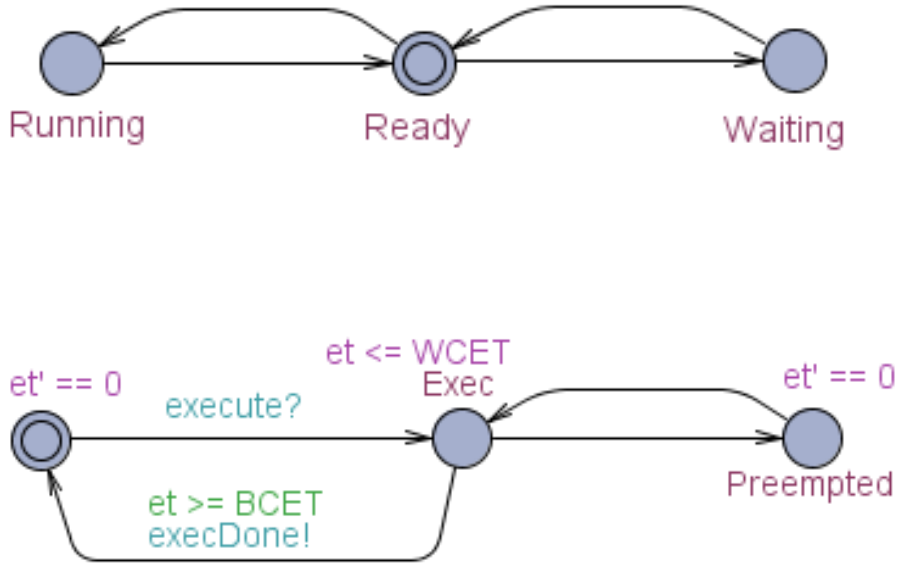
Formal Methods

- Model and Verify Software Behavior

Correctness

Formal Methods

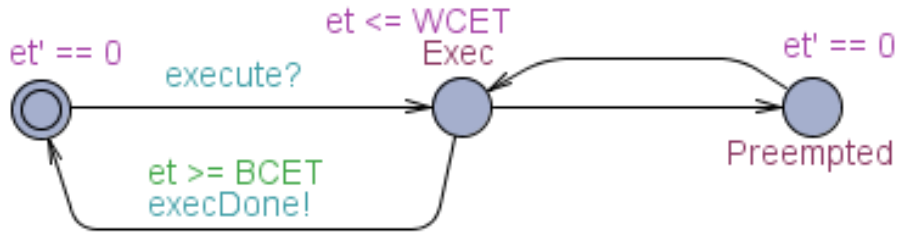
- Model and Verify Software Behavior



Correctness

Formal Methods

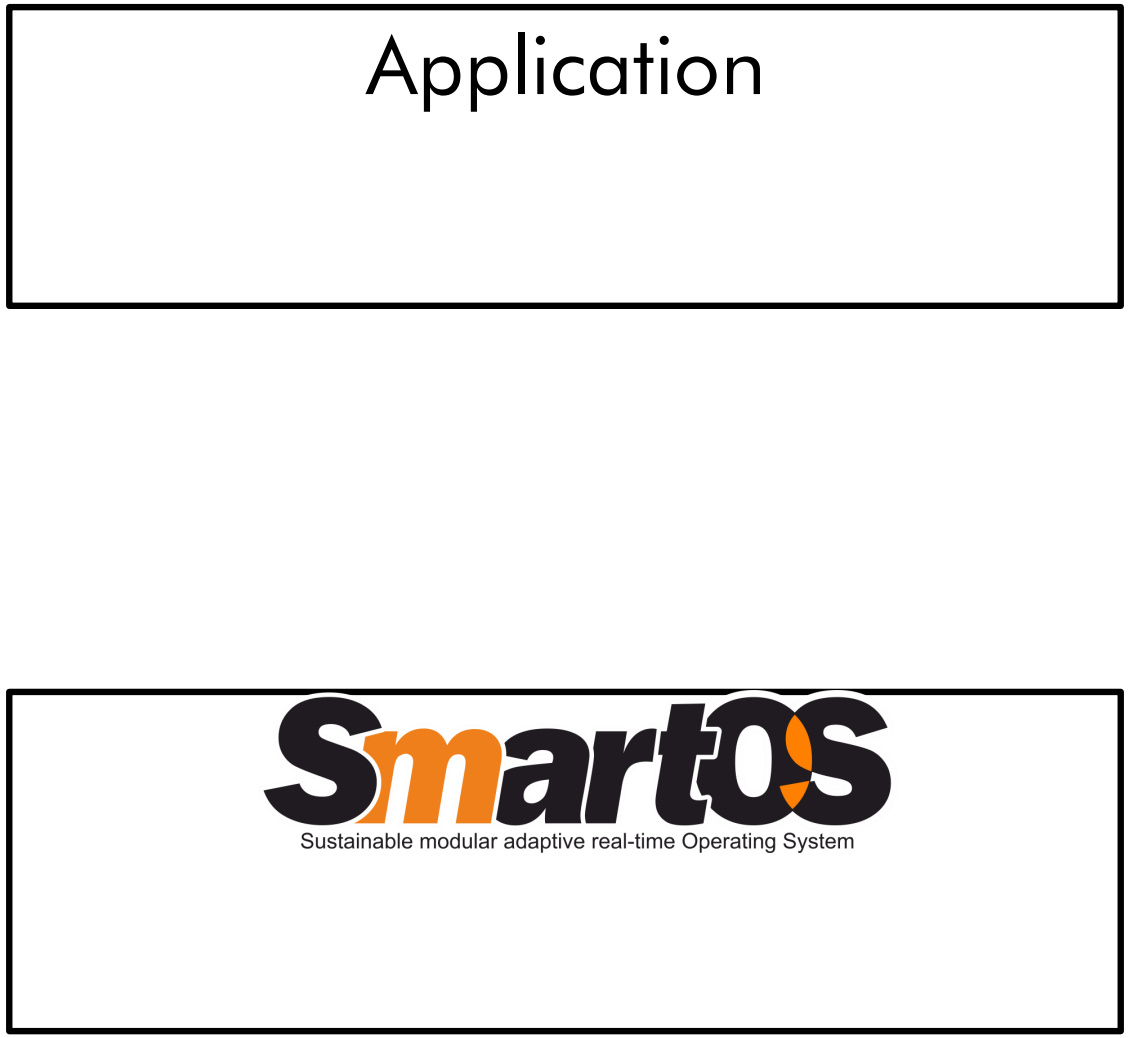
- Model and Verify Software Behavior



- Examples of Requirements
  - A deadlock must never occur
  - A given error location must never be reached
  - Internal lists must be sorted at all times

