Requirements Monitoring

Paul Grünbacher

Christian Doppler Laboratory MEVSS
Institute for Software Systems Engineering
Johannes Kepler University Linz, Austria

http://mevss.jku.at

Very-Large-Scale Software Systems

JOHANNES KEPLER
UNIVERSITY LINZ

Size is just one of the challenges …

- System-of-systems architectures
- Independent elements with different execution environments
- Globally distributed development
- Emerging behavior and properties
- Variability
- Evolutionary development (20+ years)

Requirements Monitoring

JOHANNES KEPLER UNIVERSITY LINZ

"… help determine if the software meets its users' needs …" [Robinson2010]

- Analyze requirements-level properties by continuously analyzing the system in use
- Automate runtime requirements evaluation
  - Interpret low-level software events as contributors to eventual requirements satisfaction or violation
- Address invisibility arising from software complexities and changeability

Requirements Monitoring in Systems of Systems

JOHANNES KEPLER UNIVERSITY LINZ

"[...] requirements **monitors** be installed to **gather and analyze pertinent information** about the system's **run-time** environment. [...] **detect divergences** from our assumptions that adversely affect adherence to requirements."

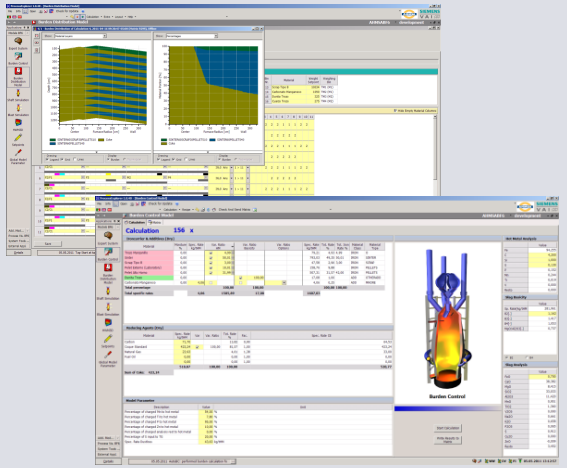
[Fickas and Feather, RE 1995]

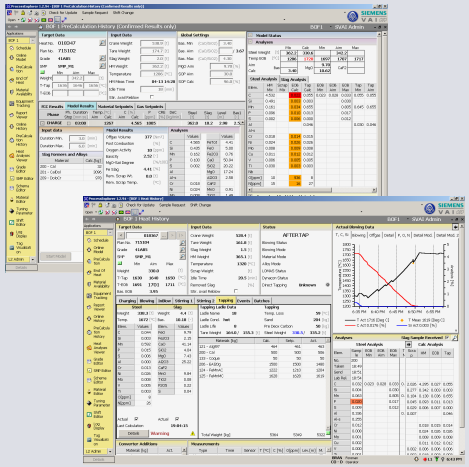
"Most large software systems result from **weaving together** many **independently developed systems**. […] **Requirements monitoring can sound the alert** should these creations fail to meet their obligations."

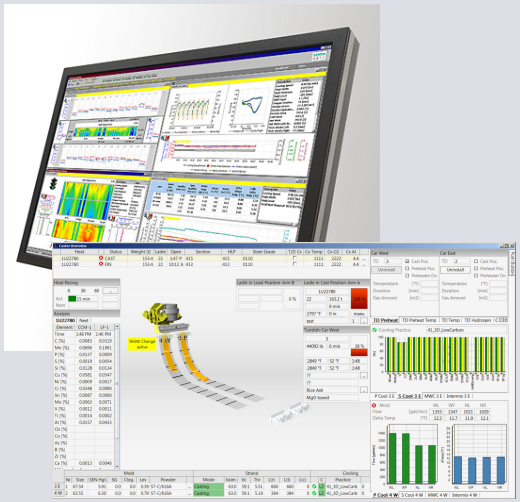
[Robinson, IEEE Software 2010]

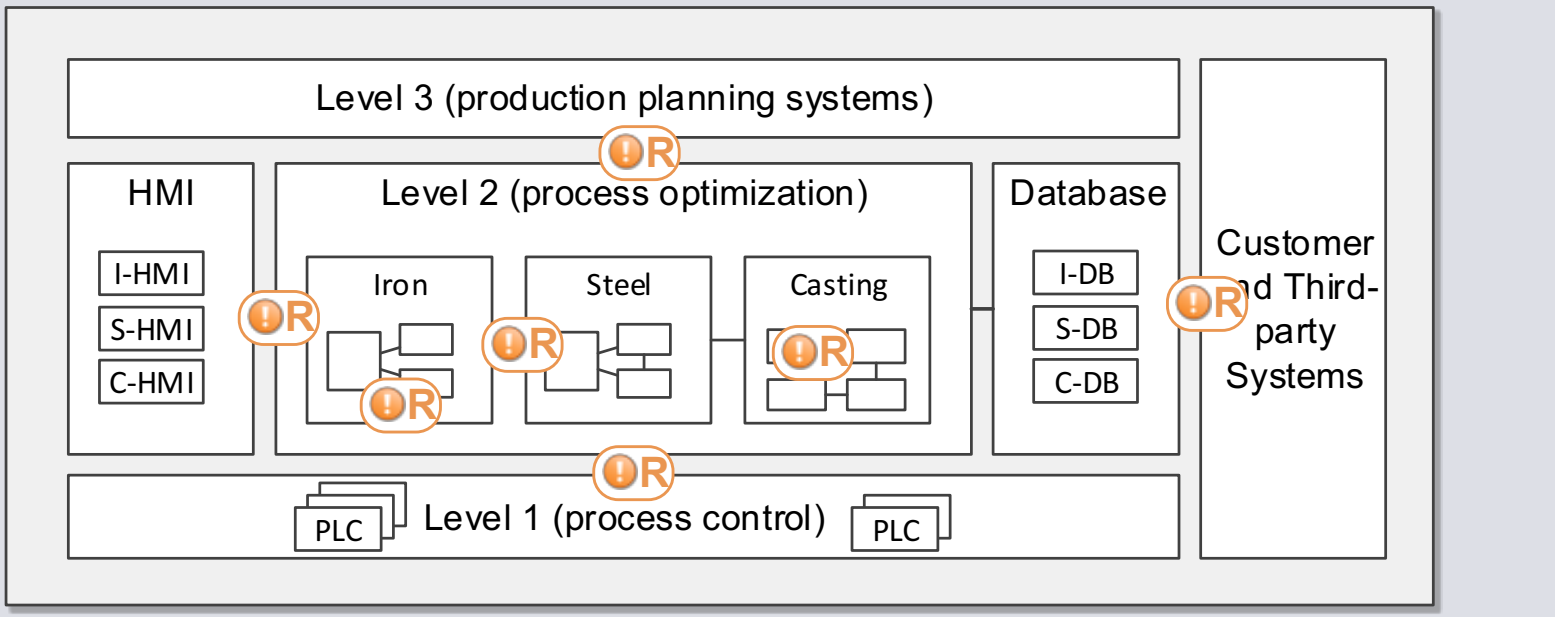
Plant Automation SoS
Ironmaking, Steelmaking, Continuous Casting

PRIMETALS TECHNOLOGIES









Requirements-based Monitoring in SoS

JOHANNES KEPLER UNIVERSITY LINZ

Operationally and managerially independent systems
→ monitoring requirements across different heterogeneous systems

Effects of the systems' interactions difficult to predict
→ monitoring when full behavior emerges during commissioning and operation

Continuous evolution and evolutionary independence
→ checking compliance with requirements after upgrades

Diverse types of requirements at many different levels
→ unified checking framework

