Adaptive Instrumentation of Java-Applications for Experiment-Based Performance Analysis

***Henning Schulz, Albert Flaig*,**
Alexander Wert and André van Hoorn

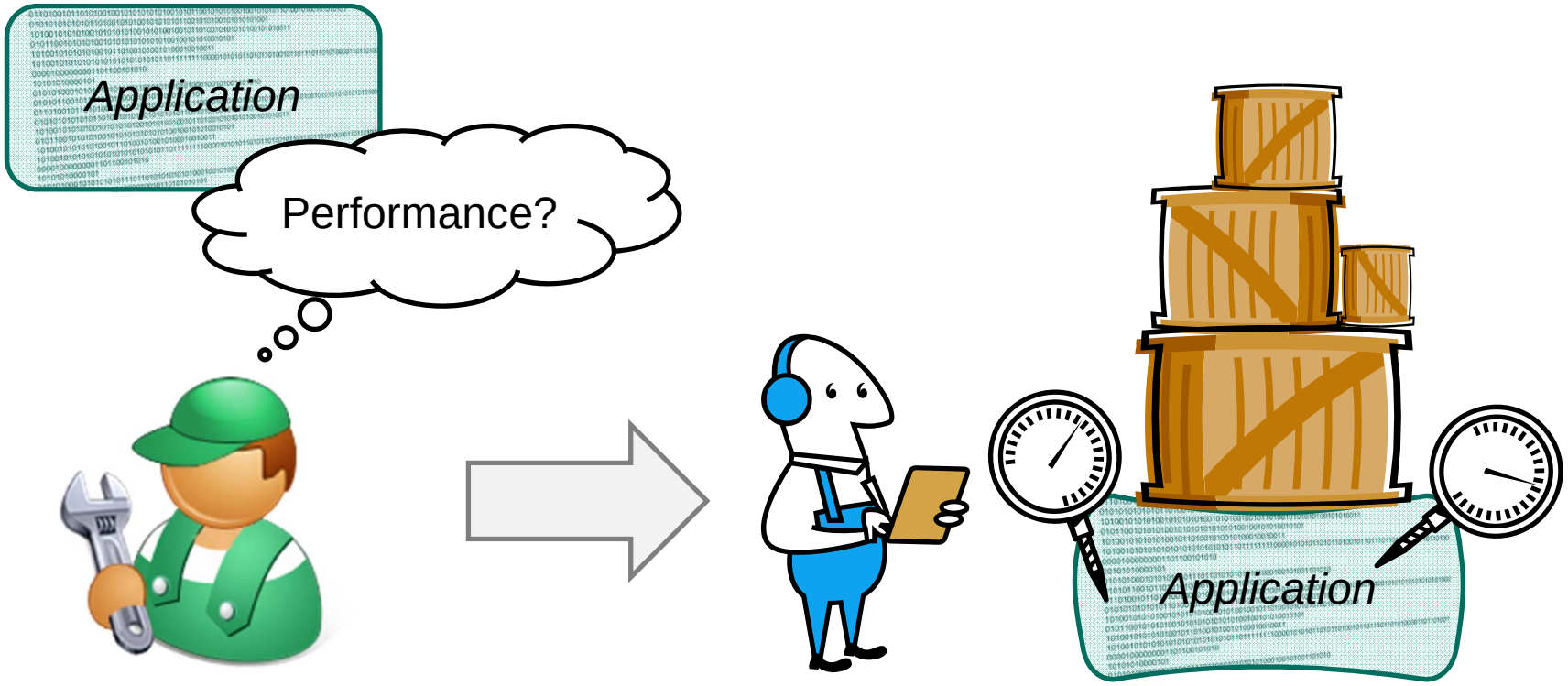
Symposium on Software Performance 2014

28.11.2014

KIT – SOFTWARE DESIGN AND QUALITY GROUP

UNIVERSITY OF STUTTGART – RELIABLE SOFTWARE SYSTEMS GROUP

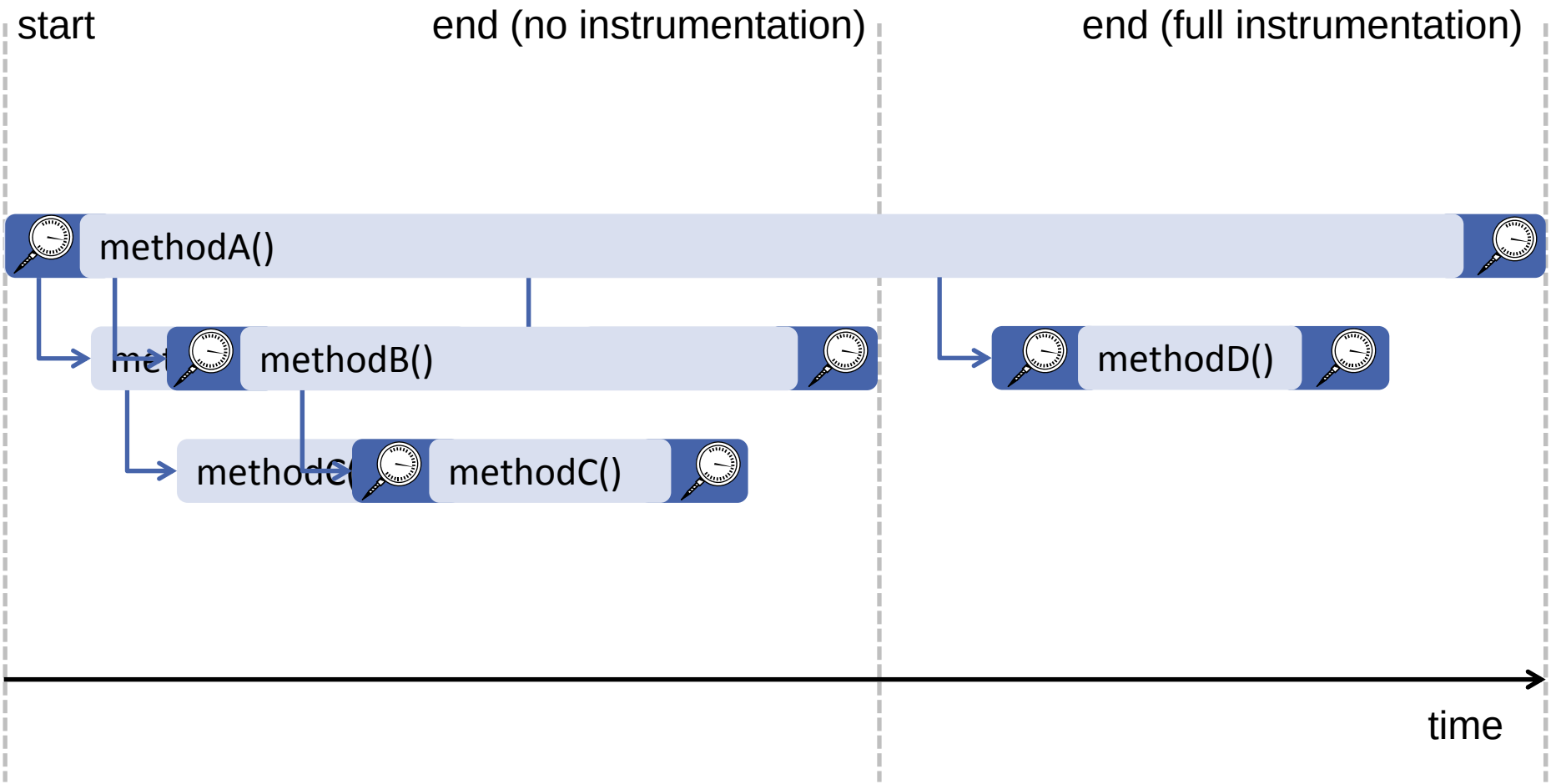
Motivation



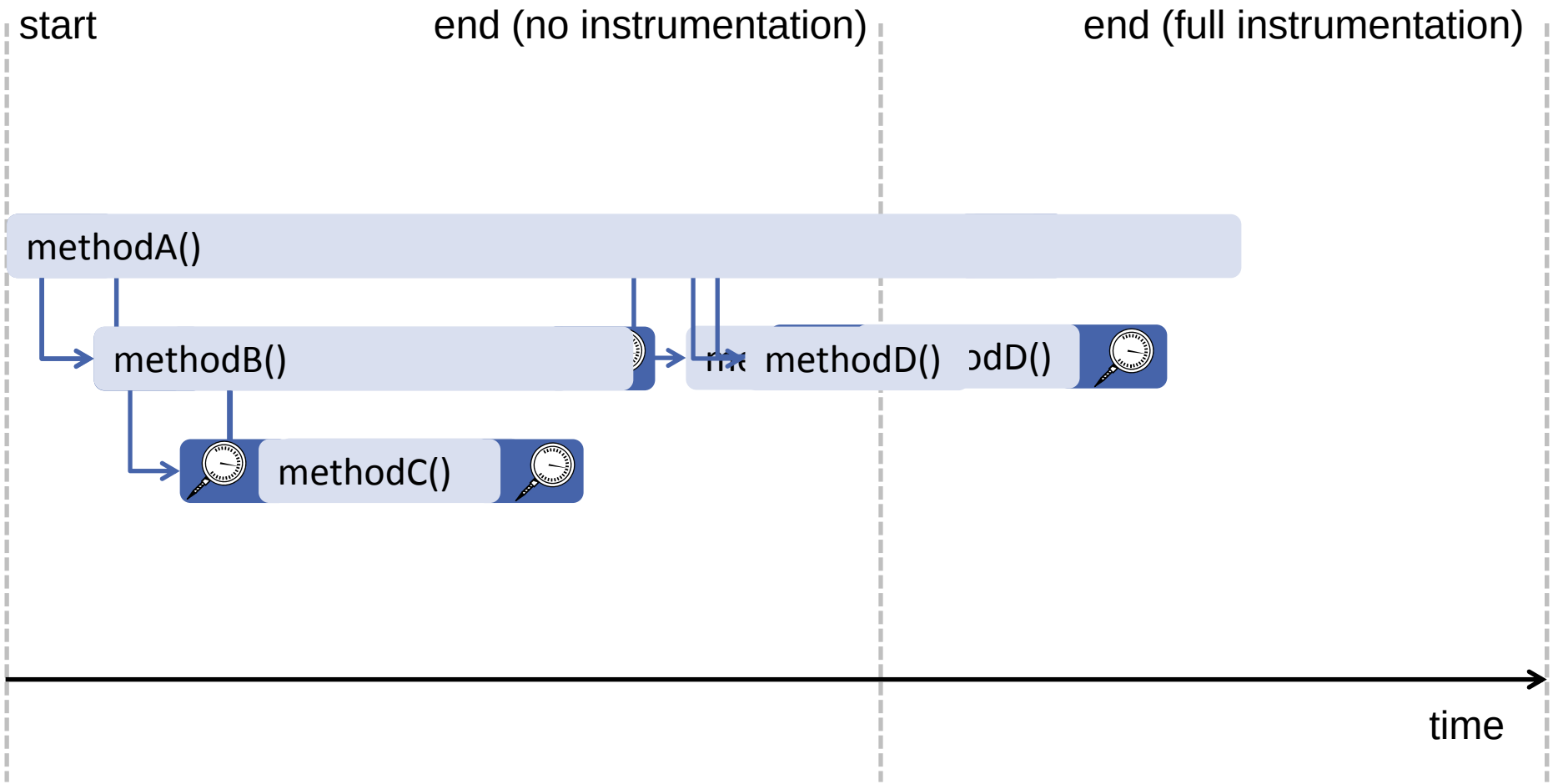
Performance Engineer

Instrumentation

Instrumentation Overhead



Adaptive Instrumentation



Kieker vs. AIM

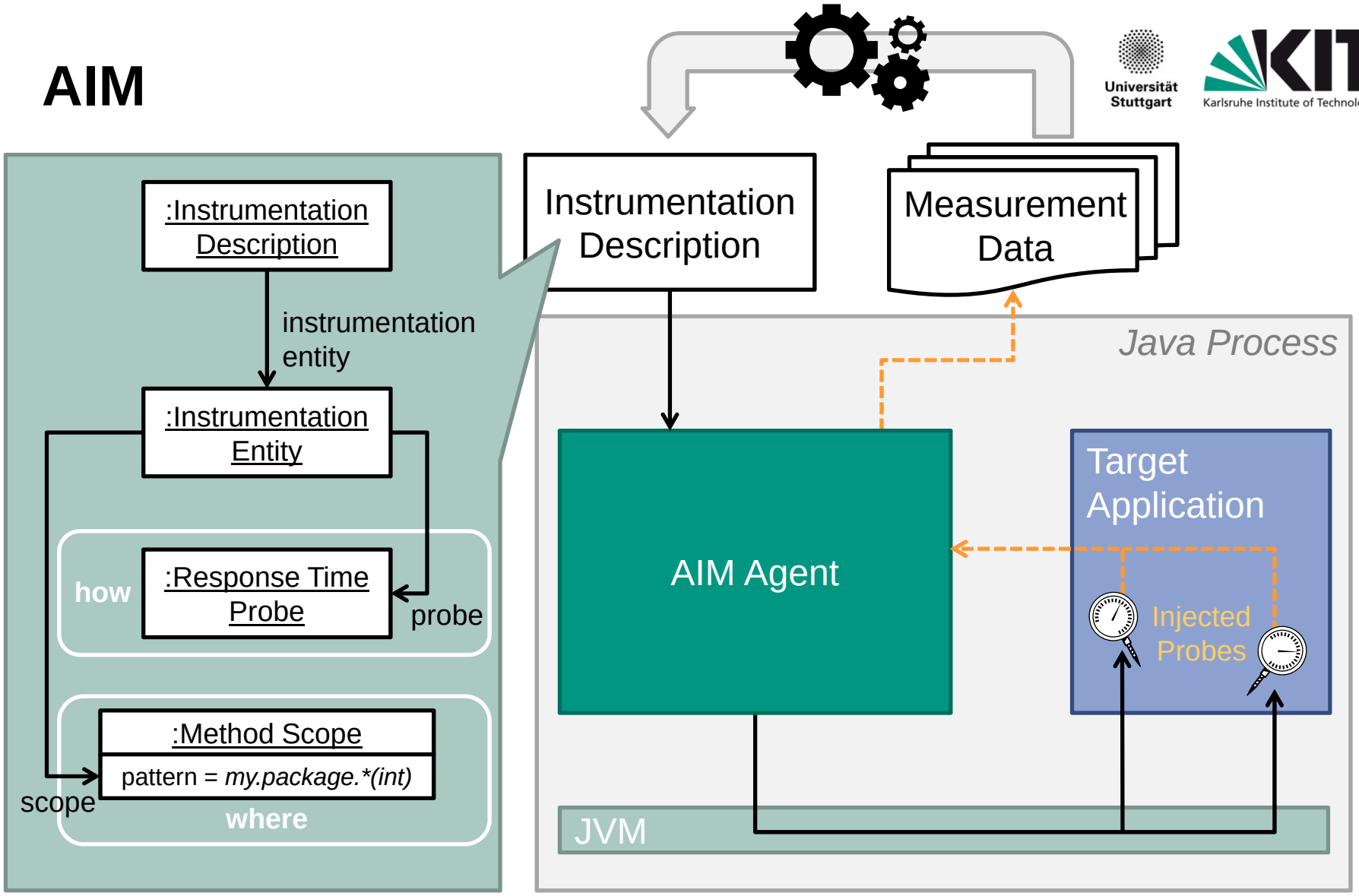
Kieker (http://kieker-monitoring.net)	**AIM** (http://sopeco.github.io/AIM)
Instrumentation at application start-up	Instrumentation on demand
Statically adaptive instrumentation	Dynamically adaptive
Extensible (writer, etc.)	Extendable (writer, etc.)
Analysis infrastructure	No analysis at all
Focus on production	Focus on experiments
Regexps for adaptive monitoring	Instrumentation description language

- Use dynamic adaptive instrumentation from AIM
- Use instrumentation description language
- Use Kieker‘s monitoring & analysis infrastructure

AIM IN EXPERIMENT-BASED CONTEXT

Henning Schulz: *Adaptive Instrumentation of Java-Applications for Experiment-Based Performance Analysis*, Bachelor‘s Thesis, Karlsruhe Institute of Technology, 2014

AIM


:Instrumentation Description
instrumentation entity
:Instrumentation Entity
how
:Response Time Probe
probe
:Method Scope
pattern = my.package.*(int)
scope
where
Instrumentation Description
Measurement Data
Java Process
AIM Agent
Target Application
Injected Probes
JVM