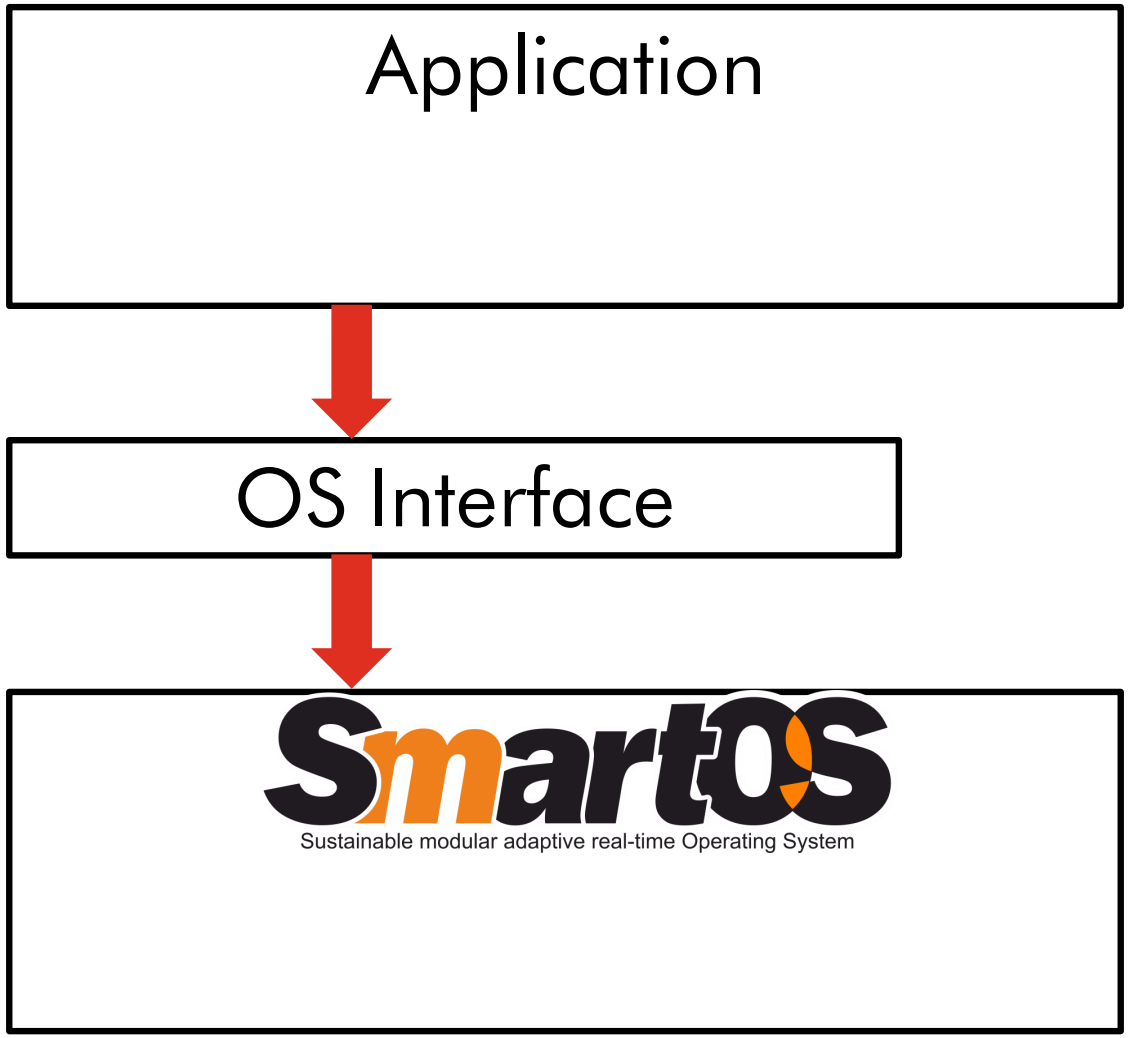
Correctness

Layered Compositional Model



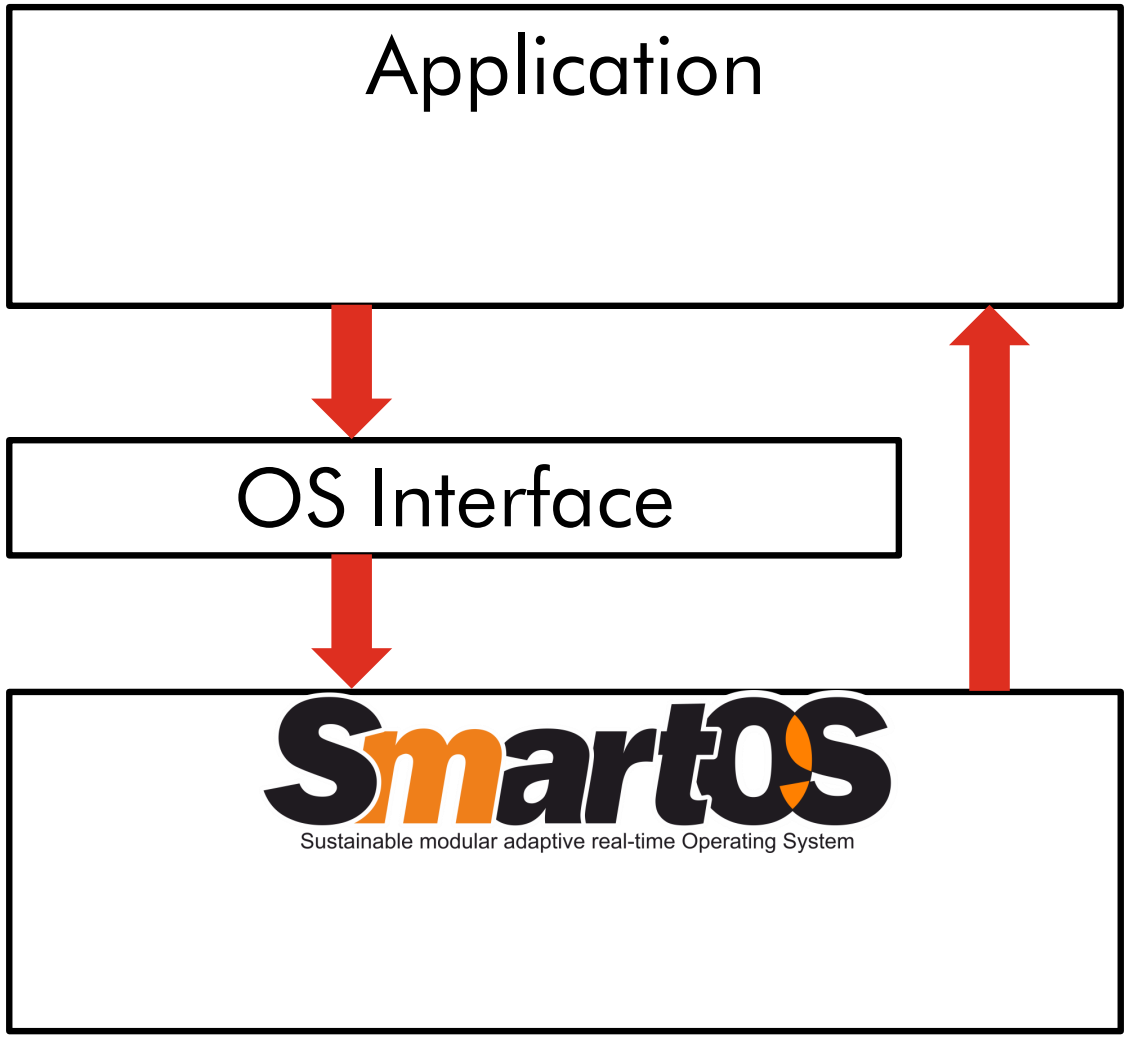
Correctness

Layered Compositional Model



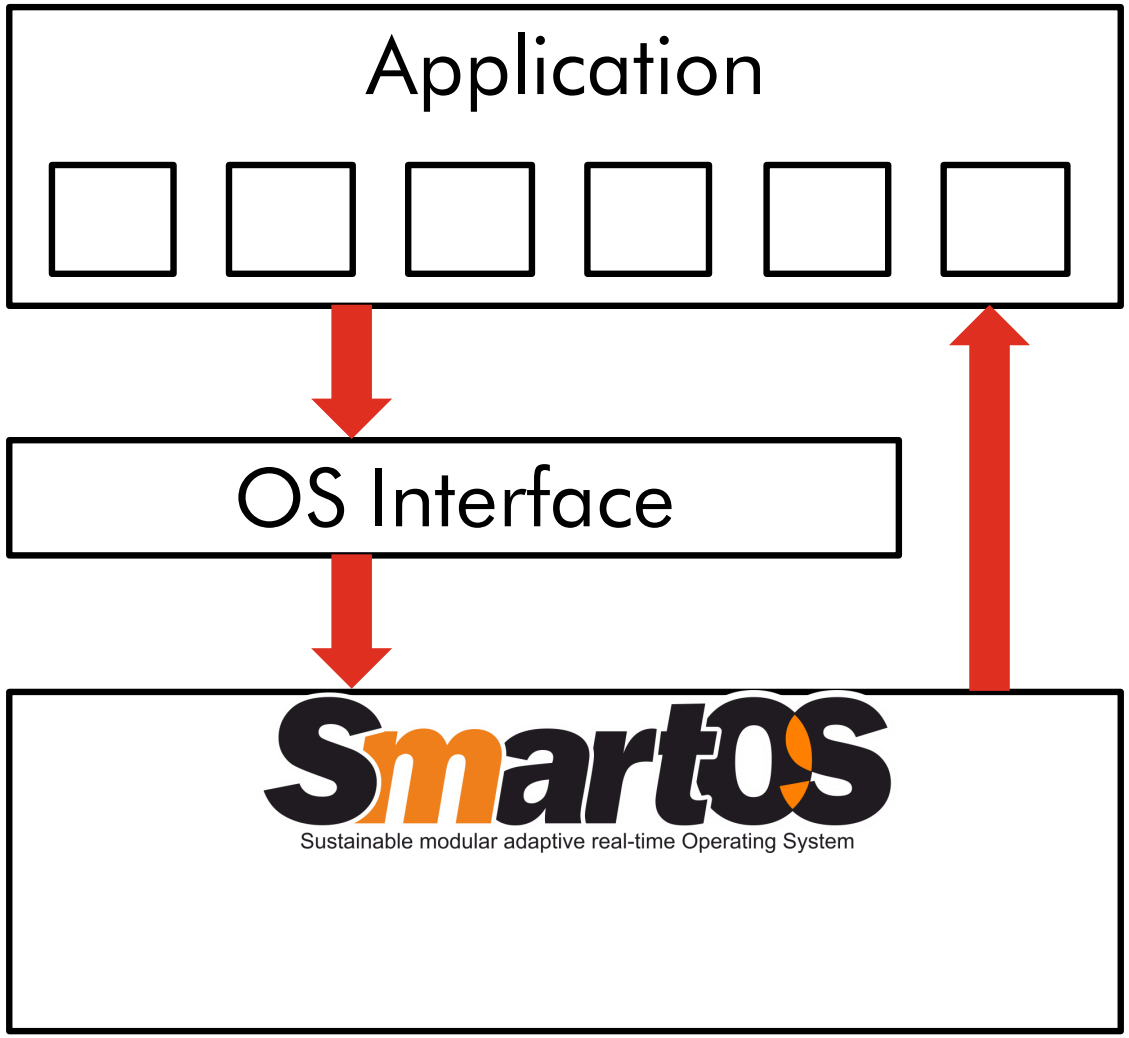
Correctness

Layered Compositional Model



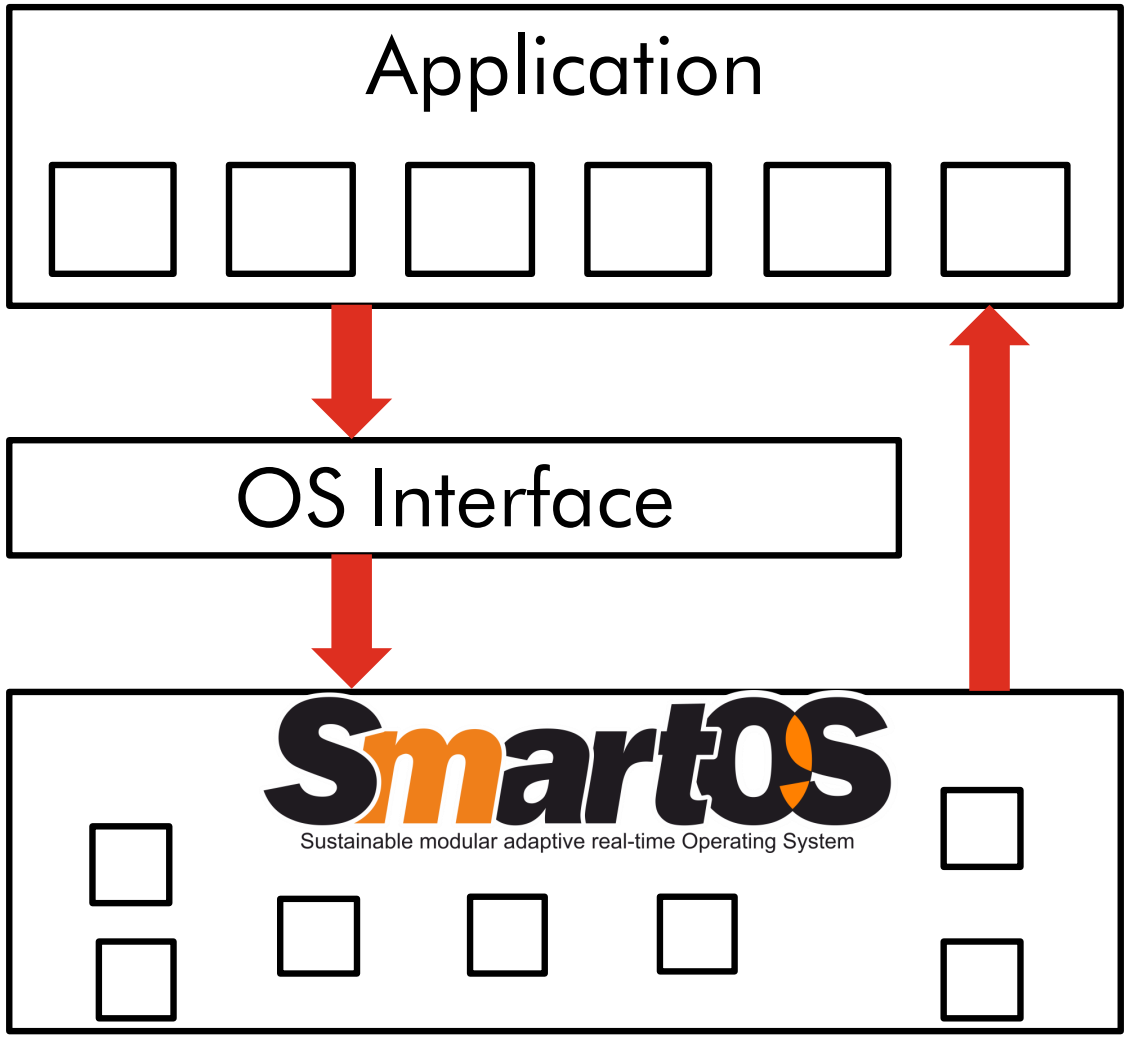