Maier 1998, Northrop 2006, Boehm 2006, Caffal and Michael 2005

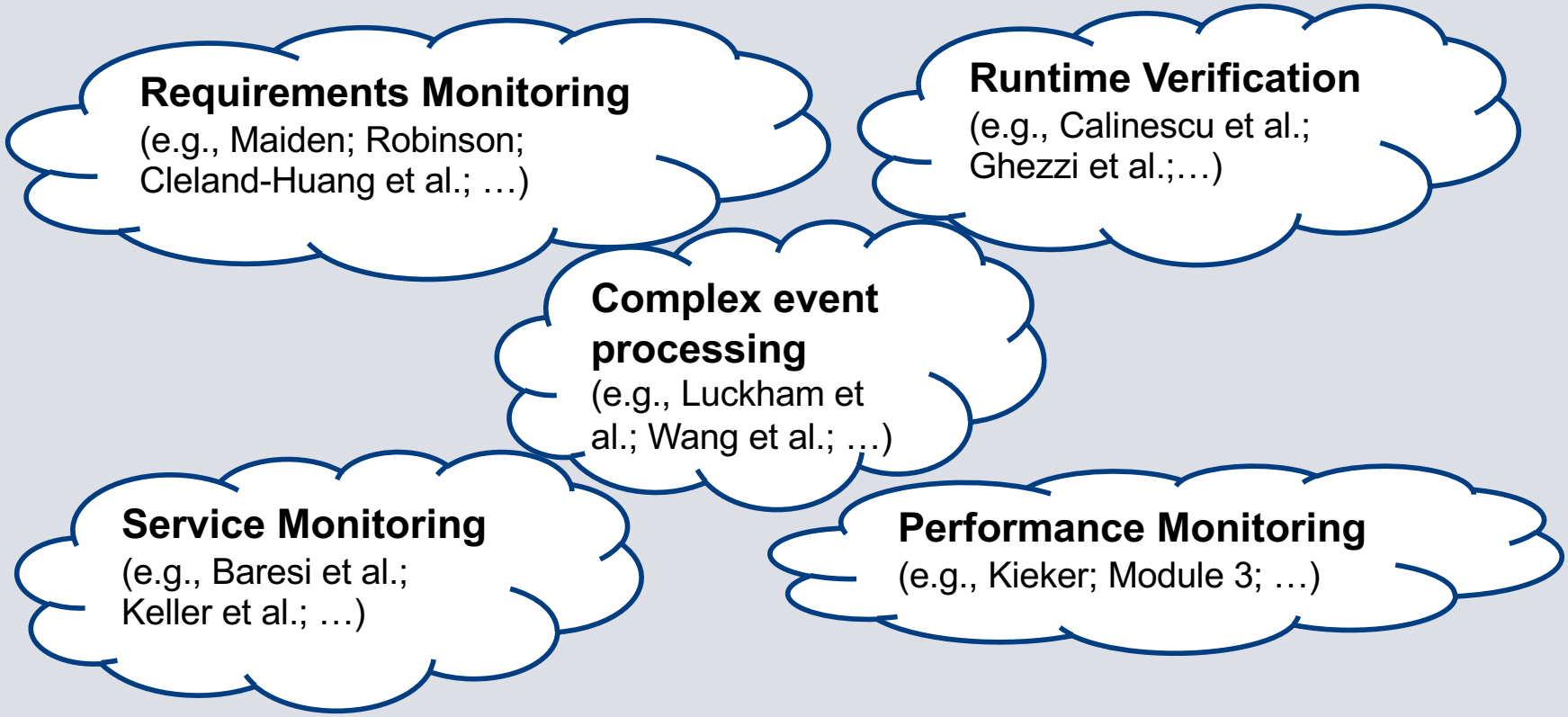
Continuous Change and Requirements Validation

JOHANNES KEPLER UNIVERSITY LINZ

- Requirements monitoring can help determine after **changes** if the software meets its users' needs.
- Validation is no longer idealized as an activity that occurs once. **Incremental development implies incremental validation**.
- Analyzers need to **continuously validate systems** against users' needs.

Related Research Areas

JOHANNES KEPLER UNIVERSITY LINZ



Often focus on specific types of systems and technologies

What is a Monitor?

JOHANNES KEPLER UNIVERSITY LINZ

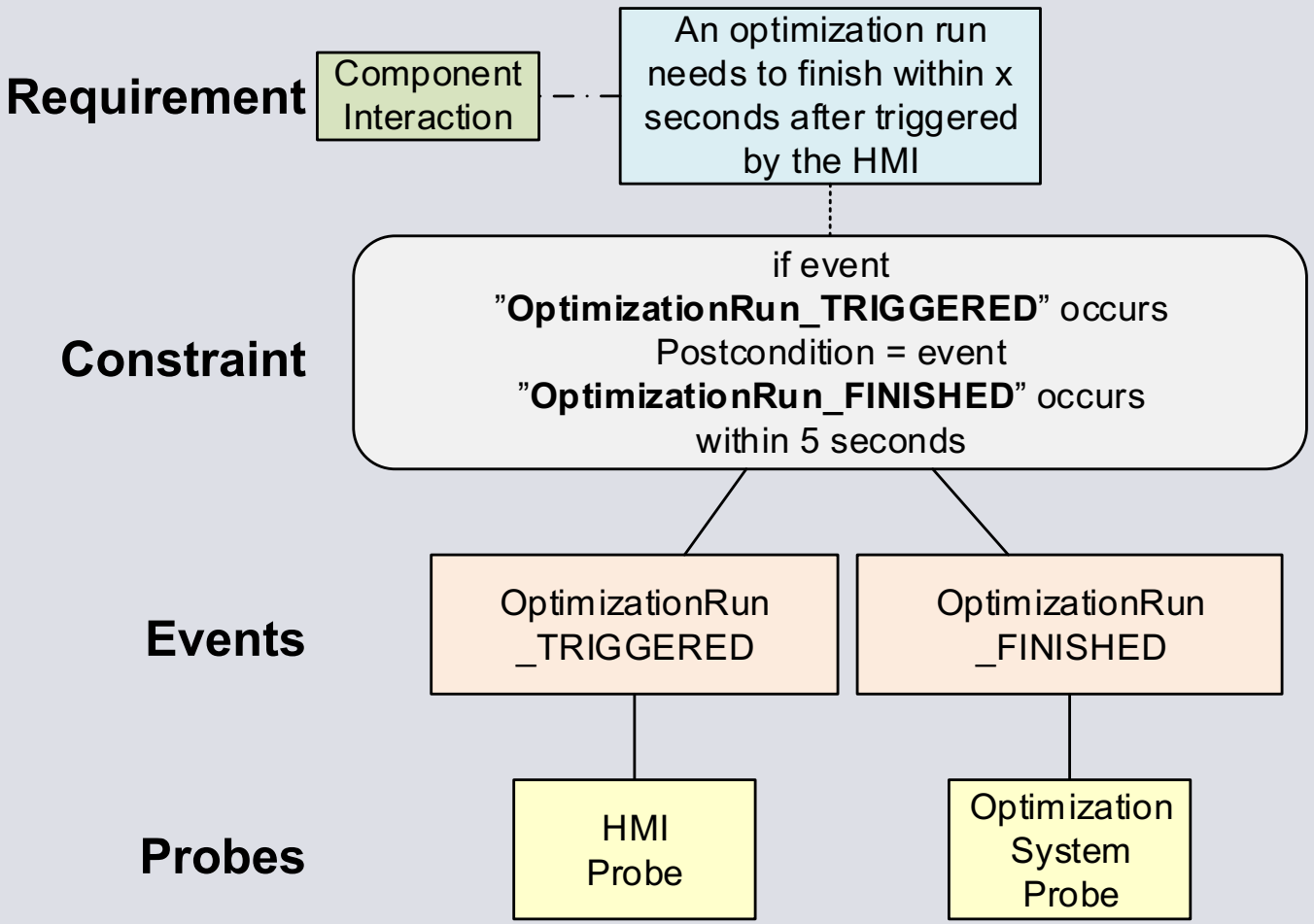
- A software system that observes and analyzes the behavior of another system.
- Determines qualities of interest, i.e., the satisfaction of the target system's requirements
- A function that processes an input data stream to determine the status of requirements.

$$\mathbf{MON(IN_{mon})} \rightarrow \mathit{Sat}\mathbf{(REQ)}$$

Note: In reality the monitored event stream comprises complex objects (XML, JSON) that are produced by event management and logging frameworks (log4j)