Evaluation

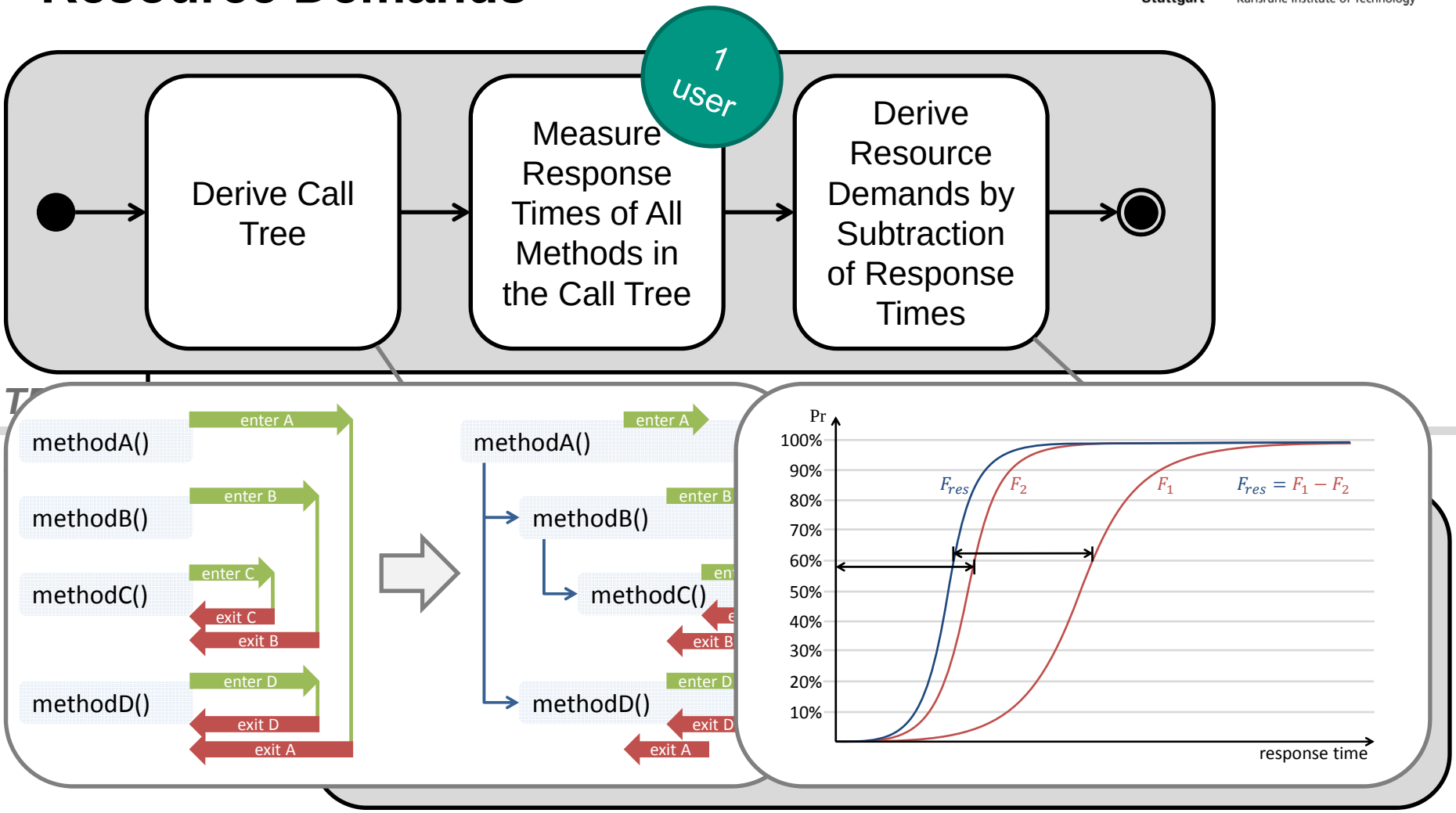
AIM is comparable to existing approaches

- Compare to Kieker and DiSL
- *Instrumentation process*:
  - fast for < 500 methods
  - comparatively slow for > 500 methods
- *Instrumentation overhead*: smaller than Kieker's

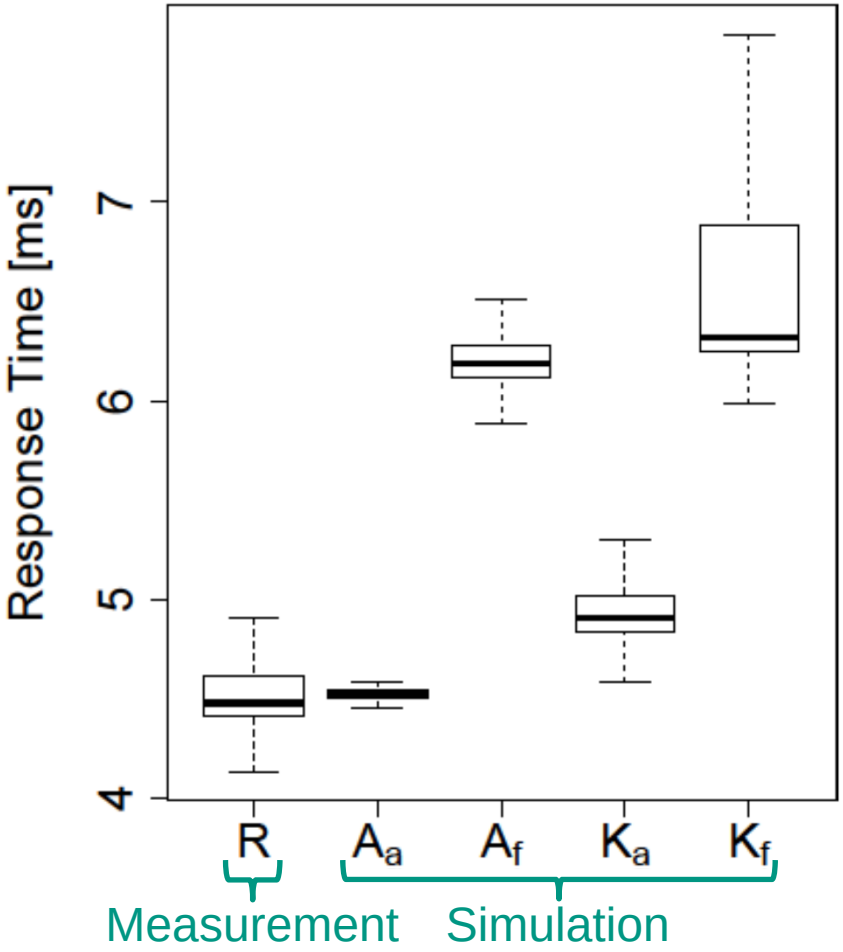
Benefits of adaptapable instrumentation

- Automated Palladio Resource Demand Estimation

Automated Estimation of Palladio Resource Demands



Comparing PRD Qualities



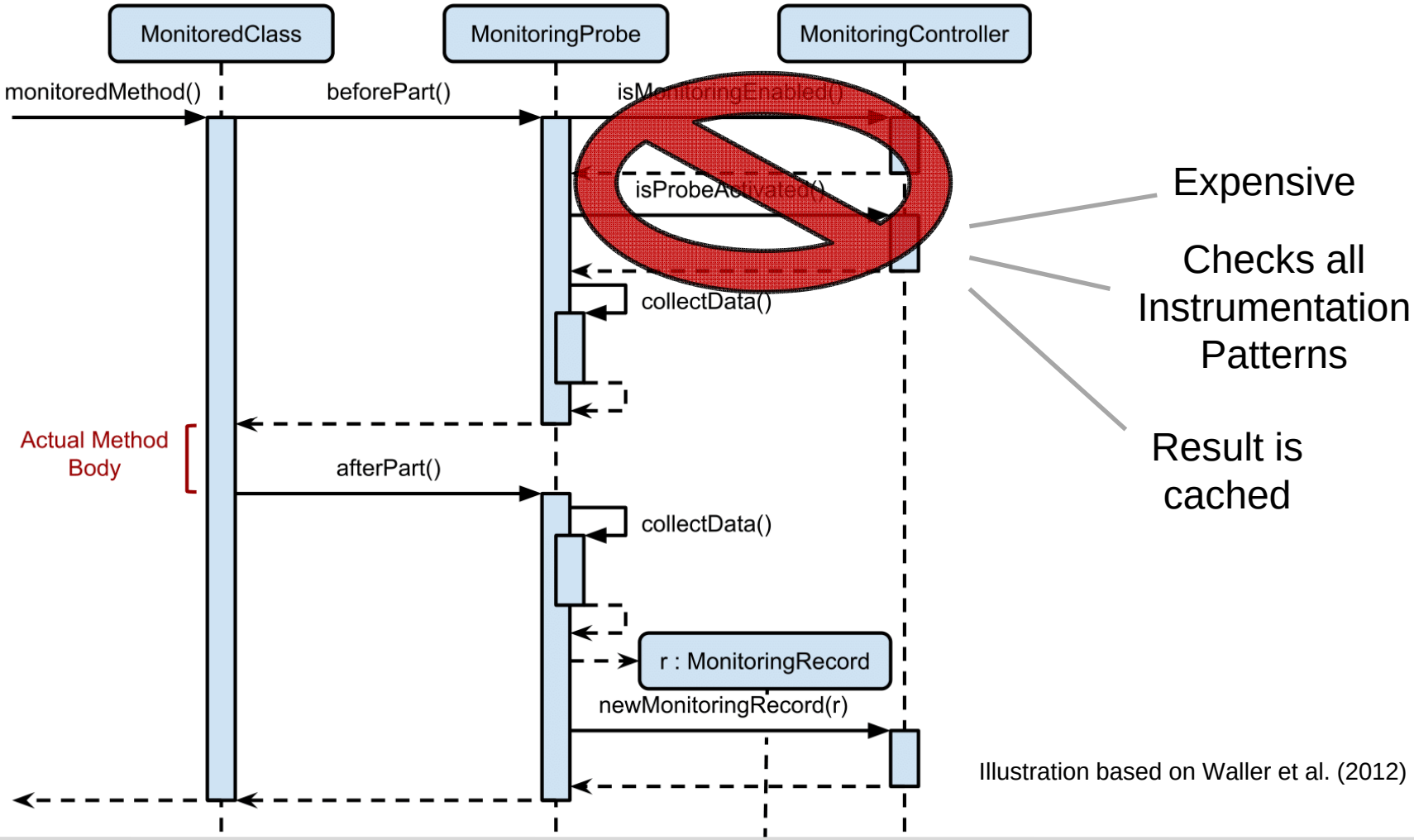
	mean [ms]	mean error [%]
Reference	4.52	-
AIM (adaptive)	4.52	**0.01**
AIM (full)	6.21	37.3
Kieker (adaptive)	4.95	9.4
Kieker (full)	6.57	45.4

Legend: **R**: Reference **A_a**: AIM adaptive **A_f**: AIM full **K_a**: Kieker adaptive **K_f**: Kieker full

