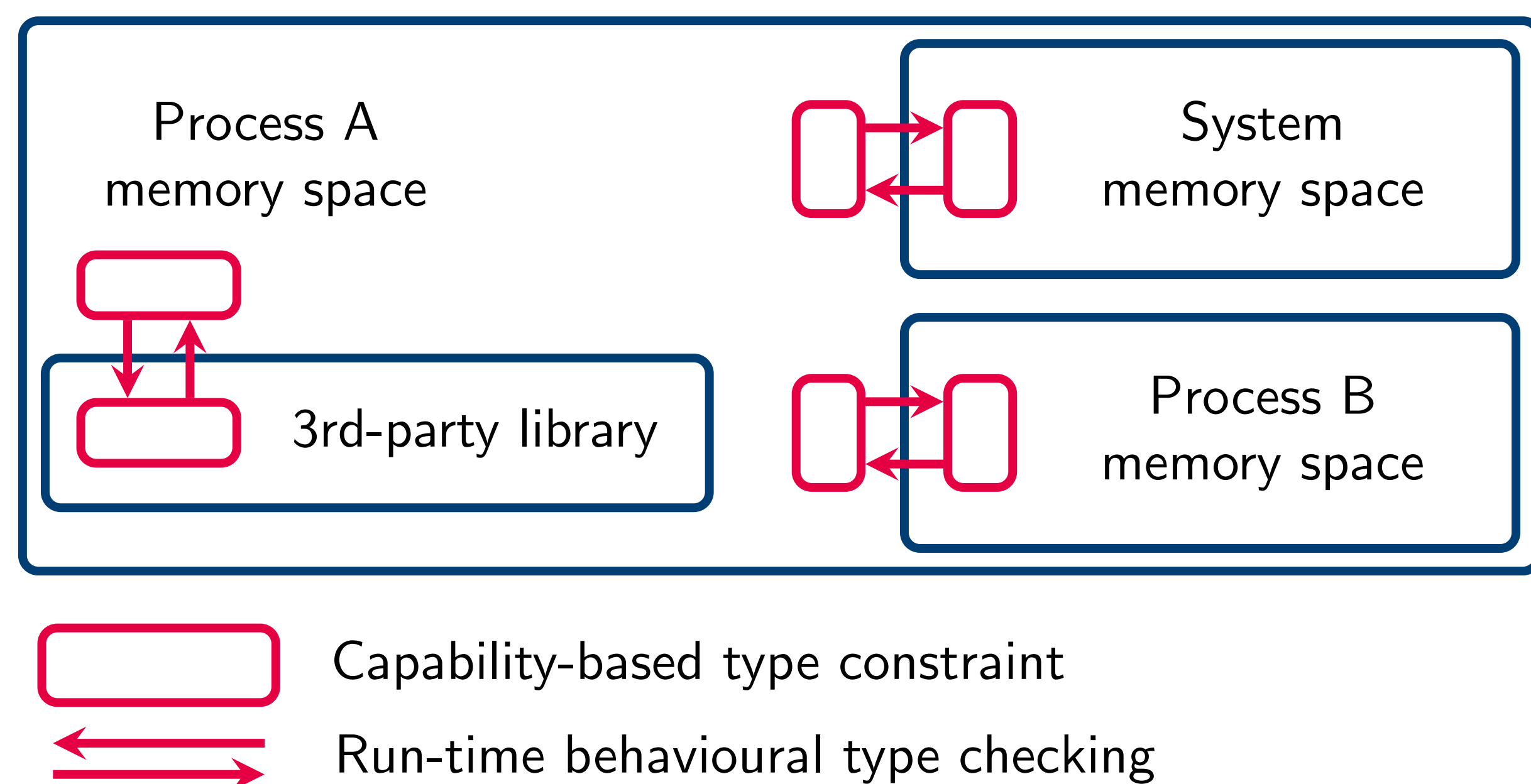


# APPCONTROL: ENFORCING APPLICATION BEHAVIOUR THROUGH TYPE-BASED CONSTRAINTS

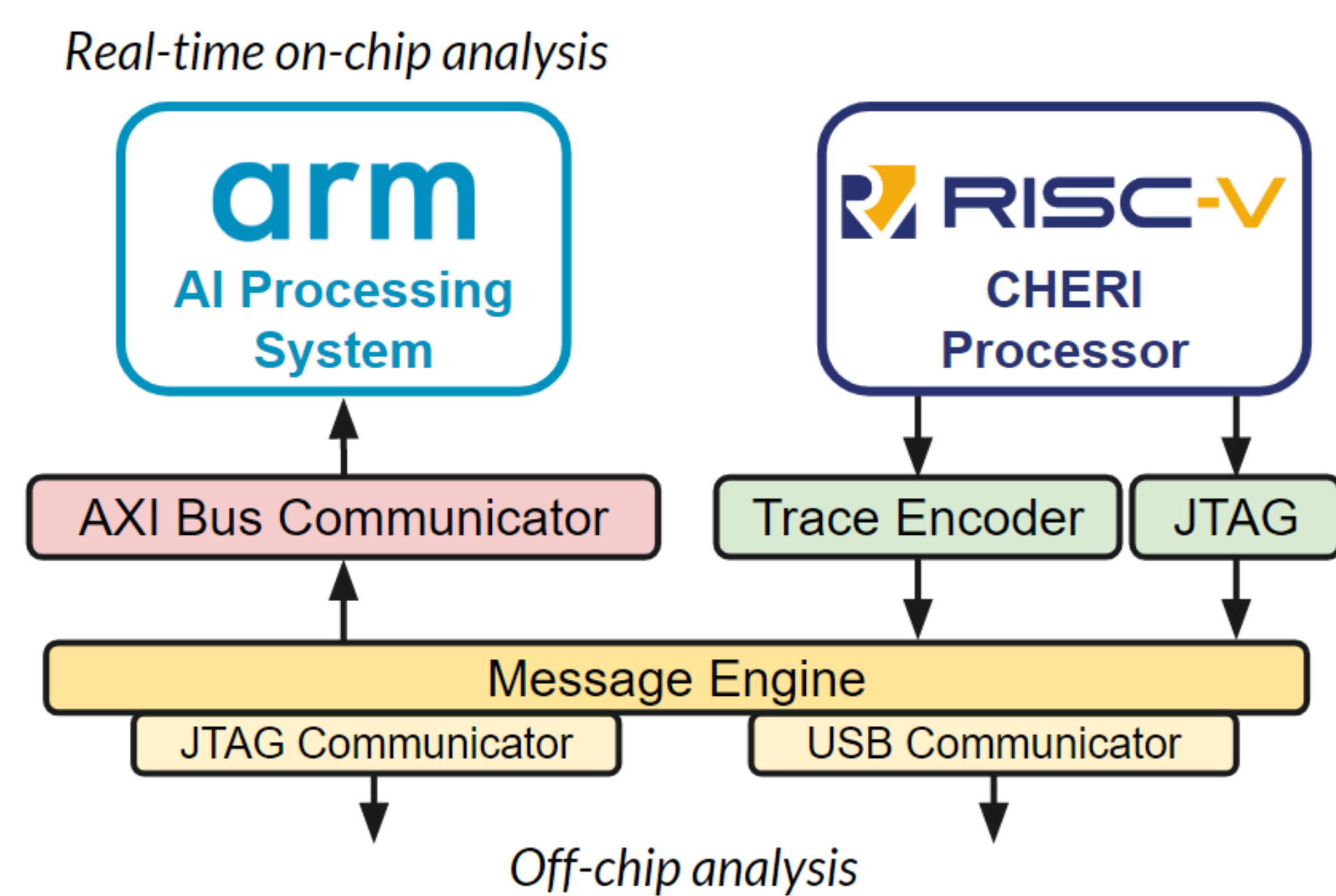
Wim Vanderbauwhede<sup>1</sup>, José Cano<sup>1</sup>, Jan de Muijnck-Hughes<sup>1</sup>, Laura Voinea<sup>1</sup>, Nobuko Yoshida<sup>2</sup>, Adam Barwell<sup>2</sup>, Klaus McDonald-Maier<sup>3</sup>, Xiaojun Zhai<sup>3</sup>, Ludovico Poli<sup>3</sup>, Michal Borowski<sup>3</sup>  
<sup>1</sup>University of Glasgow, <sup>2</sup>Imperial College London, <sup>3</sup>University of Essex