Monitoring a PAS Requirement – Challenges

JOHANNES KEPLER UNIVERSITY LINZ



multiple levels and layers of granularity

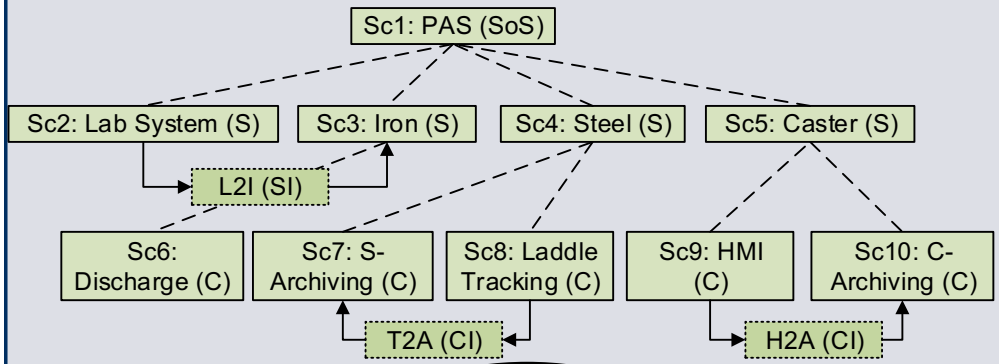
diverse types of constraints

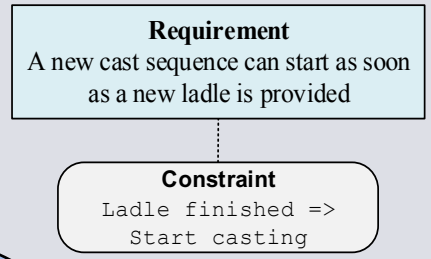
different types of events and data/structures

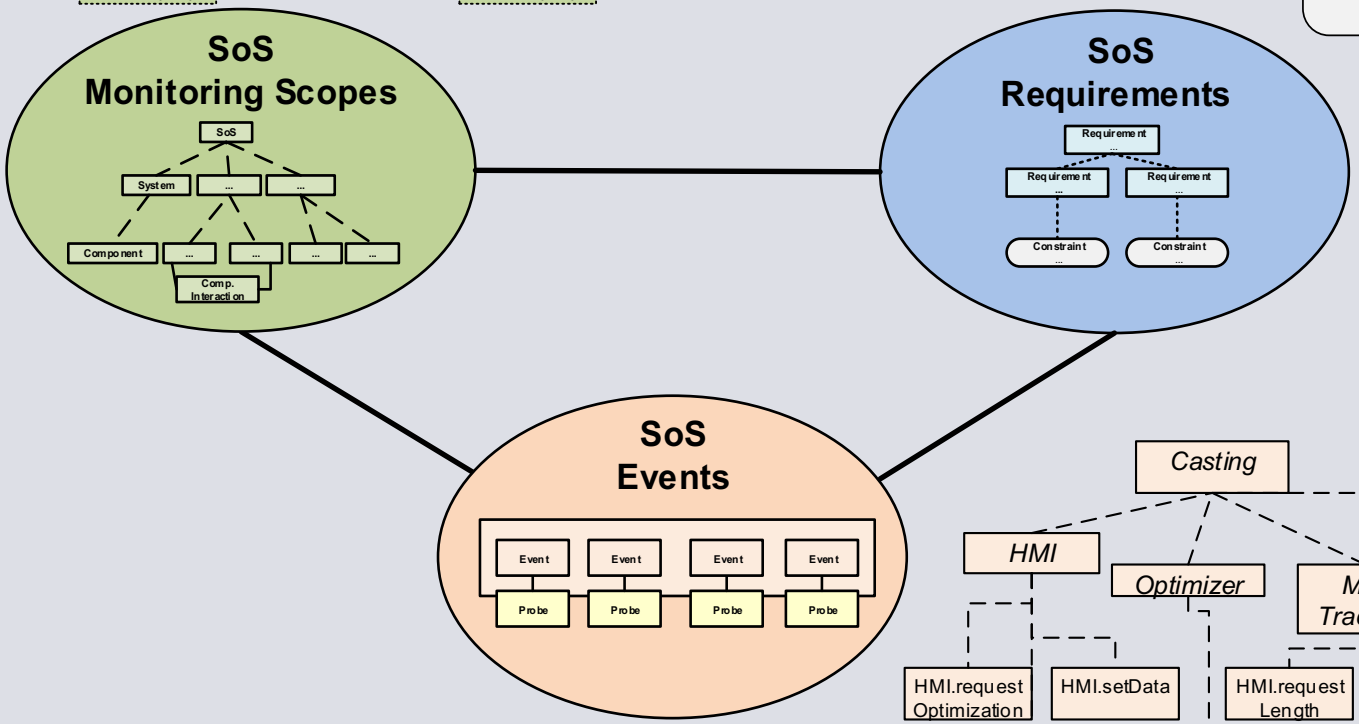
different technologies for instrumentation

REMINDS Requirements Monitoring Model (RE'15)

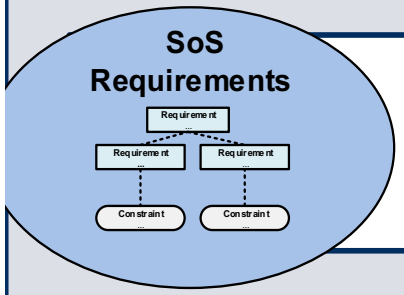
JOHANNES KEPLER UNIVERSITY LINZ







M. Vierhauser, R. Rabiser, P. Grünbacher, B. Aumayr, "**A Requirements Monitoring Model for Systems of Systems**". *Requirements Engineering Conference, 2015.*