Correctness

Layered Compositional Model



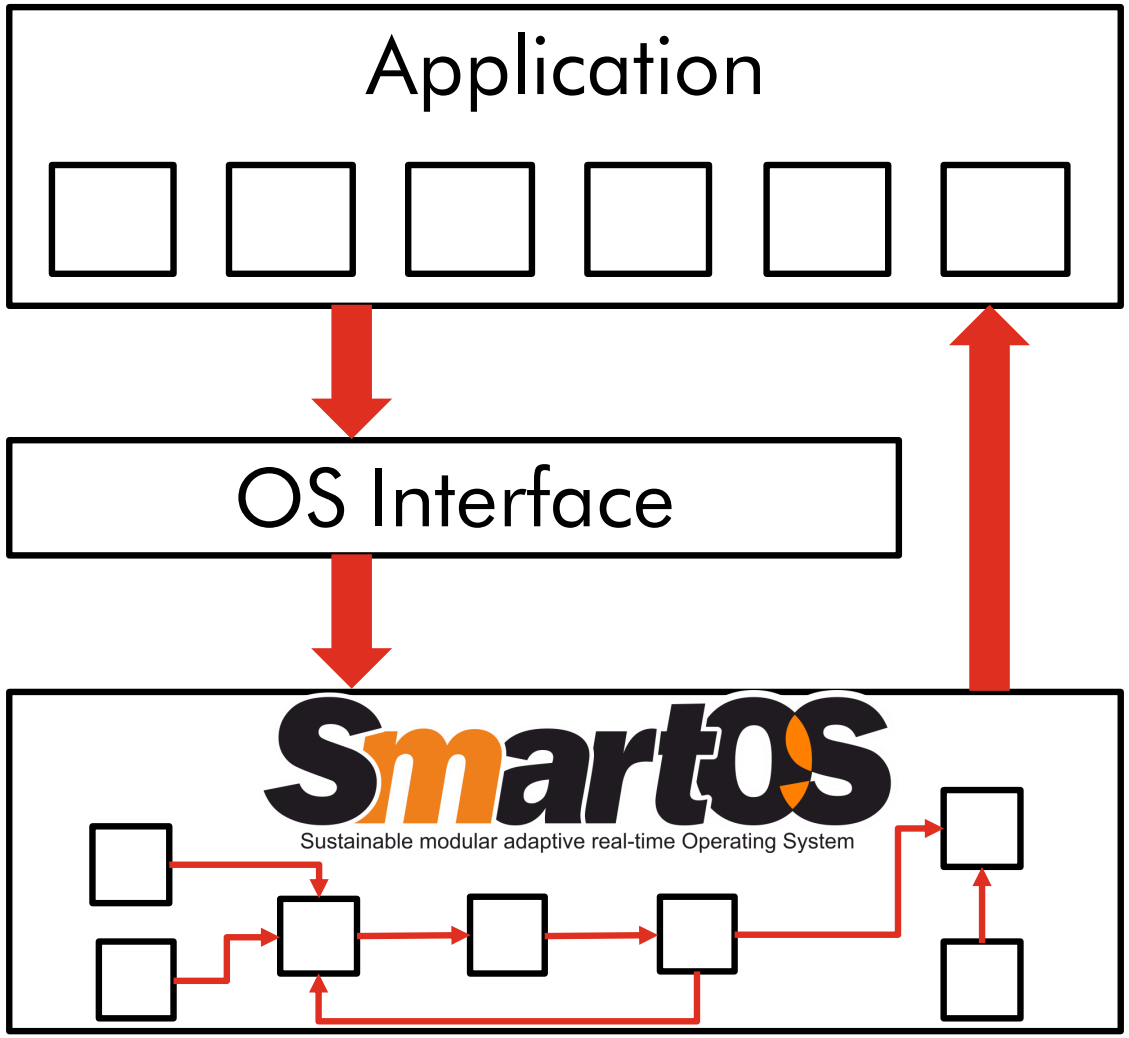
Correctness

Layered Compositional Model



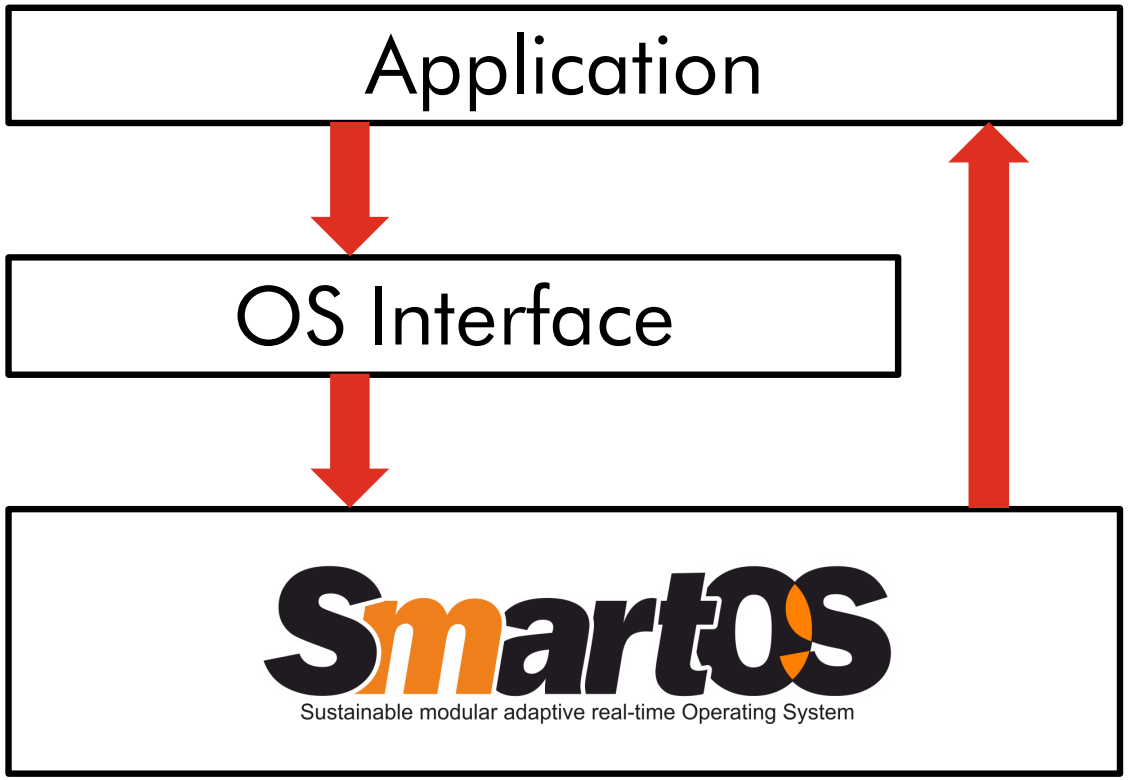
Correctness

Layered Compositional Model



Correctness

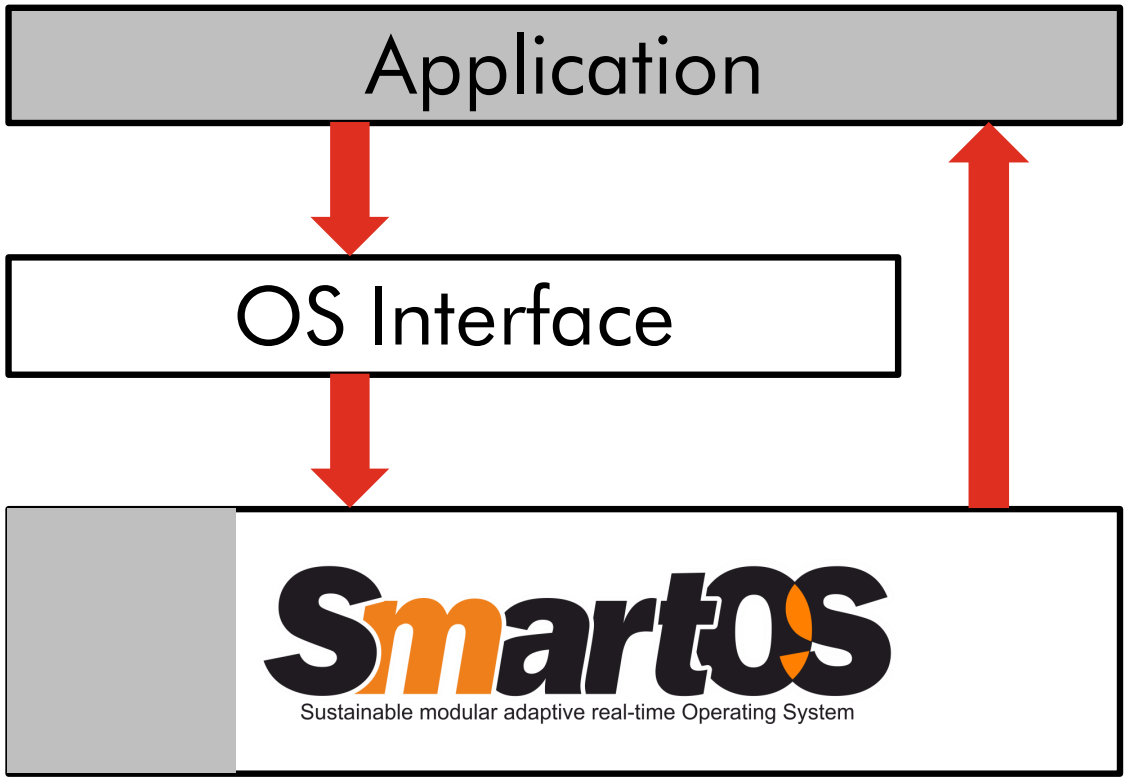
Verification Caveats