## PROJECT OVERVIEW

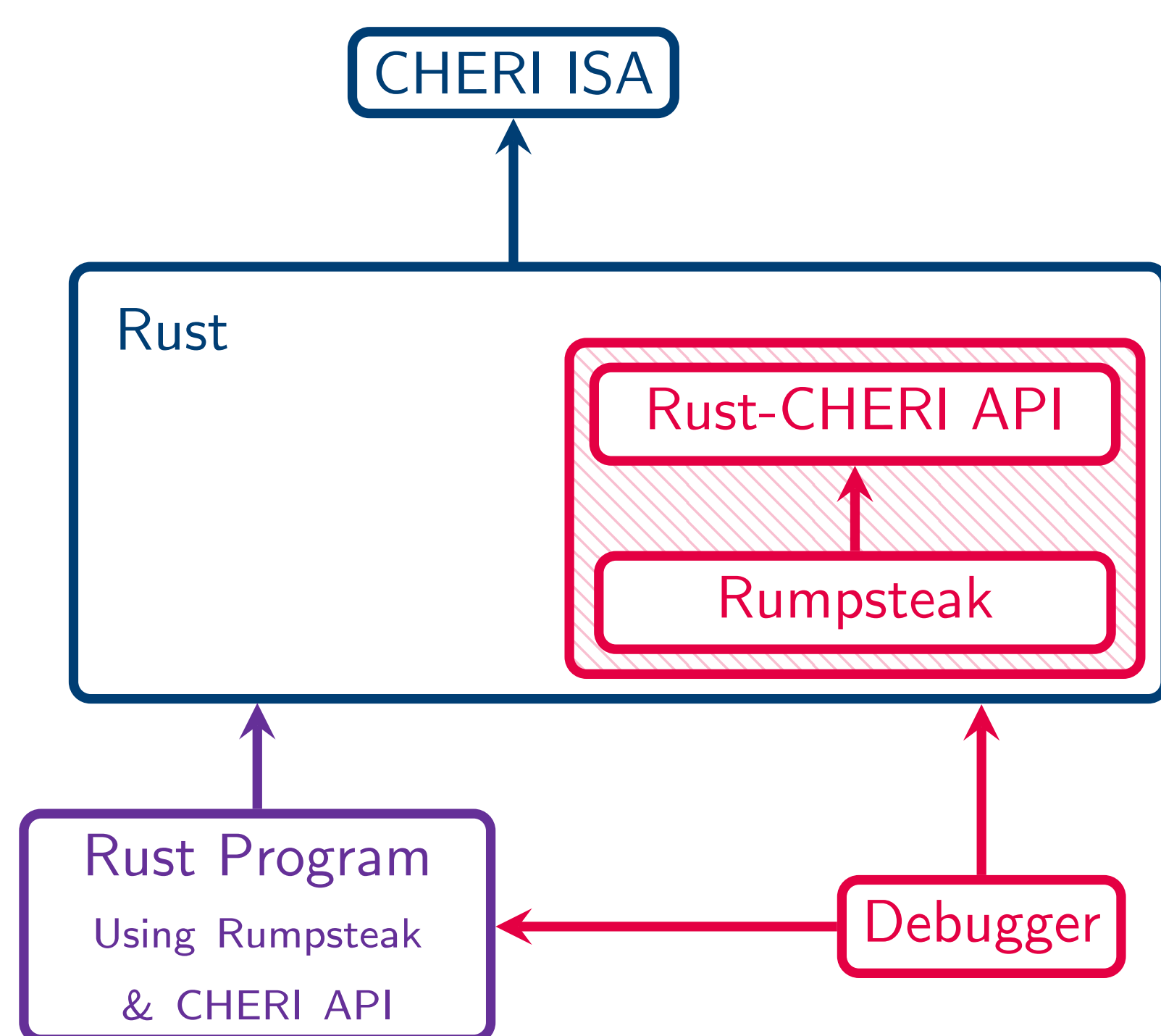
CHERI capabilities provide fine-grained memory protection, limiting access privileges of third-party applications. However, in order to **secure program interaction**, since capabilities say nothing about **program behaviour**, we use **Behavioural Types** to capture the behavioural structure of application interfaces.



Behavioural types will ensure correctness of behaviour, provided that the specification is correct. Debugging a specification-based system demands the ability to **debug the specification at run-time**.