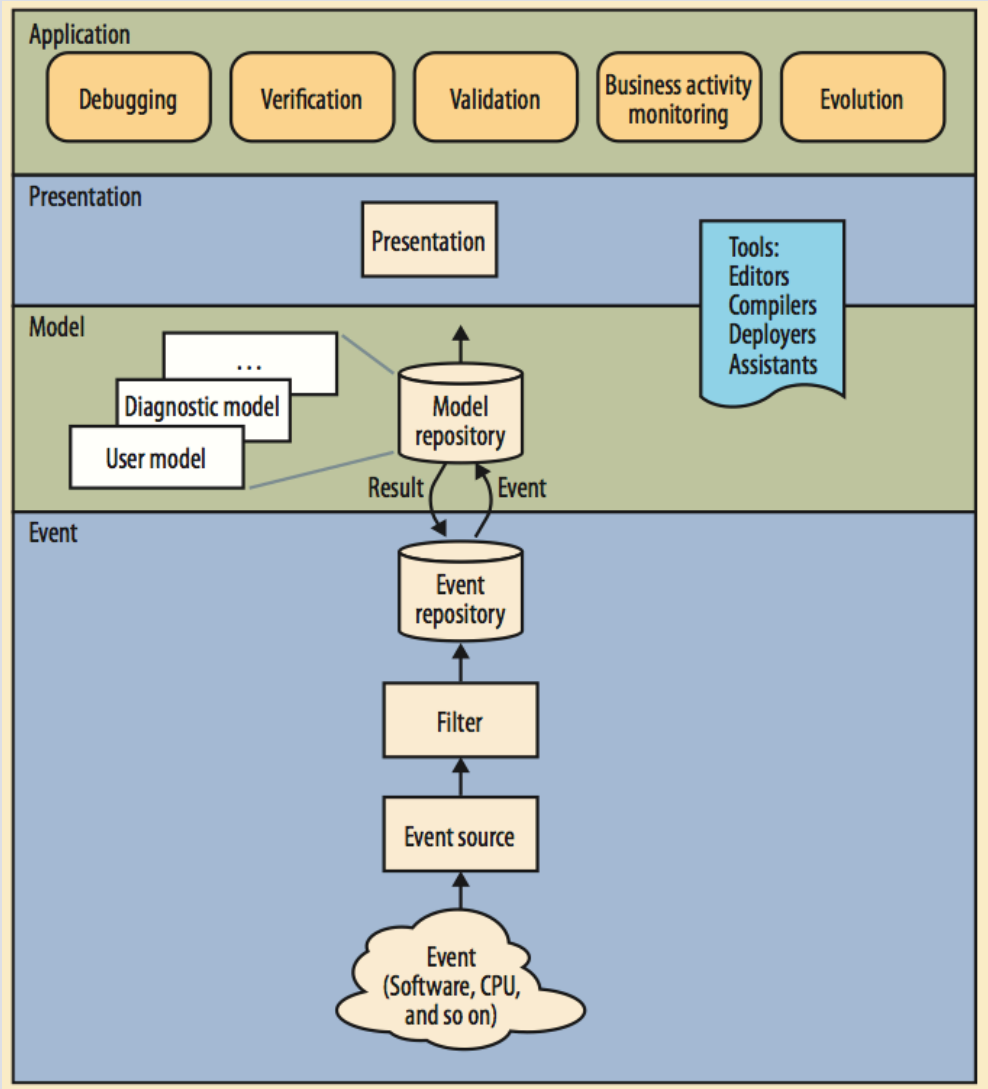
SoS Monitoring Challenges

JOHANNES KEPLER
UNIVERSITY LINZ

- **Constraint diversity**
- **Incremental definition**
- **Runtime management of constraints**
- **End-user support**

Robinson's General 4-Layer Framework

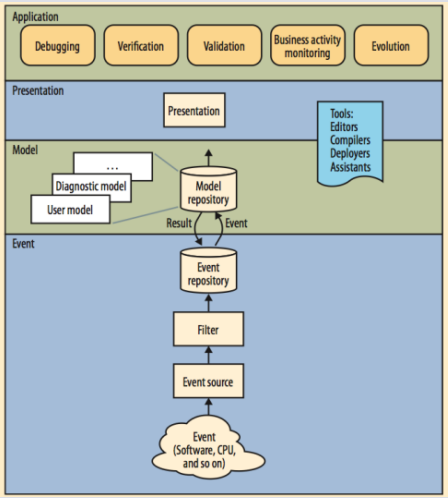
JOHANNES KEPLER UNIVERSITY LINZ



Event Layer

JOHANNES KEPLER UNIVERSITY LINZ

- Services for event acquisition, transportation, filtering, and storage
- An event is a description of a **significant action in time** typically leading to a state change
  - Completion of a method call
  - CPU reaching 90%
  - Shipment arriving at destination
  - …



Event-based Monitoring

JOHANNES KEPLER UNIVERSITY LINZ

Optimiz

Optimizer_finished

Cutting_completed

cutlength=4.75

Optimizer_writeResult


```
  "CurrentHeat": 0,
  "CurrentHeatWeight": 0,
  "NextHeatList": [0],
  "NextHeatWeightList": [0],
  "TopHeatOfTundish": 0,
  "TundishInCastPosition": false,
  "TundishWeight": 0
 }},
 "AltLengthInfoContainer": {"Map": {}},
 "CutListContainer": {
  "CutListsNearMold": {
   "1": {
    "CutList": [],
    "ExtraUncutProducts": []
   },
   "2": {
    "CutList": [{
     "CalculatedWeight":
1414.1809725250364,
     "CutCode": {"Name": "TAIL"},
     "Cutting": false,
     "FormatId": 2,
     "HeatId": 250,
     "IsPartialCut": false,
     "Length": 0.5046733009549911,
     "LengthShrinkage": 1.011,
```

Model Layer

JOHANNES KEPLER UNIVERSITY LINZ

Interpreting events using models

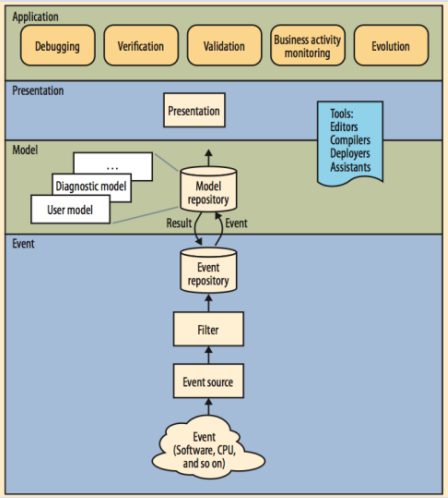
Analyzing software behavior

- states and potential responses
- important to provide guidance for the software's evolution

Interpreting user behavior

- user's state and likely responses

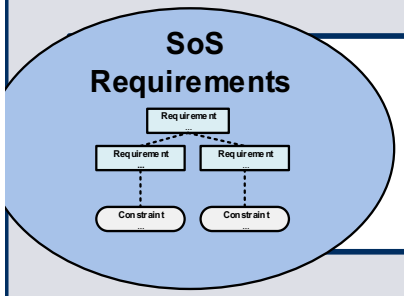
  important to improve user guidance



Models for Analyses

JOHANNES KEPLER UNIVERSITY LINZ

- Functional models defining the allowed states and transitions
- Goal models
- Antimodels defining undesirable behavior (e.g., attacks)
- Quality of Service models
- Discovery Models (e.g., data mining to learn user behavior)



SoS Requirements

JOHANNES KEPLER
UNIVERSITY LINZ

Tap start must be (optionally) followed by slag start, which must be followed by tap end: A - B(optional) - C

When a ladle is full (net ladle weight > limit) and a tap is running, a new ladle must be started.

Sample defined -> optimization run must start within x seconds

An optimization run needs to finish within x seconds after triggered by another component

Skip cycle: e.g. Skip waits at stockhouse – Skip is filled at stockhouse – skip moves to top hopper – skip fills into top hopper – skip moves to stock house

TAILING -> isSetPointValid

An optimization run needs to finish within x seconds after triggered by another component

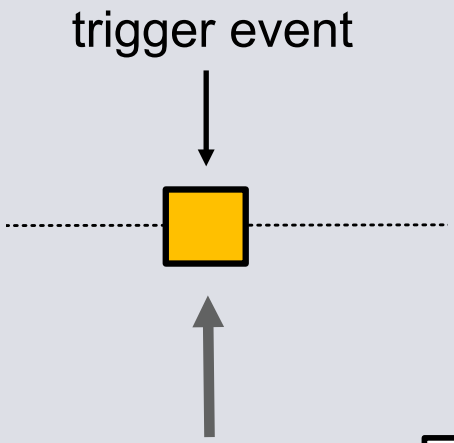
When the tundish car moves to the casting position the ladle must have been mounted within the last 30 seconds

The cutlength must not be > maxtransportable and not < mintransportable

If property dummyBarFeedingMode = BOTTOM StrandStatuses AtTCM.Casting must be false in OptiDataContainer

Types of Constraints

JOHANNES KEPLER UNIVERSITY LINZ



The cutlength must not be > maxtransportable

When the tundish car moves to the casting position the ladle must have been mounted within

An optimization run needs to finish within 10 seconds and perform actions in a specific order

Data Conditions

Past Occurrence

Future Occurrence

```
trigger =
 if event "Cutting_completed" occurs
condition =
```