Correctness

Verification Caveats

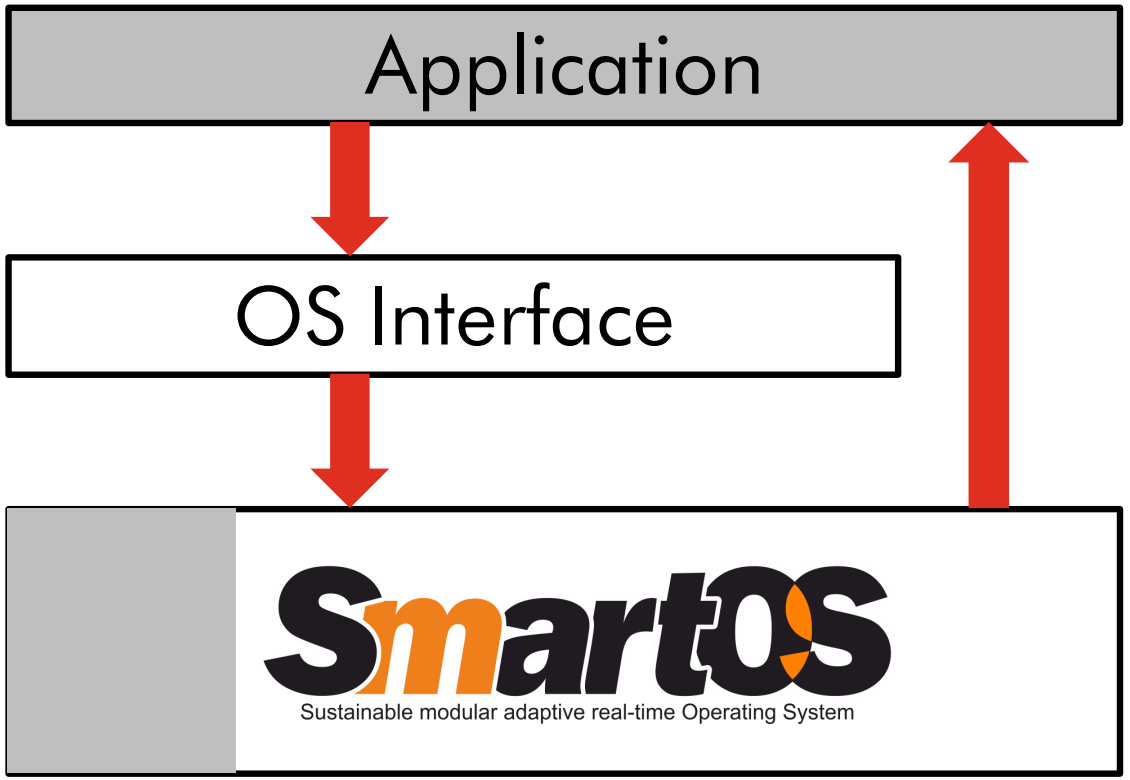
- Successful Verification



Correctness

Verification Caveats

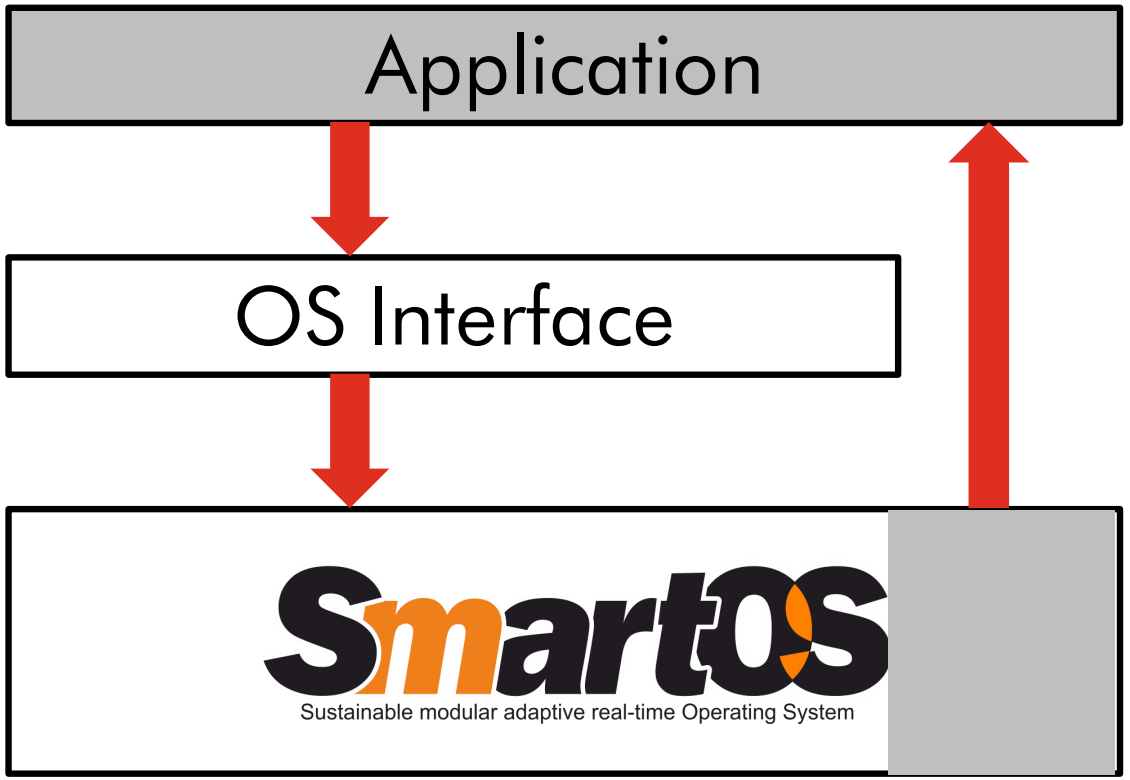
- Successful Verification
  - OS layer not fully verified



Correctness

Verification Caveats

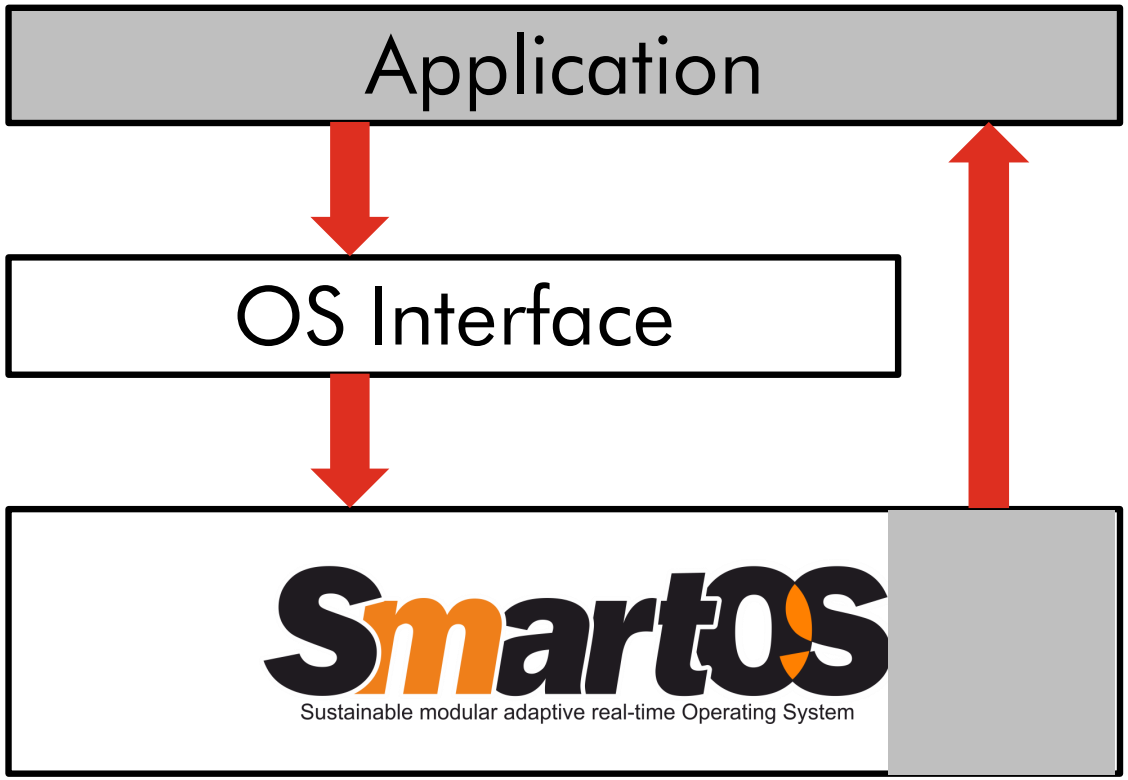