## OUR APPROACH



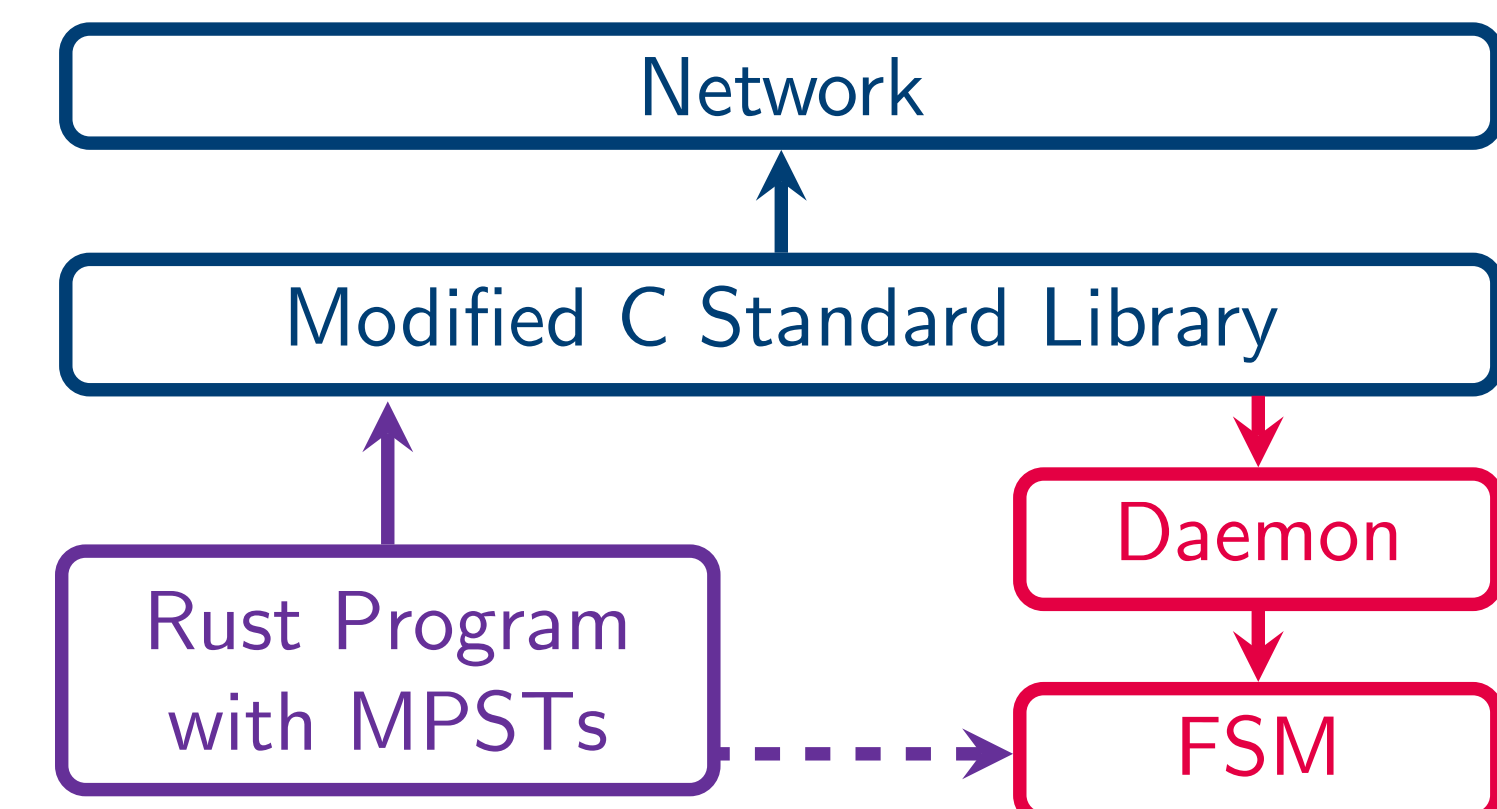
- Develop a **Rust API** enabling use of CHERI Capabilities.
- Develop **multiparty session type (MPST)** theories and tools to ensure capability-based behavioural properties in Rust.
- Develop a framework to enable the monitoring and **debugging** of capability-supporting Rust code.

## ACKNOWLEDGEMENTS

The authors thank the UKRI Digital Security by Design (DSbD) Programme for funding the AppControl project through grant EP/V000462/1 and Morello-Hat through grant EP/X015955/1.

## RUST API

The Rust API will connect MPSTs with CHERI's memory control capabilities. We identify socket layer system calls to be the most important use case for monitoring adherence to the specification. As Rust's POSIX socket library is built on top of the C standard library, we can address this by modifying the libC to communicate these calls to a listening daemon. The daemon can maintain a finite-state machine (FSM) determined by the specification.