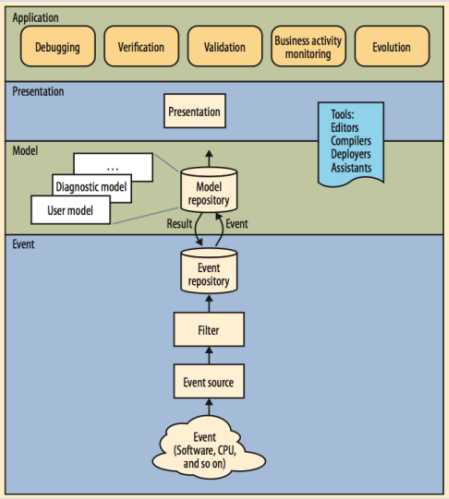
```
trigger =
```

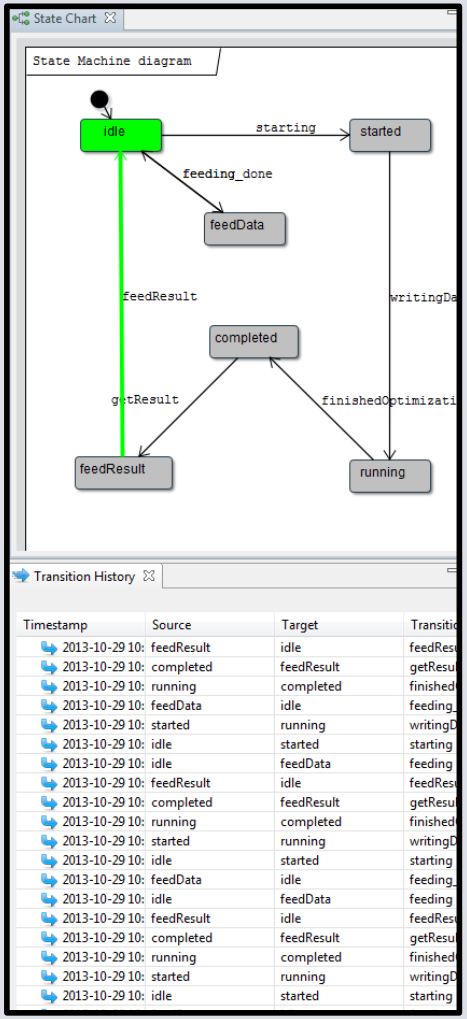
```
trigger =
   if event "Tundish.carArrivedCastPosition" occurs
condition =
   event "Tundish.ladleArrived"
has occurred in the last 30 seconds
```

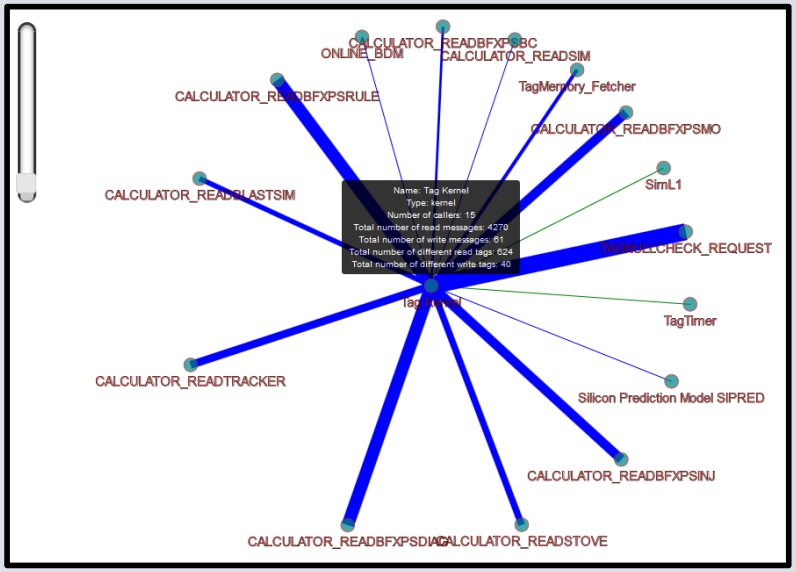
```
  "Optimizer.writeOptiData",
  "Optimizer.optimize_FINISHED",
  "Optimizer.getOptimizationResult",
  "Cutting.feedOptimizationResult"
 occur consecutively within 10 seconds
```

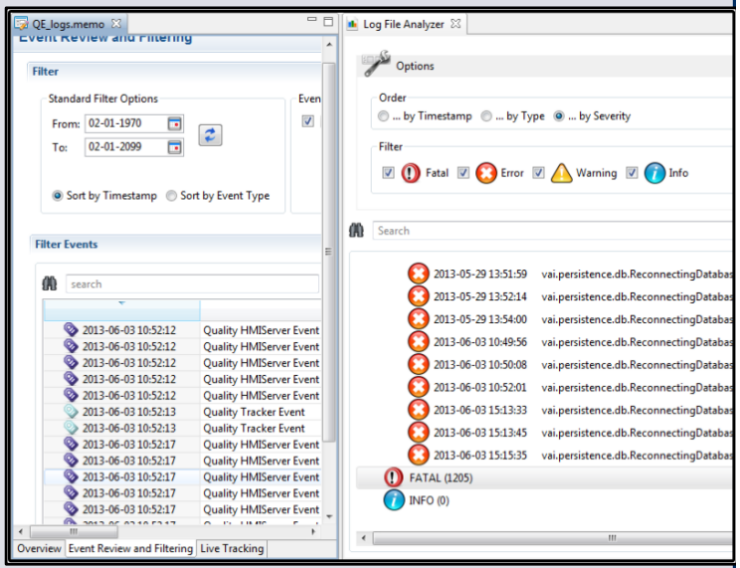
Views Layer

- Services for presenting analyses to developers
- Example: Event data aggregation & visualization





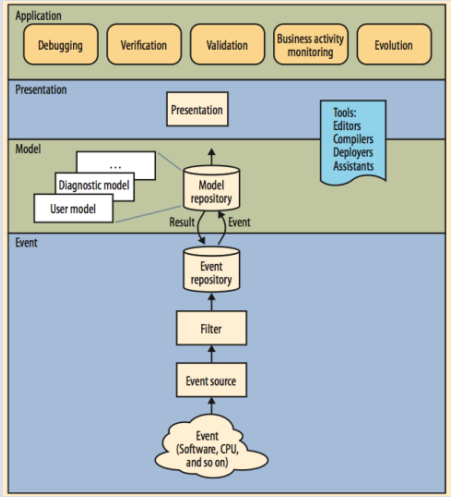




Application Layer

JOHANNES KEPLER UNIVERSITY LINZ

- Application-specific services to present, control, and modify the target system
- Debugging, verification, business activity monitoring, evolution
- Examples
  - Evolution Support: Compare results of multiple simulation runs, e.g., before and after an upgrade
  - Capture & Replay: Use collected production data for offline system simulations