INTEGRATION OF KIEKER & AIM

Albert Flaig: *Dynamic Instrumentation in Kieker Using Runtime Bytecode Modification*,
Bachelor‘s Thesis, University of Stuttgart, 2014

Statically Adaptive Monitoring in Kieker



Expensive

Checks all Instrumentation Patterns

Result is cached

Illustration based on Waller et al. (2012)

Dynamically Adaptive Monitoring in Kieker

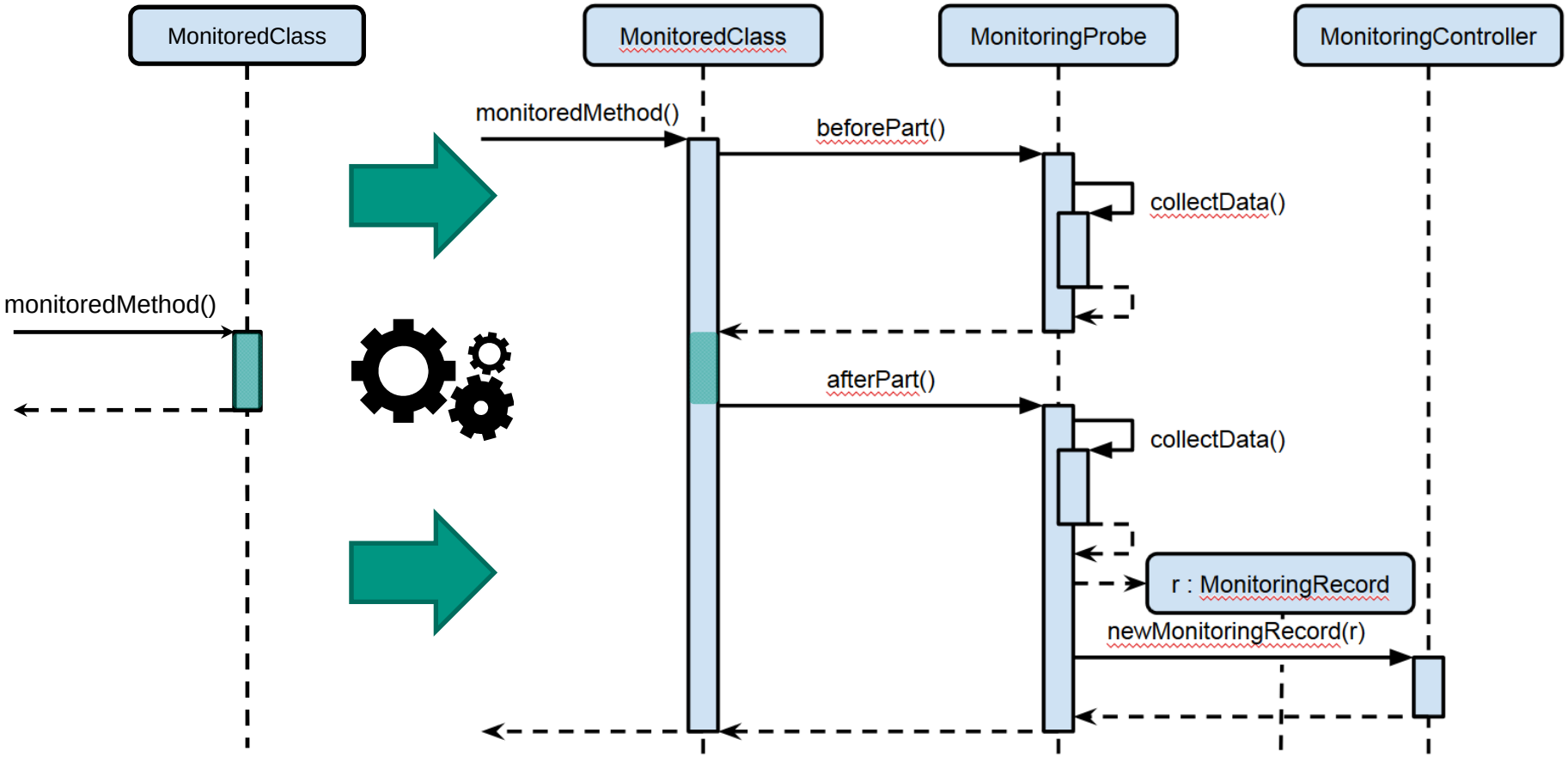
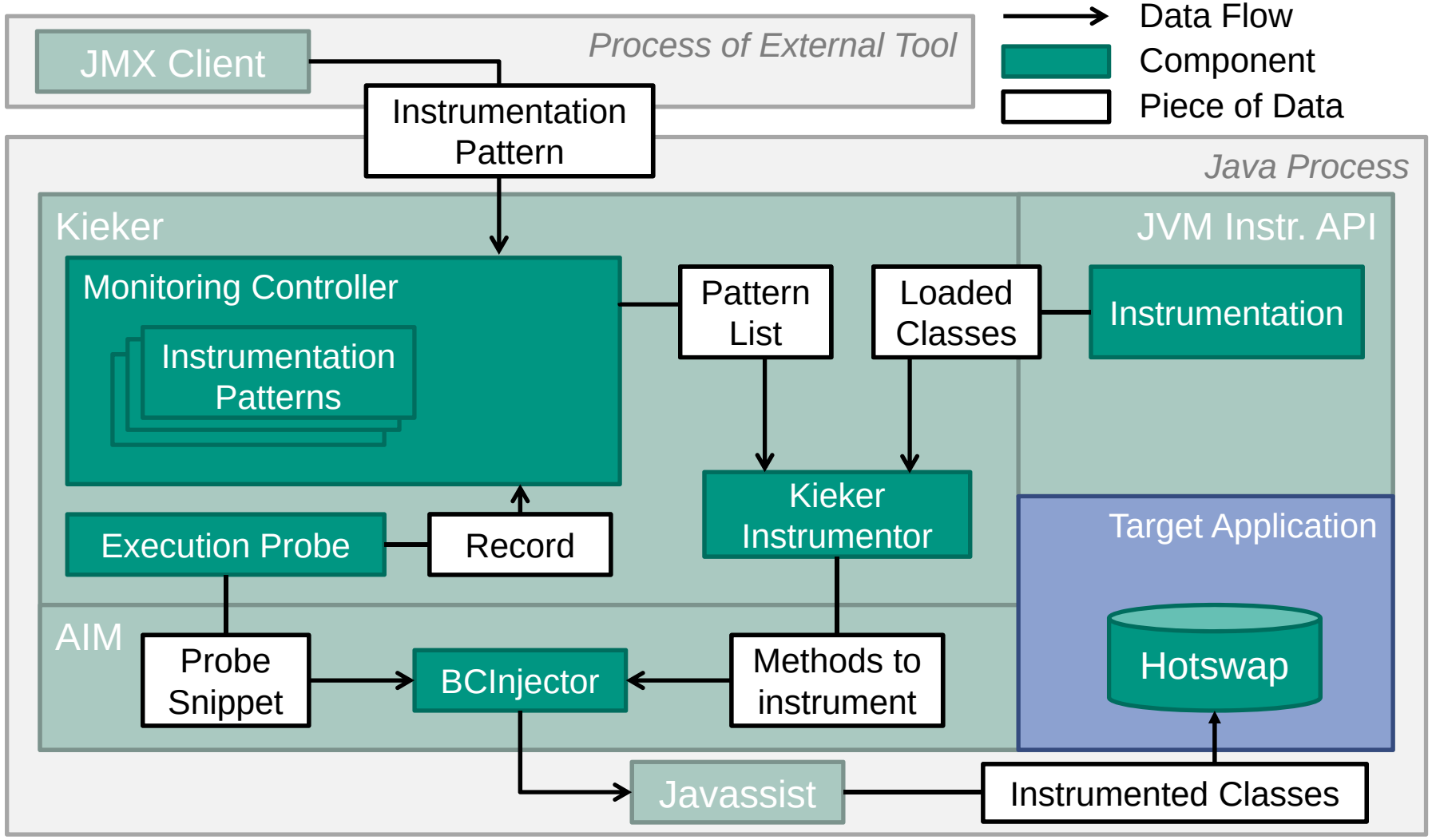


Illustration based on Waller et al. (2012)