- Successful Verification
  - OS layer not fully verified
- Unsuccessful Verification



Correctness

Verification Caveats

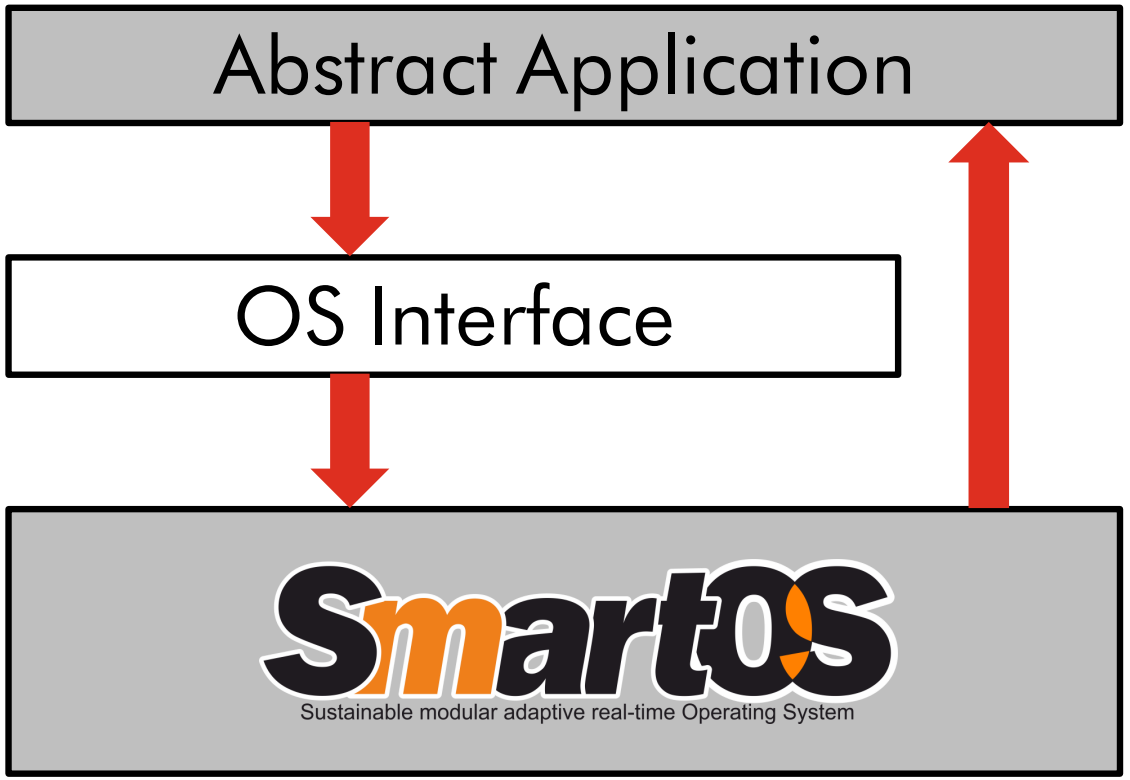
- Successful Verification
  - OS layer not fully verified
- Unsuccessful Verification
  - Unknown source of error