Industrial SoS Monitoring Applications

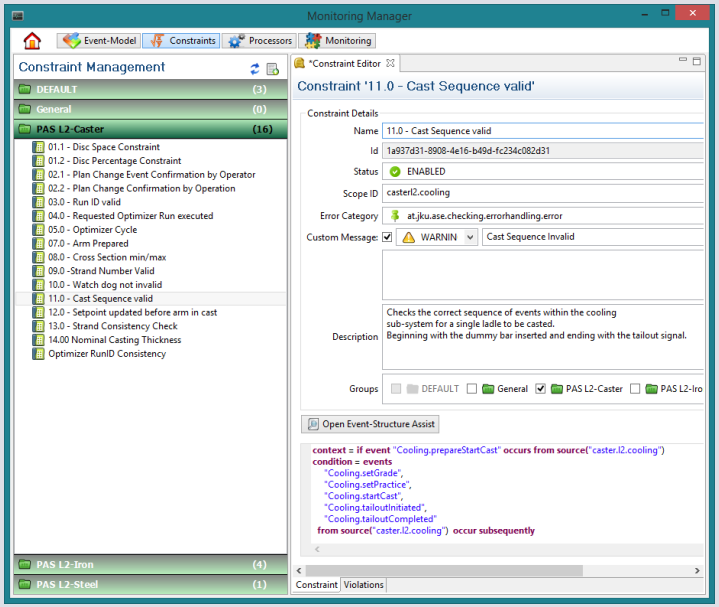
JOHANNES KEPLER UNIVERSITY LINZ

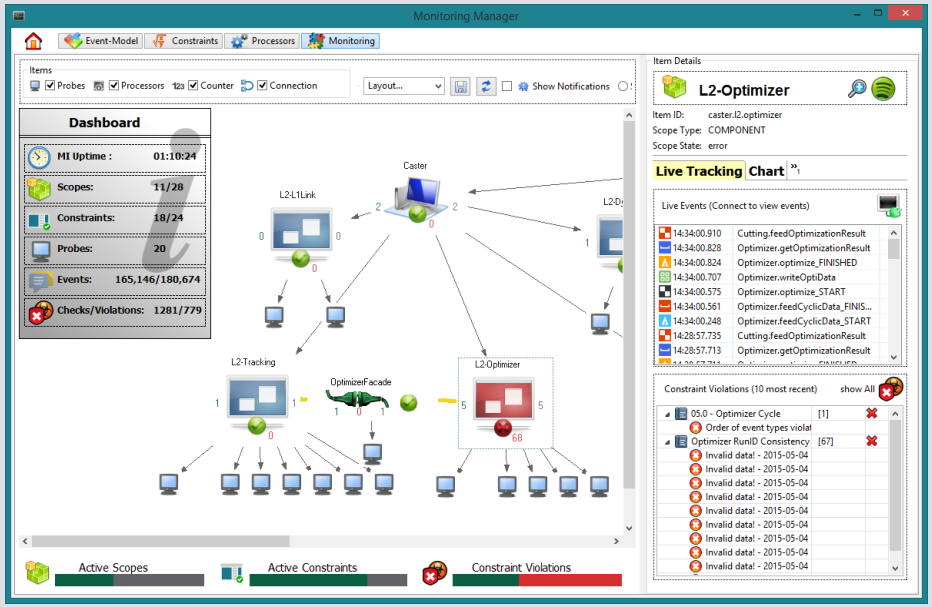
- **Continuous monitoring** and **cross-system analyses** during customer acceptance tests
- **Capture and replay** to facilitate **offline diagnosis** across multiple systems
- **Capture and compare** after SoS evolution to ease **anomaly** detection

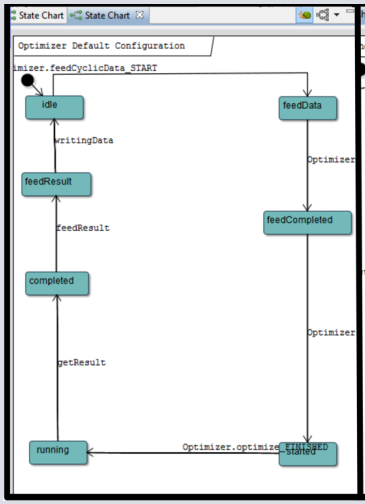
Existing Monitoring Frameworks

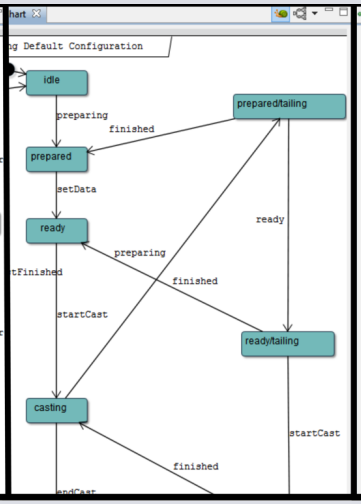
JOHANNES KEPLER UNIVERSITY LINZ

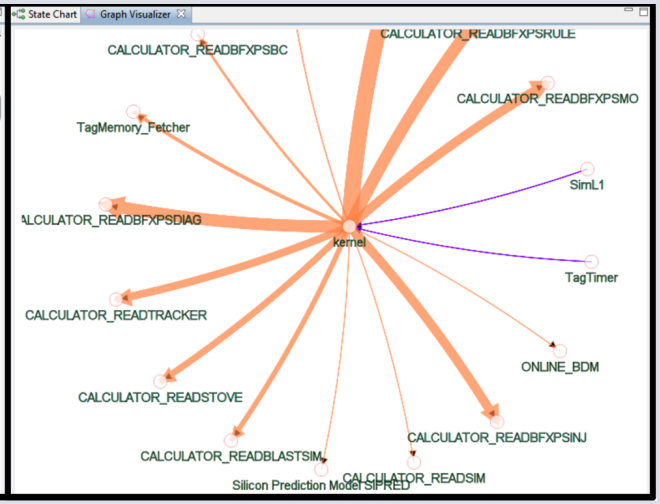
- Kieker
- ReqMon
- Spass Meter
- REMINDS Framework (CDL MEVSS)
- ...


Monitoring Manager
Event-Model
Constraints
Processors
Monitoring
Constraint Management
DEFAULT (3)
General (0)
PAS L2-Caster (16)
01.1 - Disc Space Constraint
01.2 - Disc Percentage Constraint
02.1 - Plan Change Event Confirmation by Operator
02.2 - Plan Change Confirmation by Operation
03.0 - Run ID valid
04.0 - Requested Optimizer Run executed
05.0 - Optimizer Cycle
07.0 - Arm Prepared
08.0 - Cross Section min/max
09.0 - Strand Number Valid
10.0 - Watch dog not invalid
11.0 - Cast Sequence valid
12.0 - Setpoint updated before arm in cast
13.0 - Strand Consistency Check
14.00 Nominal Casting Thickness
Optimizer RunID Consistency
PAS L2-Iron (4)
PAS L2-Steel (1)
*Constraint Editor
Constraint '11.0 - Cast Sequence valid'
Constraint Details
Name 11.0 - Cast Sequence valid
Id 1a937d31-8908-4e16-b49d-fc234c082d31
Status ENABLED
Scope ID casterl2.cooling
Error Category at.jku.ase.checking.errorhandling.error
Custom Message: WARNIN Cast Sequence Invalid
Description
Checks the correct sequence of events within the cooling
sub-system for a single ladle to be casted.
Beginning with the dummy bar inserted and ending with the tailout signal.
Groups DEFAULT General PAS L2-Caster PAS L2-Iro
Open Event-Structure Assist
context = if event "Cooling.prepareStartCast" occurs from source("caster.l2.cooling")
condition = events
"Cooling.setGrade",
"Cooling.setPractice",
"Cooling.startCast",
"Cooling.tailoutInitiated",
"Cooling.tailoutCompleted"
from source("caster.l2.cooling") occur subsequently
Constraint
Violations


Monitoring Manager
Event-Model
Constraints
Processors
Monitoring
Items
Probes
Processors
Counter
Connection
Layout...
Show Notifications
Dashboard
MI Uptime : 01:10:24
Scopes: 11/28
Constraints: 18/24
Probes: 20
Events: 165,146/180,674
Checks/Violations: 1281/779
Caster
L2-L1Link
L2-Tracking
OptimizerFacade
L2-Optimizer
Active Scopes
Active Constraints
Constraint Violations
Item Details
L2-Optimizer
Item ID: caster.l2.optimizer
Scope Type: COMPONENT
Scope State: error
Live Tracking
Chart
Live Events (Connect to view events)
14:34:00.910 Cutting.feedOptimizationResult
14:34:00.828 Optimizer.getOptimizationResult
14:34:00.824 Optimizer.optimize_FINISHED
14:34:00.707 Optimizer.writeOptiData
14:34:00.575 Optimizer.optimize_START
14:34:00.561 Optimizer.feedCyclicData_FINIS...
14:34:00.248 Optimizer.feedCyclicData_START
14:28:57.735 Cutting.feedOptimizationResult
14:28:57.713 Optimizer.getOptimizationResult
Constraint Violations (10 most recent)
show All
05.0 - Optimizer Cycle [1]
Order of event types violat
Optimizer RunID Consistency [67]
Invalid data! - 2015-05-04