AIM Integration in Kieker



Evaluation

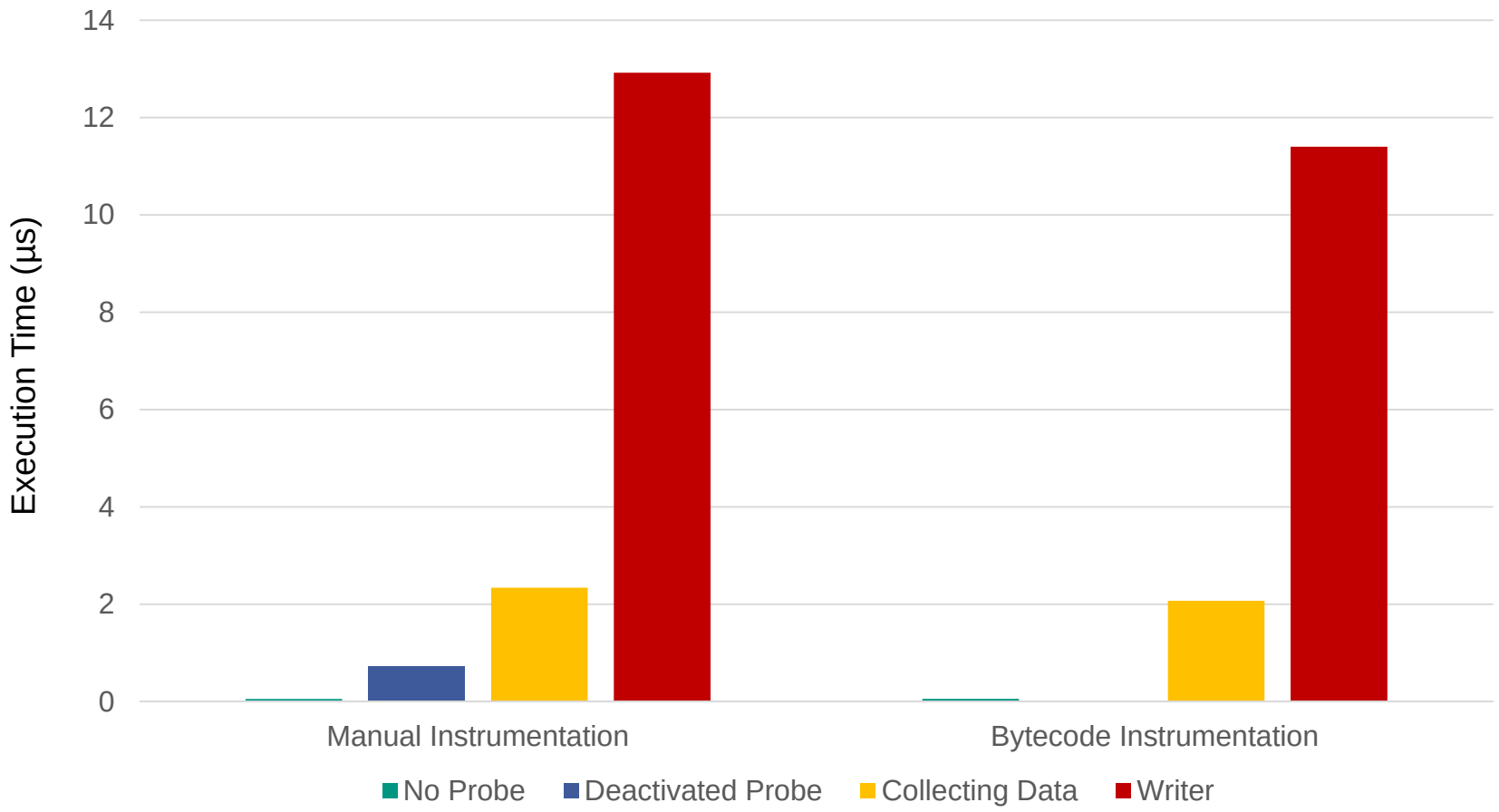
- Bytecode instrumentation vs. Manual

1. Overhead
   - RQ1: How does it fare in terms of performance?
2. Turnaround
   - RQ2a: How does it fare in terms of turnaround time?
   - RQ2b: Does it scale well?

MooBench

3. Lost Transactions
   - RQ3: Is it reliable?

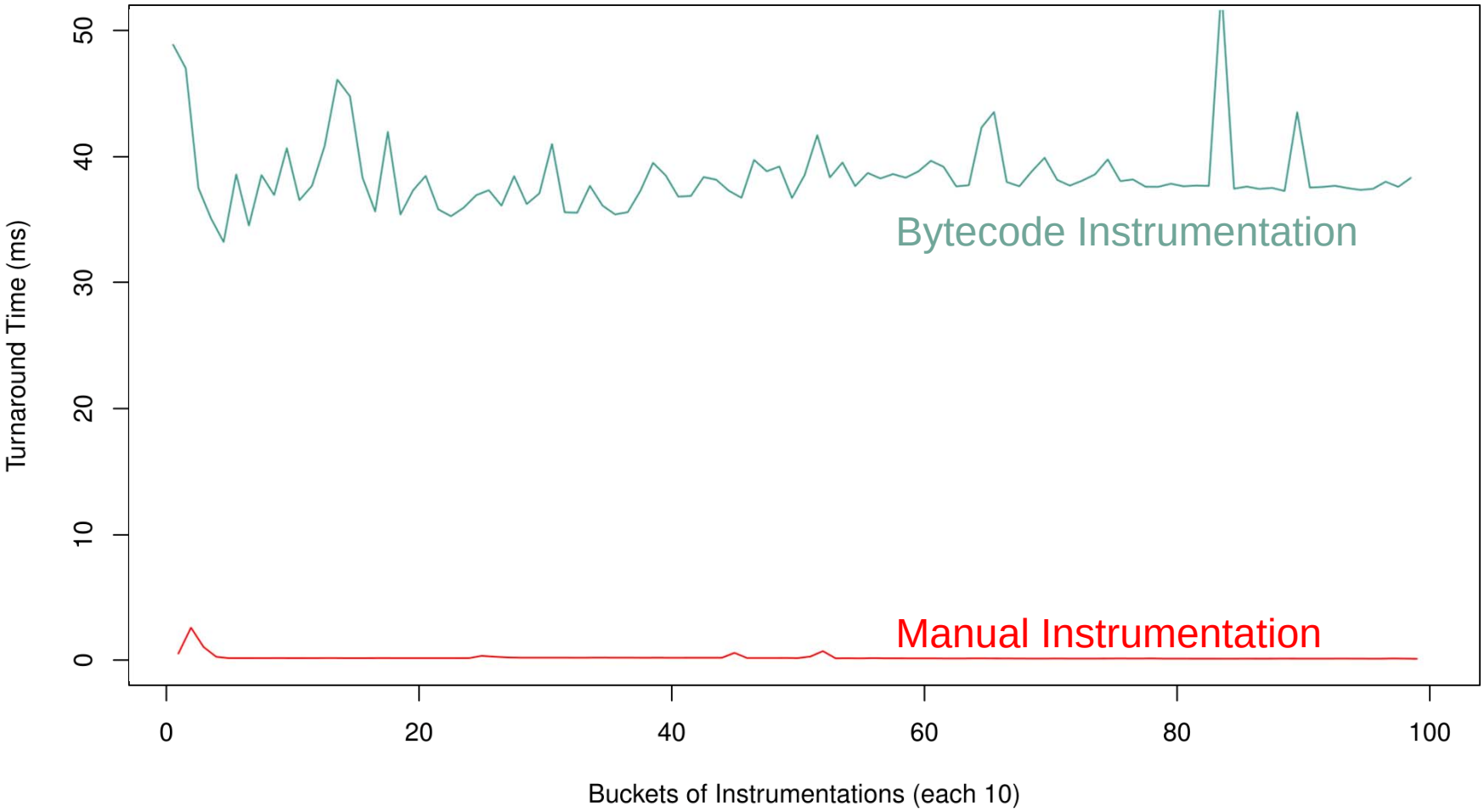
Evaluation 1: Overhead Comparison



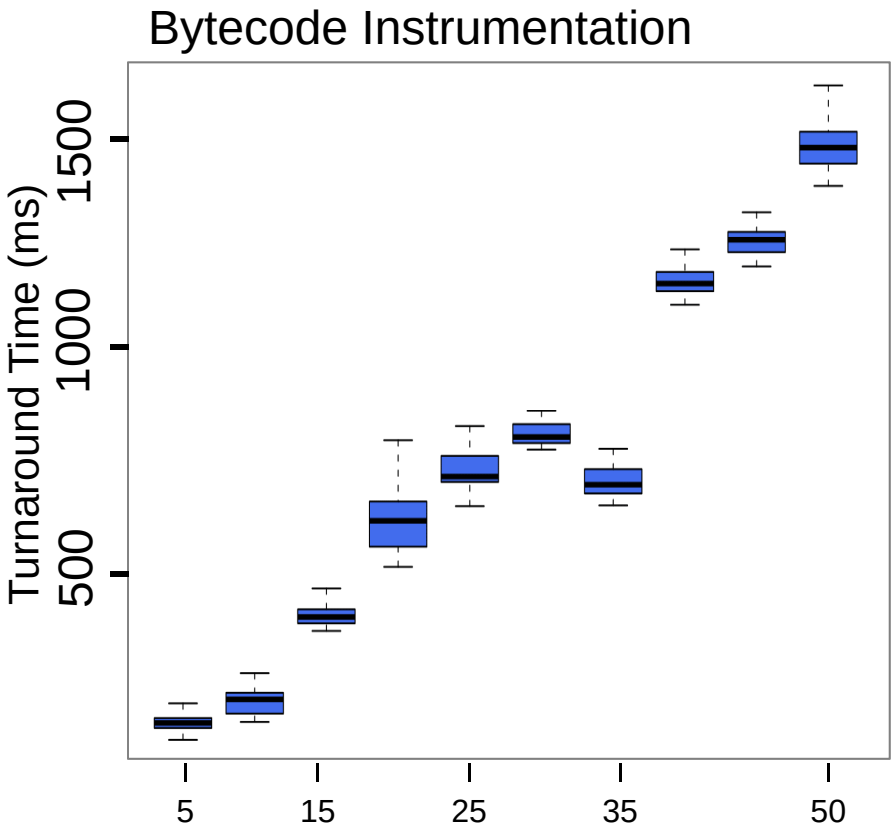
Evaluation 2: Turnaround (Setting)