Correctness

Verification Caveats

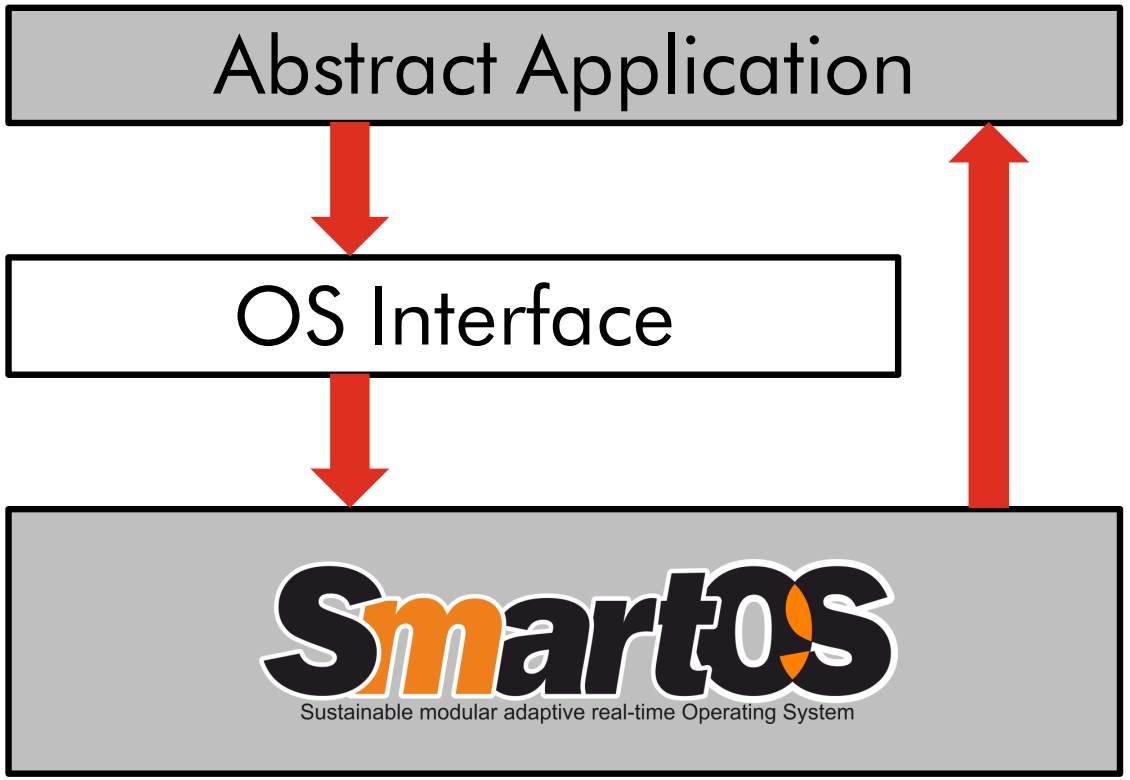
- Successful Verification
  - OS layer not fully verified
- Unsuccessful Verification
  - Unknown source of error
- Our Approach
  - Abstract application



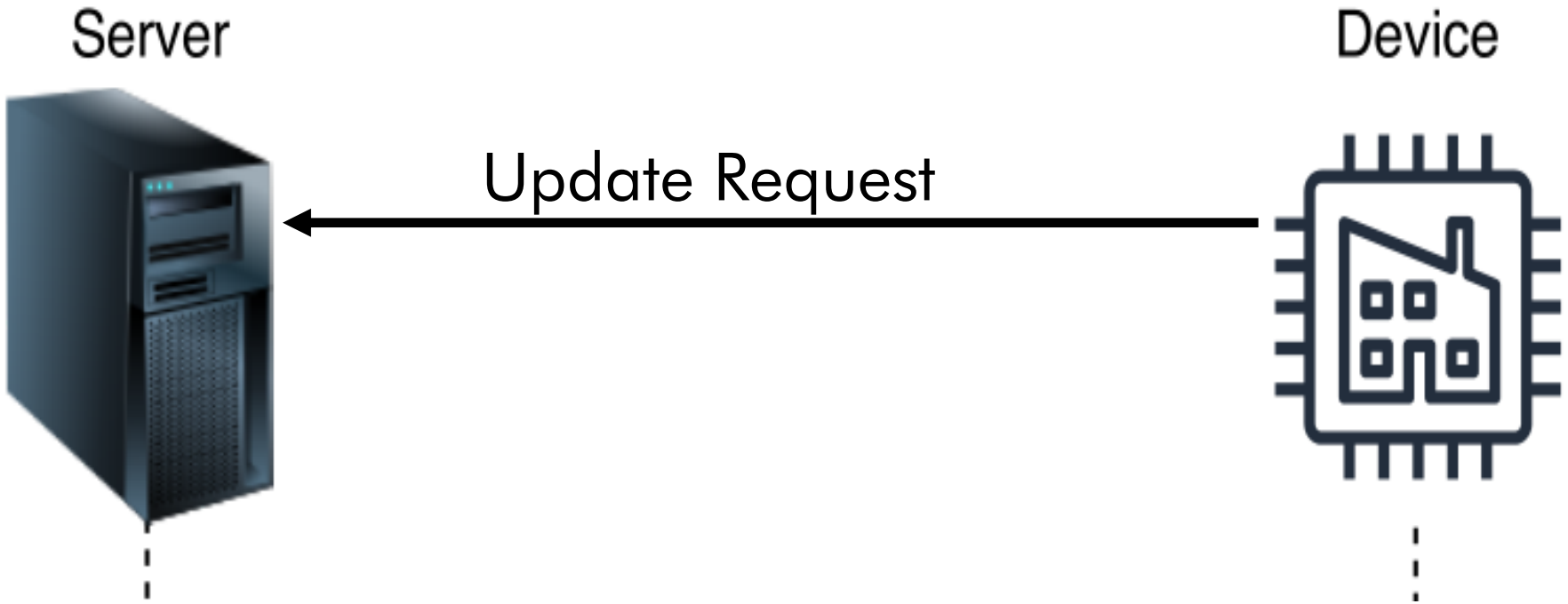
Correctness

Verification Caveats

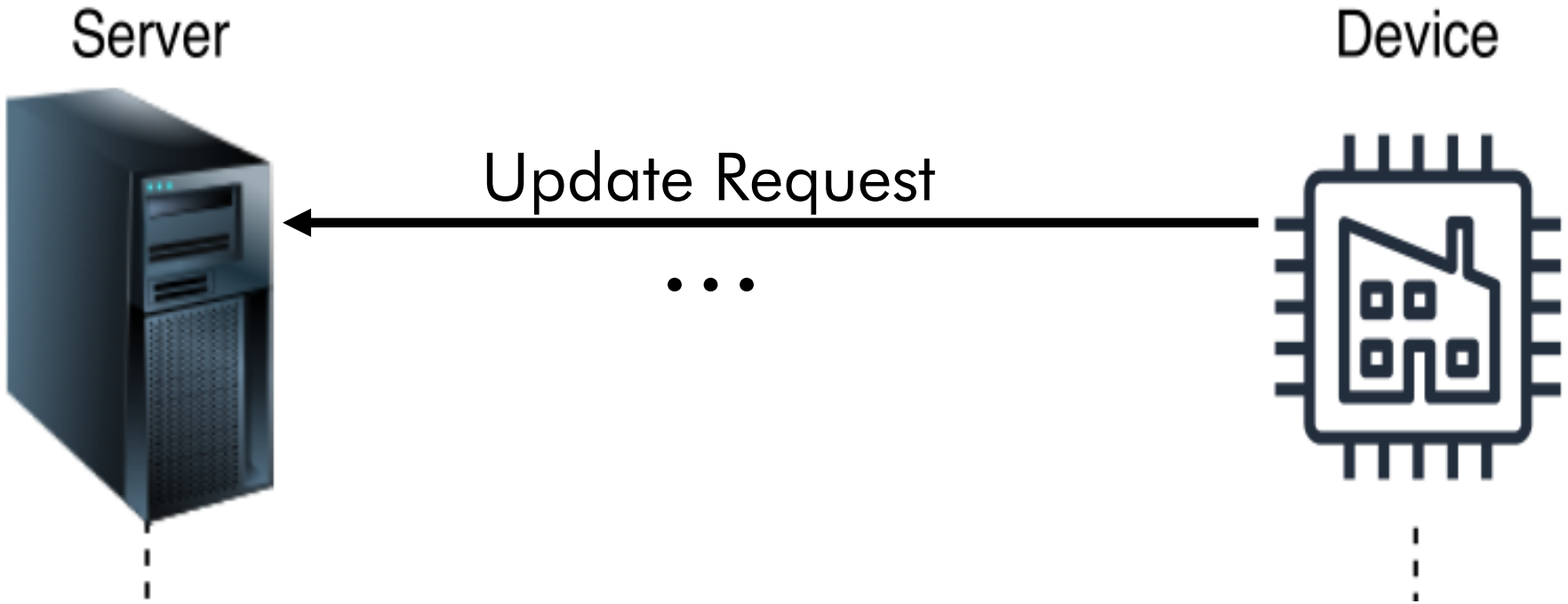
- Successful Verification
  - OS layer not fully verified
- Unsuccessful Verification
  - Unknown source of error
- Our Approach
  - Abstract application
    - Cheap full OS layer verification