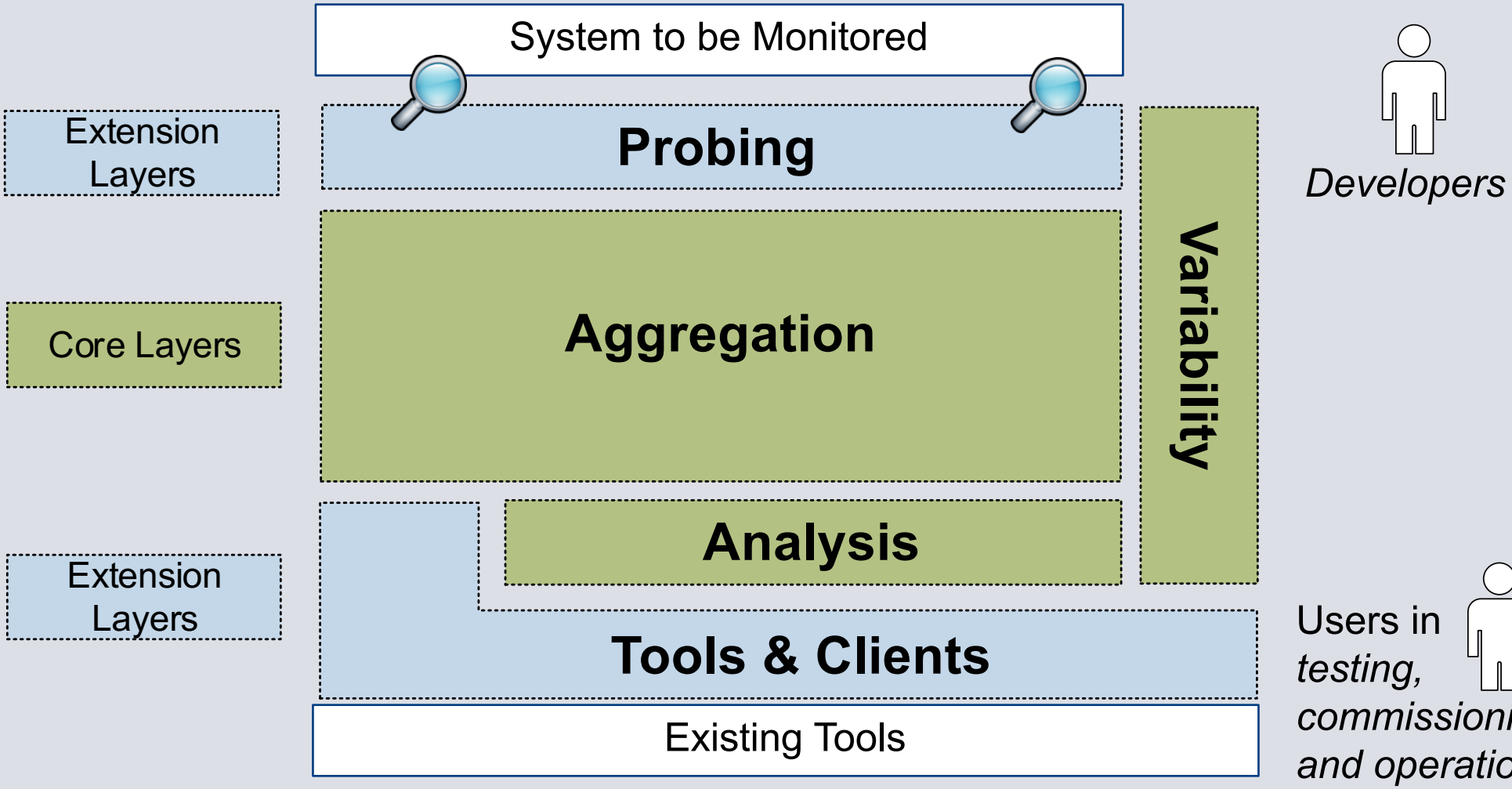

State Chart
State Chart
Optimizer Default Configuration
idle
feedData
writingData
feedResult
feedCompleted
completed
getResult
running
Optimizer.optimize_STARTED


Default Configuration
idle
preparing
finished
prepared/tailing
prepared
setData
ready
preparing
finished
ready
startCast
readyTailing
casting
finished
startCast


State Chart
Graph Visualizer
CALCULATOR_READBFXPSRULE
CALCULATOR_READBFXPSBC
CALCULATOR_READBFXPSMO
TagMemory_Fetcher
SimL1
ALCULATOR_READBFXPSDIAG
kernel
TagTimer
CALCULATOR_READTRACKER
ONLINE_BDM
CALCULATOR_READSTOVE
CALCULATOR_READBFXPSINJ
CALCULATOR_READBLASTSIM
CALCULATOR_READSIM
Silicon Prediction Model SIPRED

ReMinds Framework Architecture

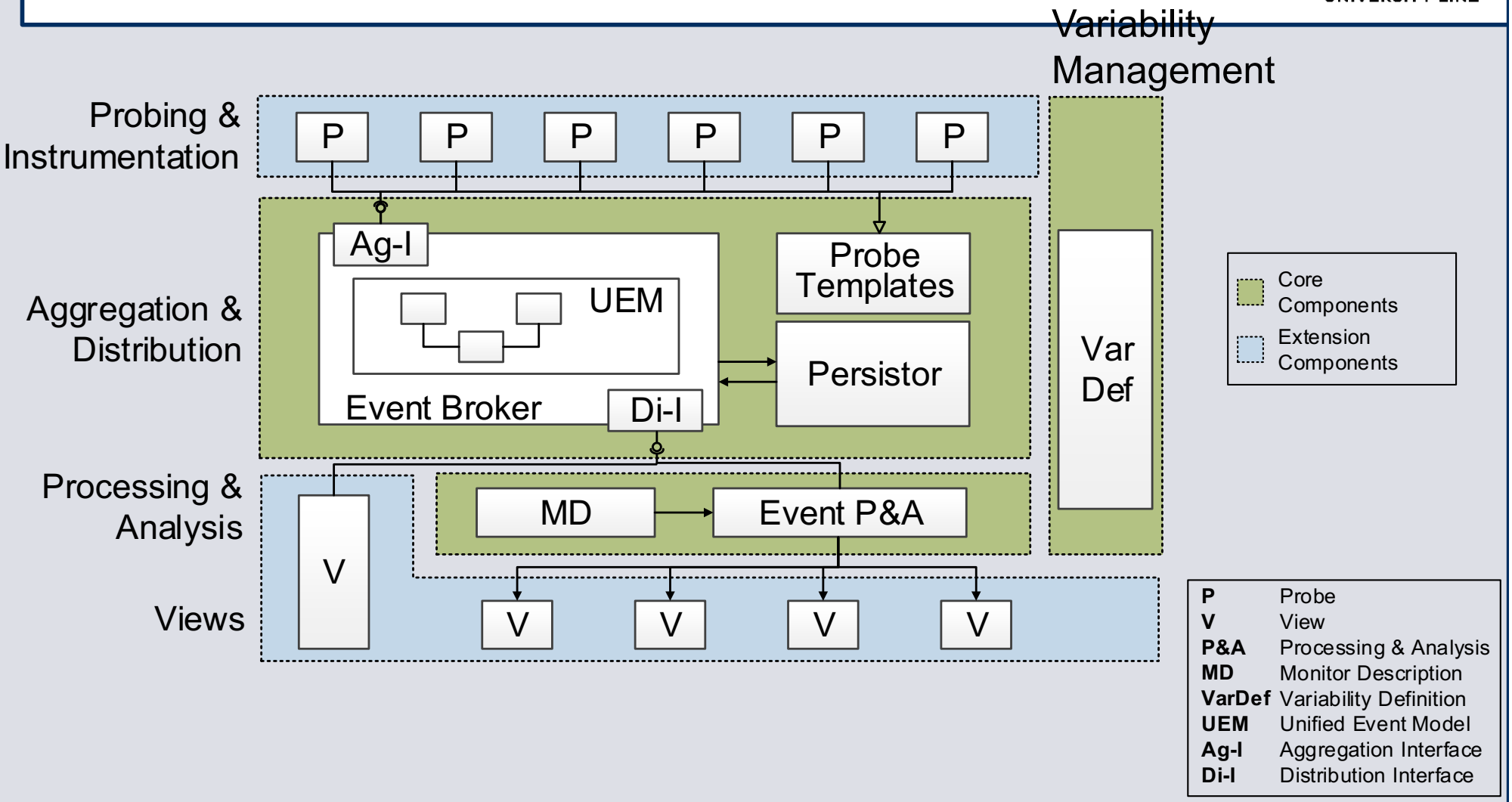
JOHANNES KEPLER UNIVERSITY LINZ



Vierhauser, Rabiser, Grünbacher, Danner, Wallner, Zeisel: A Flexible Framework for Runtime Monitoring of System-of-Systems Architectures, WICSA 2014