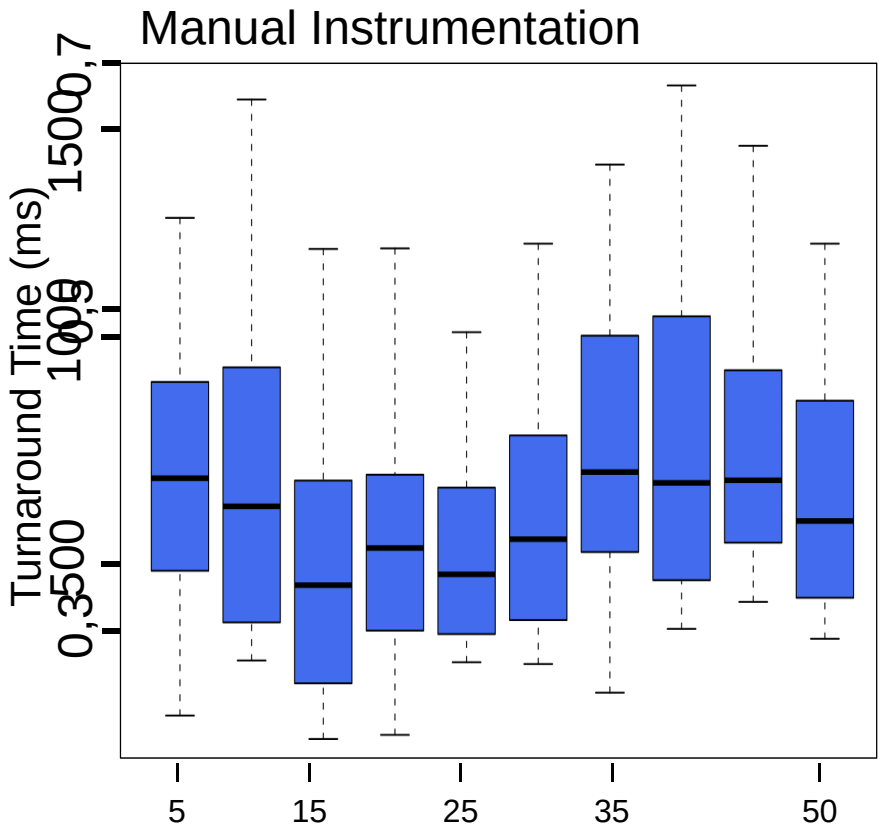
- Bytecode Instrumentation
  - Turnaround = Activating probe + first execution of probe
  - Activating probe invokes BCI
- Manual Instrumentation
  - Turnaround = Activating probe + first execution of probe
  - First execution of probe invokes caching
- Instrumenting 1,000 dummy classes
- Comparison runtime BCI vs. caching

Evaluation 2a: Turnaround Time



Evaluation 2b: Scalability Comparison





# of classes

Conclusion

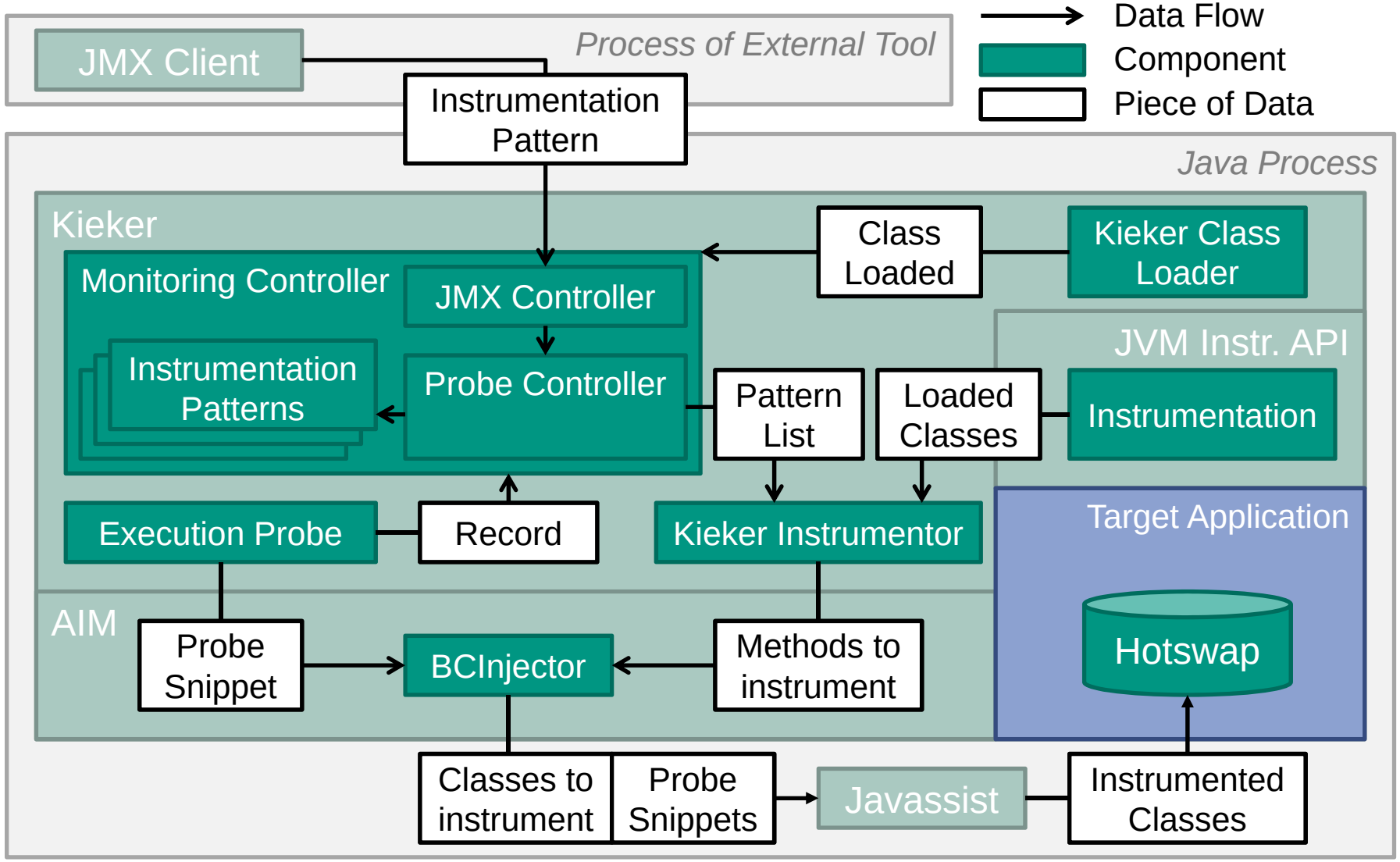
- Core Results
  - Kieker & AIM synergies
  - Small overhead reduction and higher flexibility
  - High turnaround time
    → relevant for production
    → needs optimization

- Future Work
  - Kieker supporting the instrumentation description language
  - Optimize Performance of AIM‘s instrumentation step
  - Usage in bigger context (e.g., performance problem diagnostics)
  - Integration into Kieker release