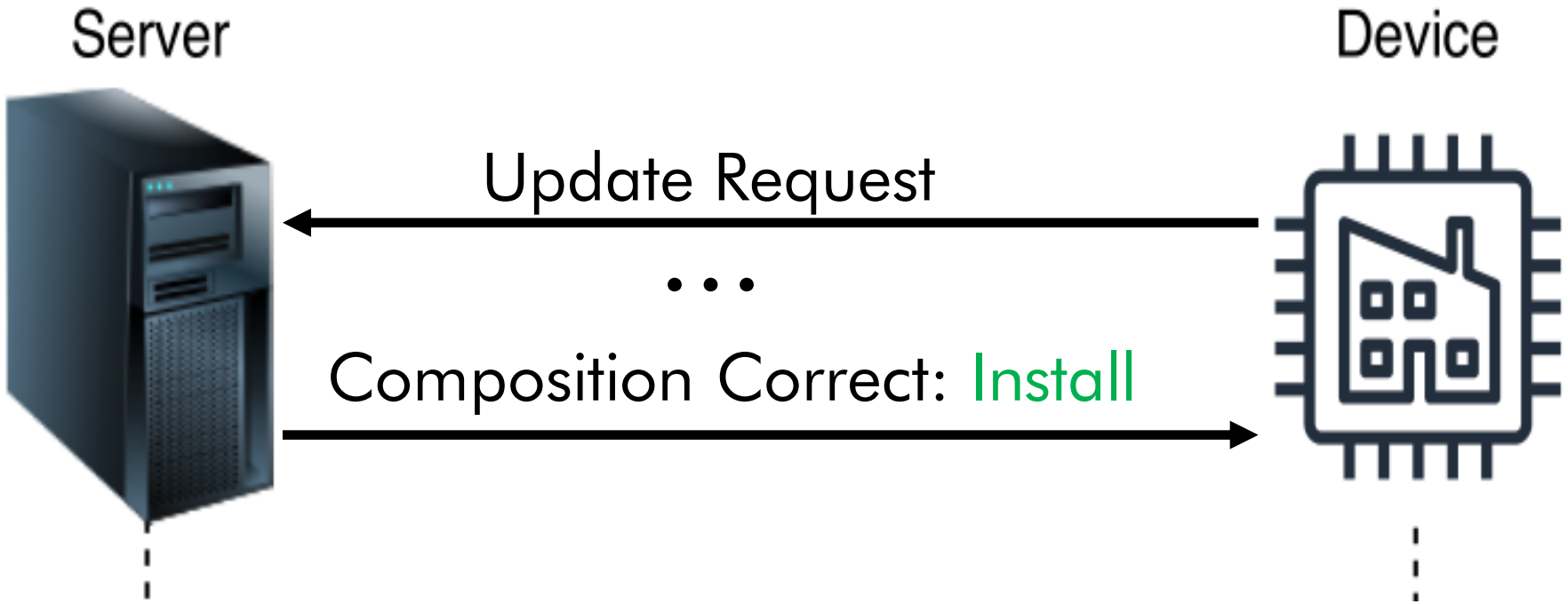
Update Overview



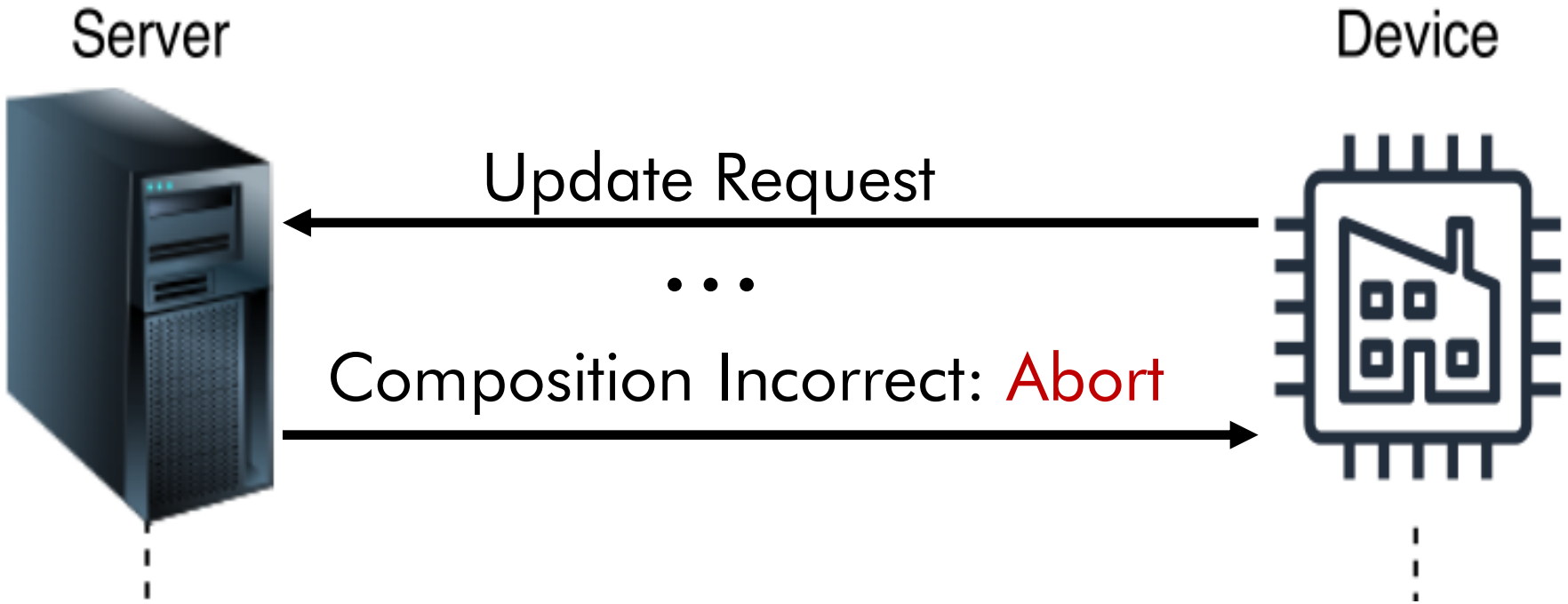
Update Overview



Update Overview



Update Overview



Potential Improvements & Future Work

- Partial Updates: Technical Features

- Formal Verification: Research

Potential Improvements & Future Work

- Partial Updates: Technical Features
  - Security Aspects

- Formal Verification: Research

Potential Improvements & Future Work

- Partial Updates: Technical Features
  - Security Aspects
  - Execution State Transfer

- Formal Verification: Research

Potential Improvements & Future Work

- Partial Updates: Technical Features
  - Security Aspects
  - Execution State Transfer

- Formal Verification: Research
  - Other Requirements

Potential Improvements & Future Work

- Partial Updates: Technical Features
  - Security Aspects
  - Execution State Transfer

- Formal Verification: Research
  - Other Requirements
  - Code-Model Mapping Correctness

Potential Improvements & Future Work

- Partial Updates: Technical Features
  - Security Aspects
  - Execution State Transfer

- Formal Verification: Research
  - Other Requirements
  - Code-Model Mapping Correctness
  - Automatic Code Generation

Potential Improvements & Future Work

- Partial Updates: Technical Features
  - Security Aspects
  - Execution State Transfer

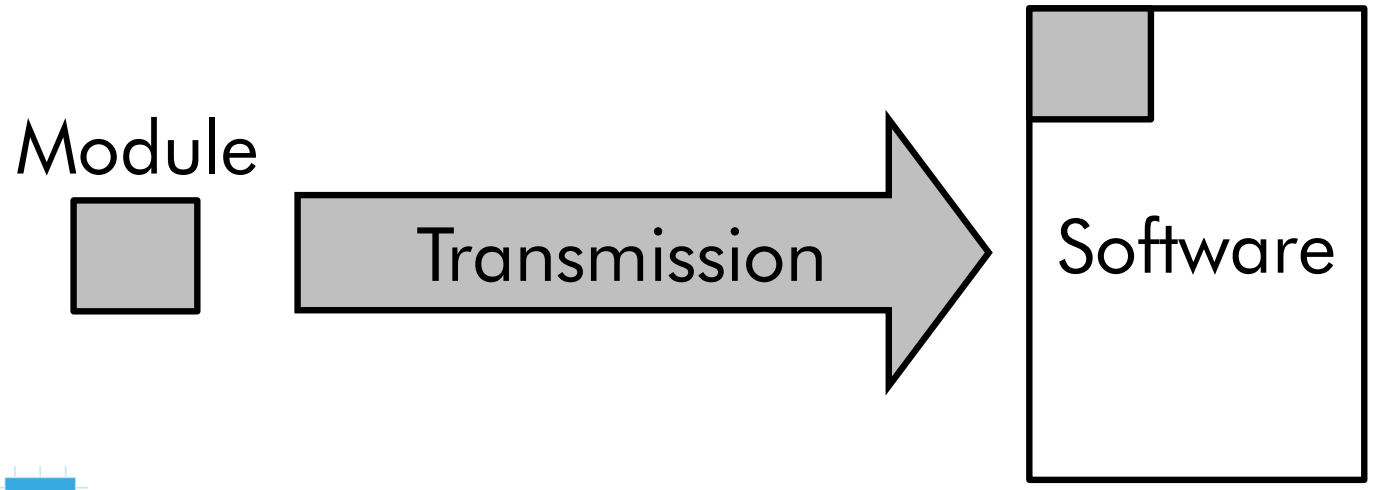
- Formal Verification: Research
  - Other Requirements
  - Code-Model Mapping Correctness
  - Automatic Code Generation
  - State Space Explosion