## MPST IN RUST

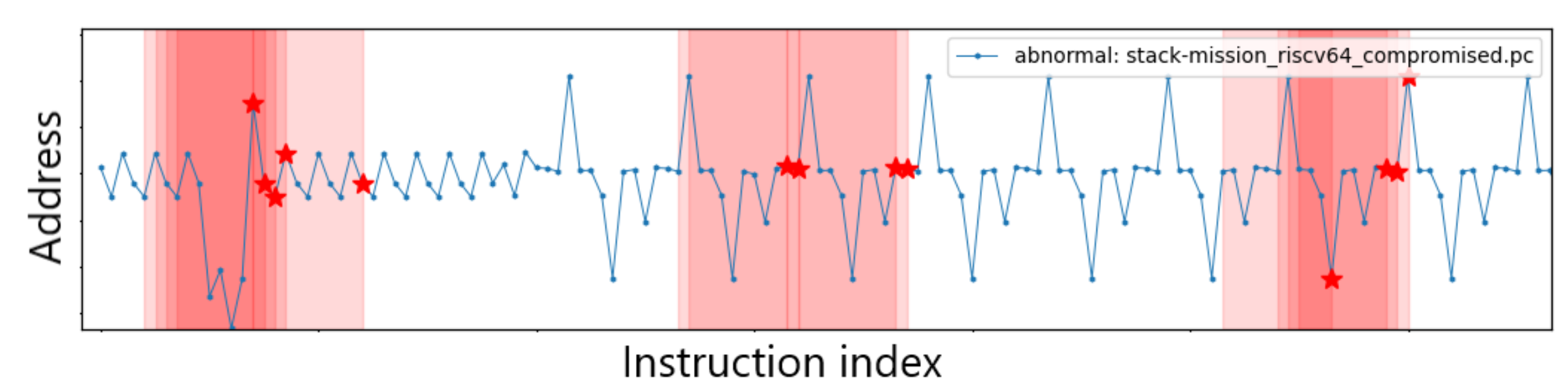
- Developed the Rumpsteak library, supporting asynchronous MPST in Rust with automatic message reordering.
- Communications between two processes or a 3rd party library can be **statically type-checked** for behavioural correctness, ensuring **deadlock-freedom** of protocols.
- We are currently adapting Rumpsteak for the Rust API.
- <https://github.com/zakcutner/rumpsteak>

A ring protocol in Rumpsteak:

```
1 type Source = Receive<T, Ready, Select<T, SourceChoice>>;
2 enum SourceChoice { Value(Value, Source),
3                     Stop(Stop, End) }
4 type Sink = Send<S, Ready, Branch<S, SinkChoice>>;
5 enum SinkChoice { Value(Value, Sink),
6                  Stop(Stop, End) }
```

## DEBUGGER AND DIAGNOSTICS TOOLS

- Development of prototypes of non-intrusive, on-chip debug system for CHERI-RISC-V processors.
- Extraction of program metrics from the custom implementation of CHERI Flute based SoC.
- Performance comparison of conventional and novel (e.g. autoencoder-forest) abnormal behaviour detection methods.



## PROJECT WEB SITE

<https://dsbd-appcontrol.github.io/>



## MORELLO-HAT: HIGH-LEVEL API AND TOOLING

Wim Vanderbauwhede<sup>1</sup>, José Cano<sup>1</sup>, Cristian Urla<sup>1</sup>, Nobuko Yoshida<sup>2</sup>, Adam Barwell<sup>2</sup>, Klaus McDonald-Maier<sup>3</sup>, Xiaojun Zhai<sup>3</sup>, Ludovico Poli<sup>3</sup>, Michal Borowski<sup>3</sup>

The **Morello-HAT** project intends to create a common API that can be used by compiler developers as well as programmers of higher-level languages, to allow them to leverage Morello's HW capabilities to improve memory security and type safety, spatial as well as temporal, of their language and programs.

The project consists of three work packages, one to develop the API using C++, Rust, Go and Dart, one to demonstrate the usability and effectiveness of the API on a series of example applications by ML-based detection of vulnerabilities and assessment of the effectiveness of mitigation through the use of the API and one to use HW capabilities to enhance the debug infrastructure.