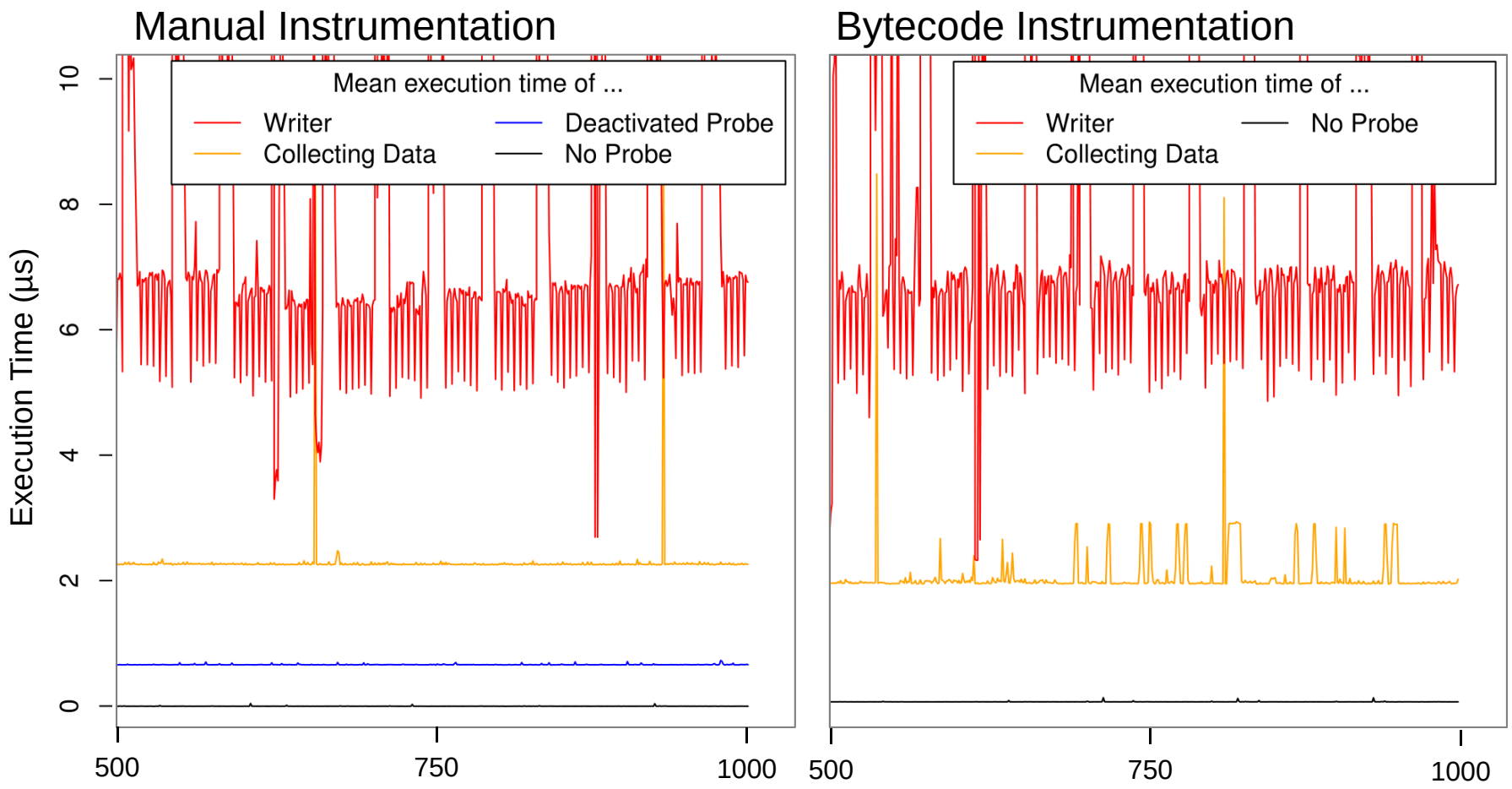
References

- [Dmitriev, 2004]
  M. Dmitriev, “Profiling java applications using code hotswapping and dynamic call graph revelation”
- [Ehlers et al, 2011]
  J. Ehlers, A. van Hoorn, J. Waller, and W. Hasselbring, “Self-adaptive software system monitoring for performance anomaly localization”
- [Moret et al, 2011]
  P. Moret, W. Binder, and E. Tanter, “Polymorphic bytecode instrumentation”
- [Marek et al, 2012]
  L. Marek, A. Villazon, Y. Zheng, D. Ansaloni, W. Binder, and Z. Qi, “Disl: A domain-specific language for bytecode instrumentation”
- [van Hoorn et al, 2009]
  A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst, “Continuous monitoring of software services: Design and application of the kieker framework”

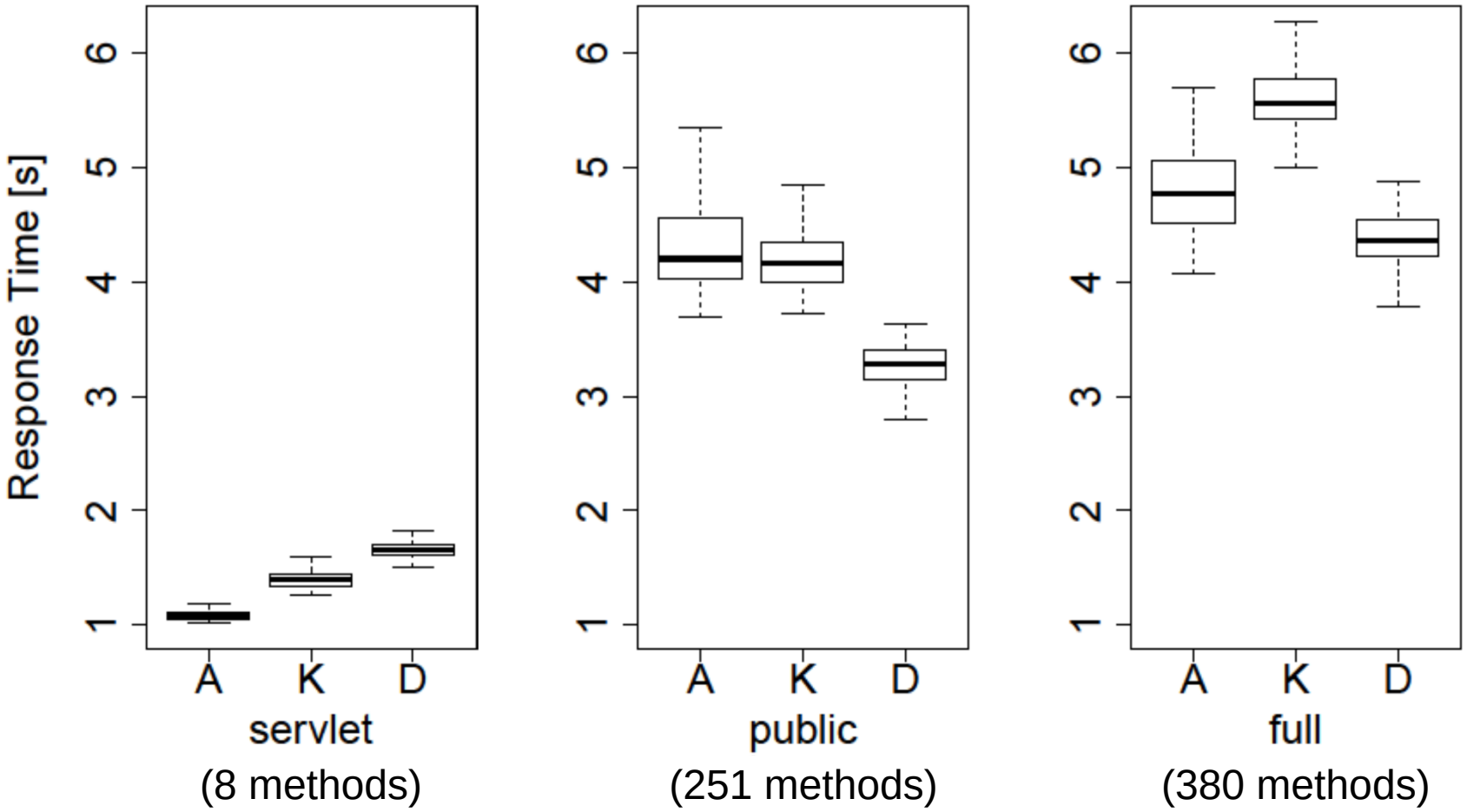
AIM Integration in Kieker Detailed



Evaluation 1: Overhead Comparison

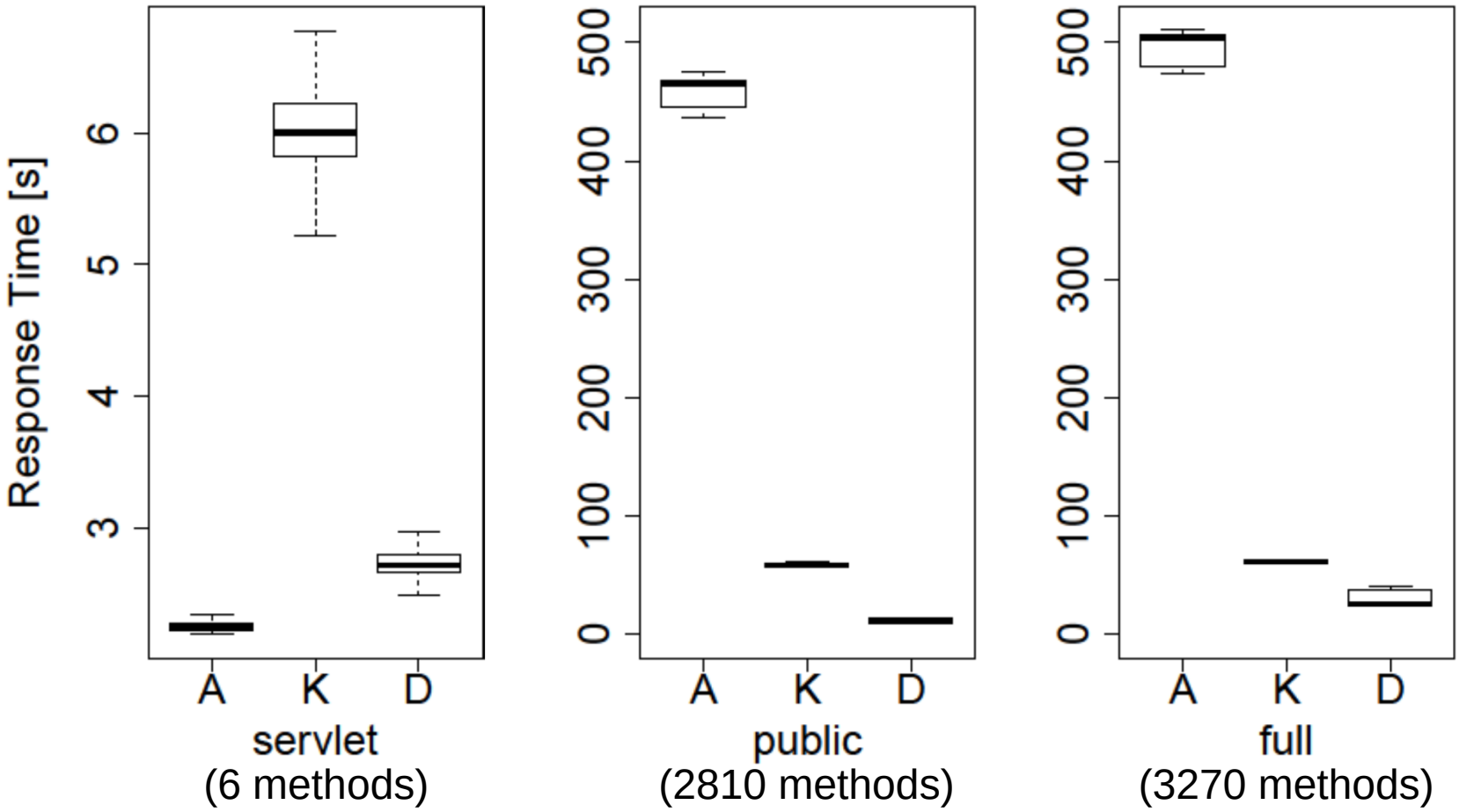


Instrumentation Duration (TPC-W)