Summary

- Work divided in two major parts
  1. How to support partial updates

Summary

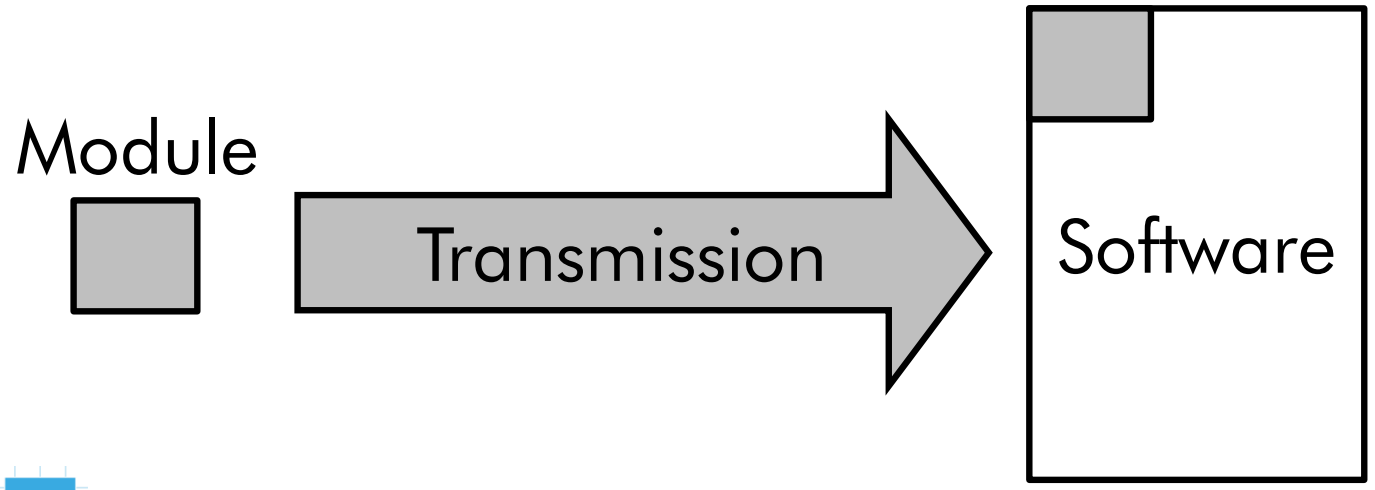
- Work divided in two major parts
  1. How to support partial updates

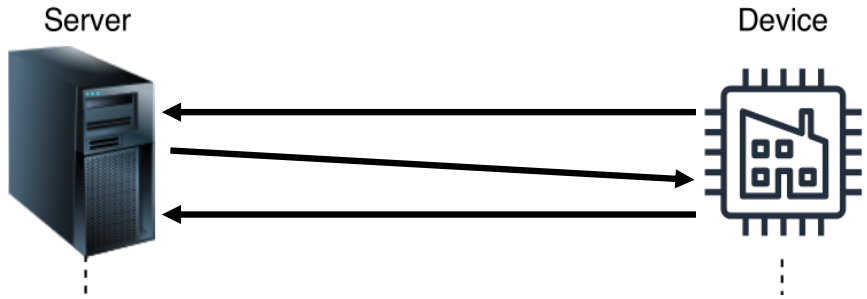


Summary

- Work divided in two major parts
  1. How to support partial updates

SmartOS
Sustainable modular adaptive real-time Operating System

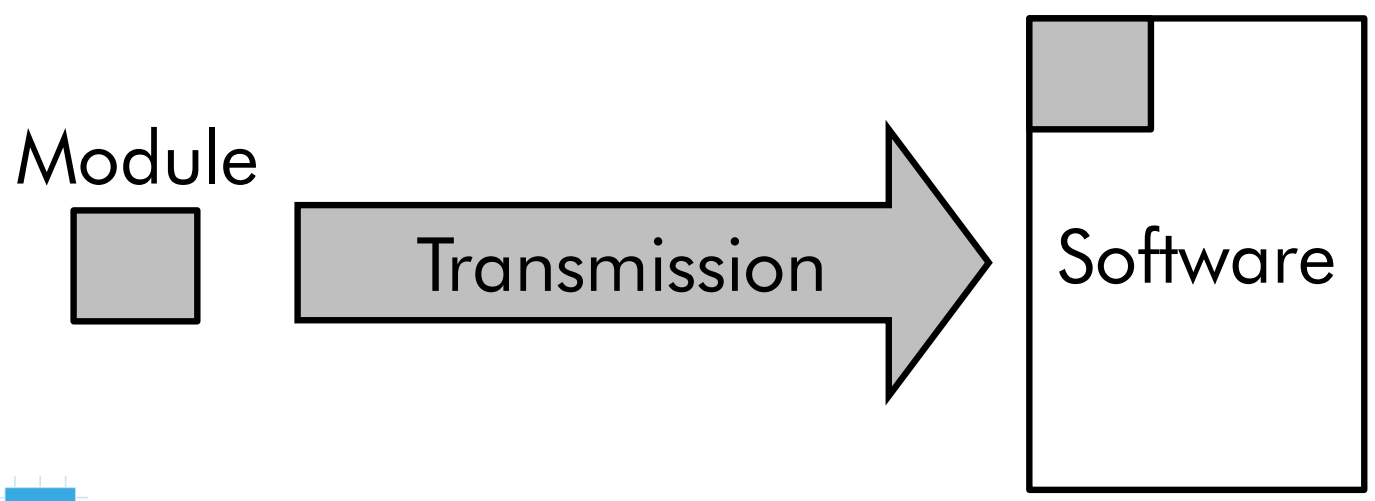


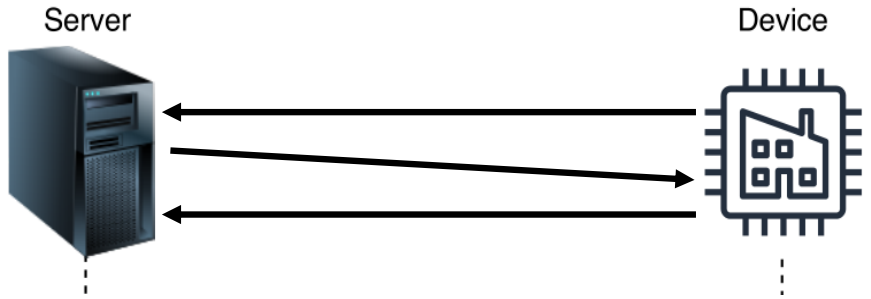


Summary

- Work divided in two major parts
  1. How to support partial updates

SmartOS
Sustainable modular adaptive real-time Operating System





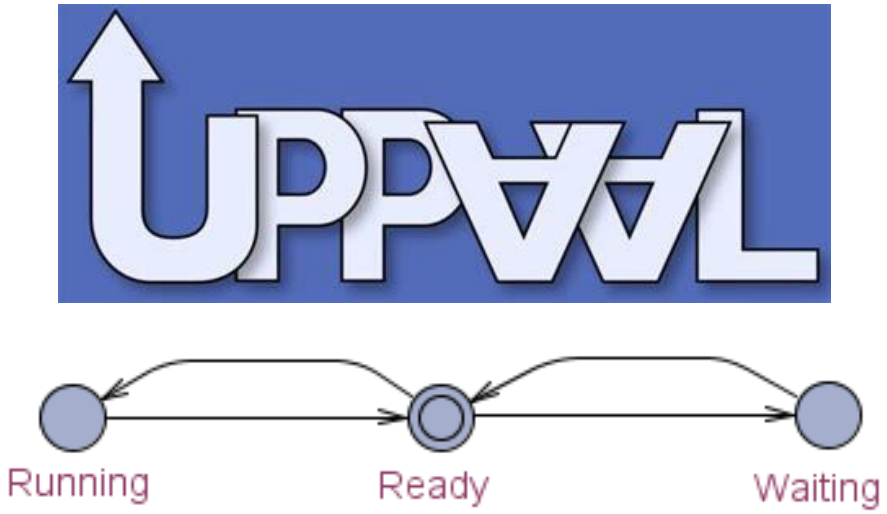
HW Support

Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions

Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions



Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions

Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions

Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions
     - Current Focus
       - Liveness
       - Real-Time

Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions
     - Current Focus
       - Liveness
       - Real-Time

- Future Work
  - Partial Updates: Technical Improvements

Summary

- Work divided in two major parts
  1. How to support partial updates
  2. How to guarantee correctness of software compositions
     - Current Focus
       - Liveness
       - Real-Time

- Future Work
  - Partial Updates: Technical Improvements
  - Correctness: Research Challenges

Thank you!

Leandro Batista Ribeiro
lbatistaribeiro@tugraz.at

Institute of Technical Informatics
Embedded Automotive Systems Group
Graz University of Technology

References

[EWSN'20] Leandro Batista Ribeiro, Fabian Schlager, and Marcel Baunach. "Towards Automatic SW Integration in Dependable Embedded Systems."
International Conference on Embedded Wireless Systems and Networks. 2020

[CODES'21] Malenko, Maja, Leandro Batista Ribeiro, and Marcel Baunach. "Improving security and maintainability in modular embedded systems with hardware support: work-in-progress."
International *Conference on Hardware/Software Codesign and System Synthesis*. 2021.

[DP] https://spin.atomicobject.com/2012/10/31/elixir-erlang-and-the-dining-philosophers/