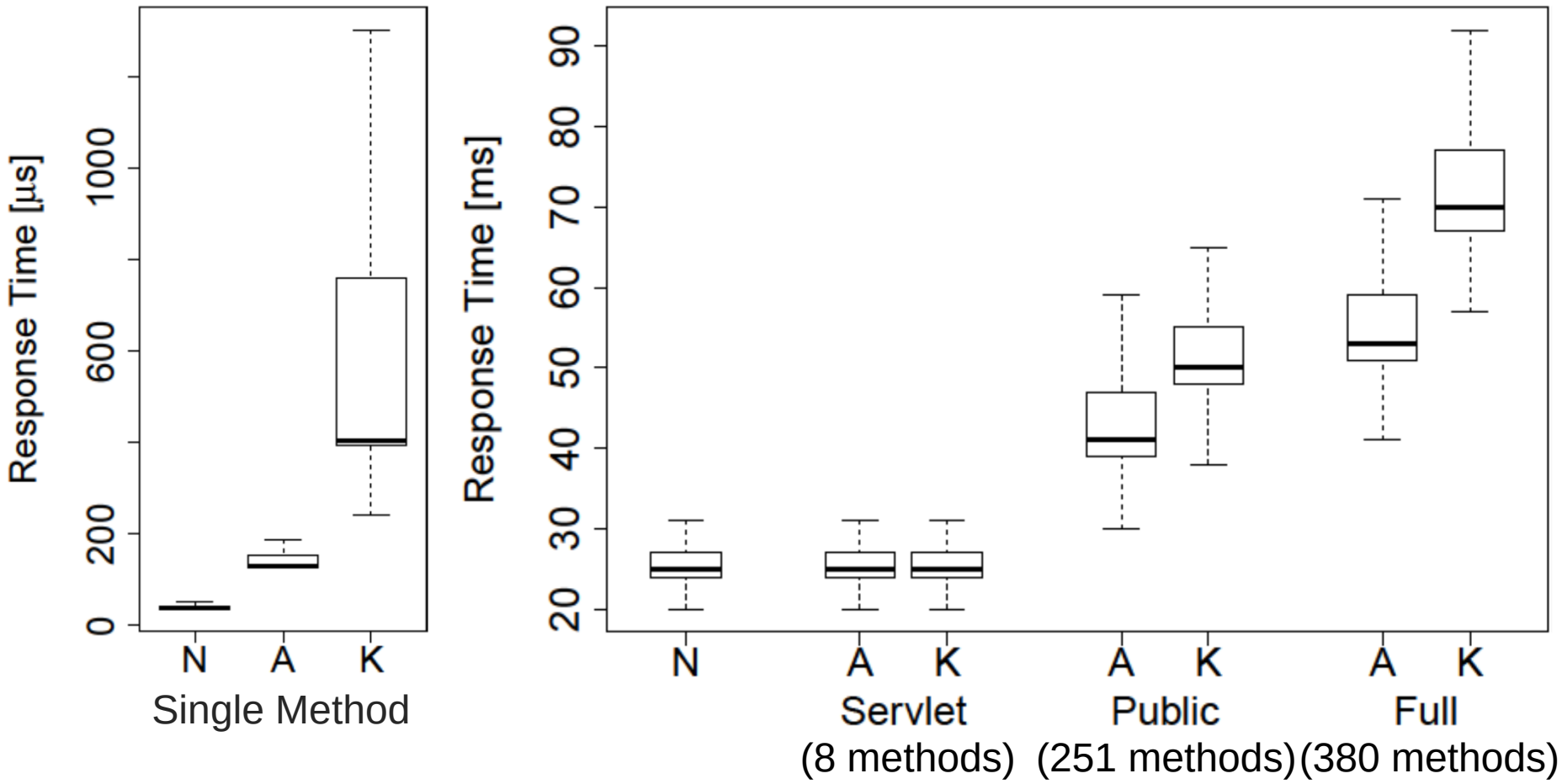
Legend: **A**: Instr. with AIM **K**: Instr. with Kieker **D**: Instr. With DiSL

Instrumentation Duration (Liferay)



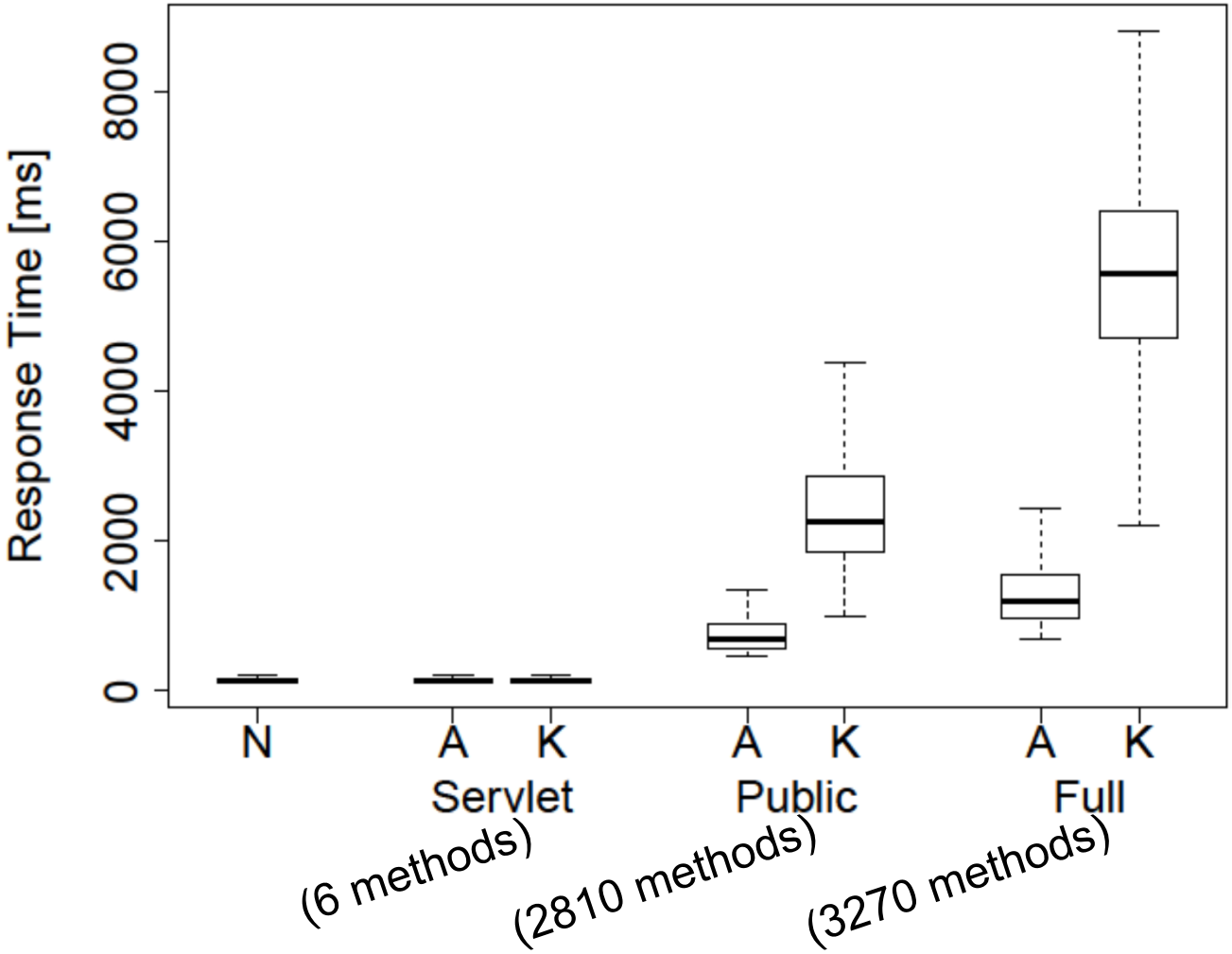
Legend: **A**: Instr. with AIM **K**: Instr. with Kieker **D**: Instr. With DiSL

Overhead (MooBench & TPC-W)



Legend: **N**: No Instr. **A**: Instr. with AIM **K**: Instr. with Kieker

Overhead (Liferay)



Legend: **N**: No Instr. **A**: Instr. with AIM **K**: Instr. with Kieker