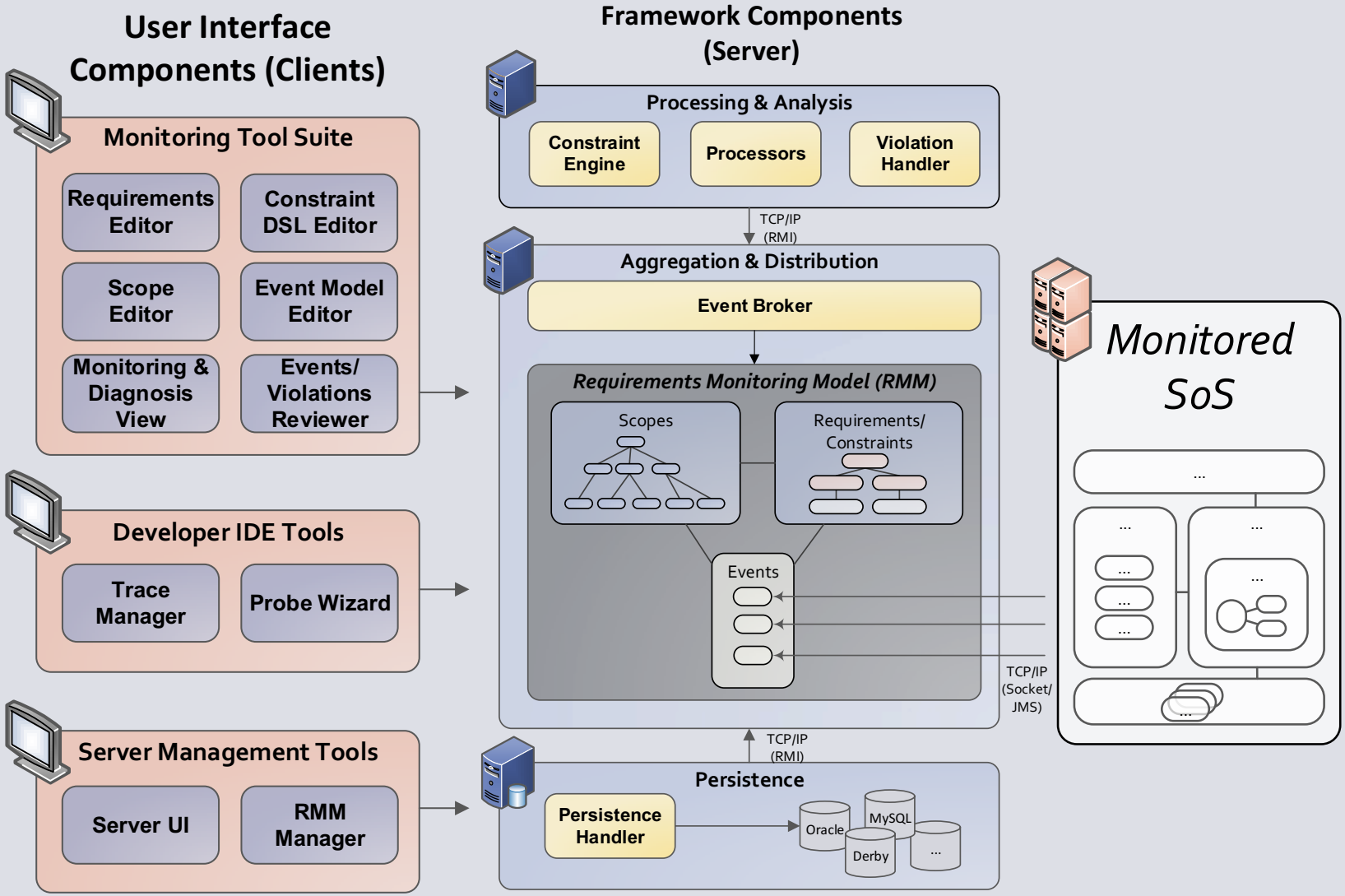
Generic Monitoring Framework

JOHANNES KEPLER UNIVERSITY LINZ



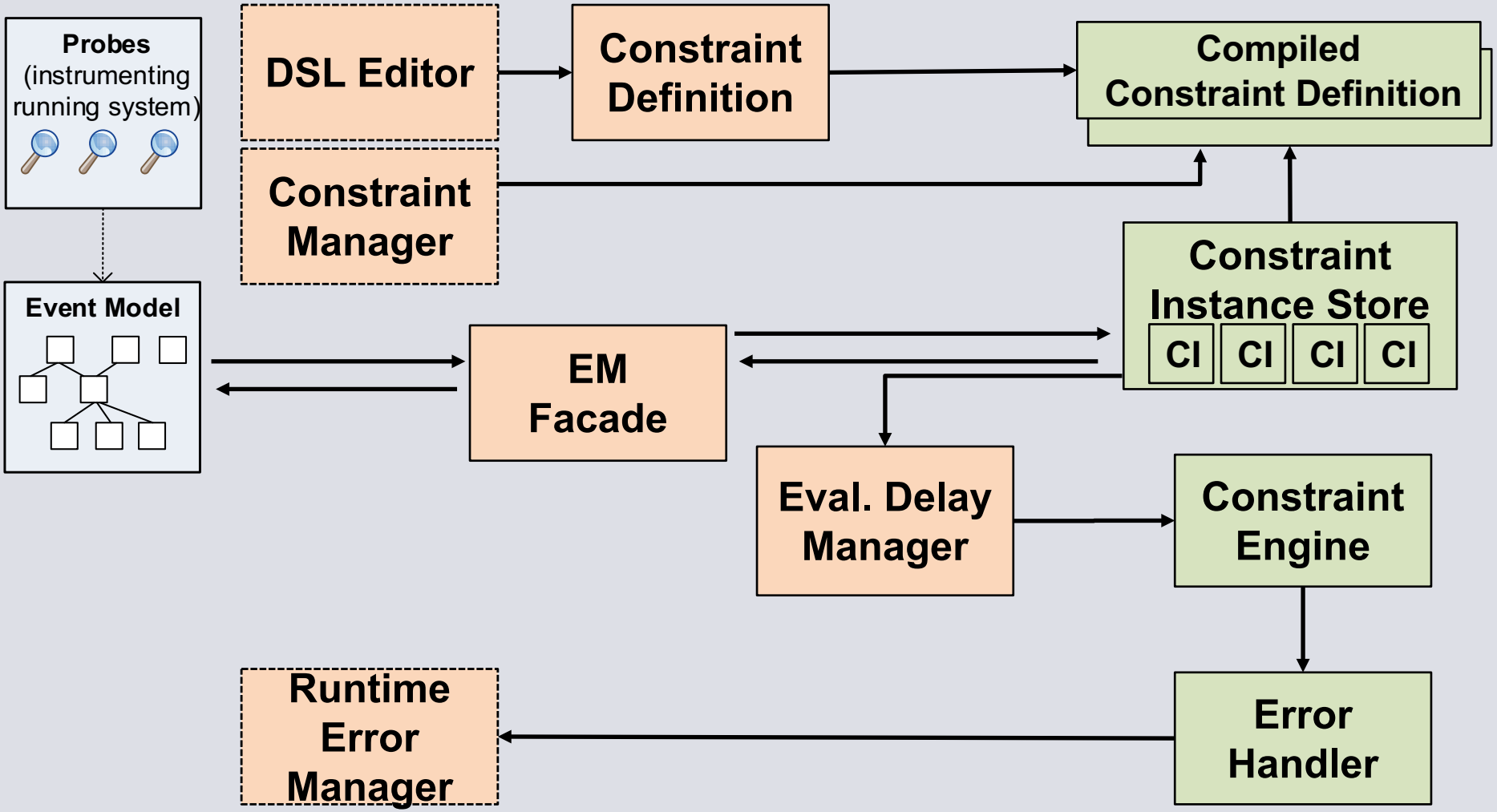
Vierhauser, Michael; Rabiser, Rick; Grünbacher, Paul; Danner, Christian; Wallner, Stefan; Zeisel, Helmut, "A Flexible Framework for Runtime Monitoring of System-of-Systems Architectures", *In: Proceedings 11th Working IEEE/IFIP Conference on Software Architecture (WICSA 2014)*, Sydney, Australia, 2014

Tool Architecture



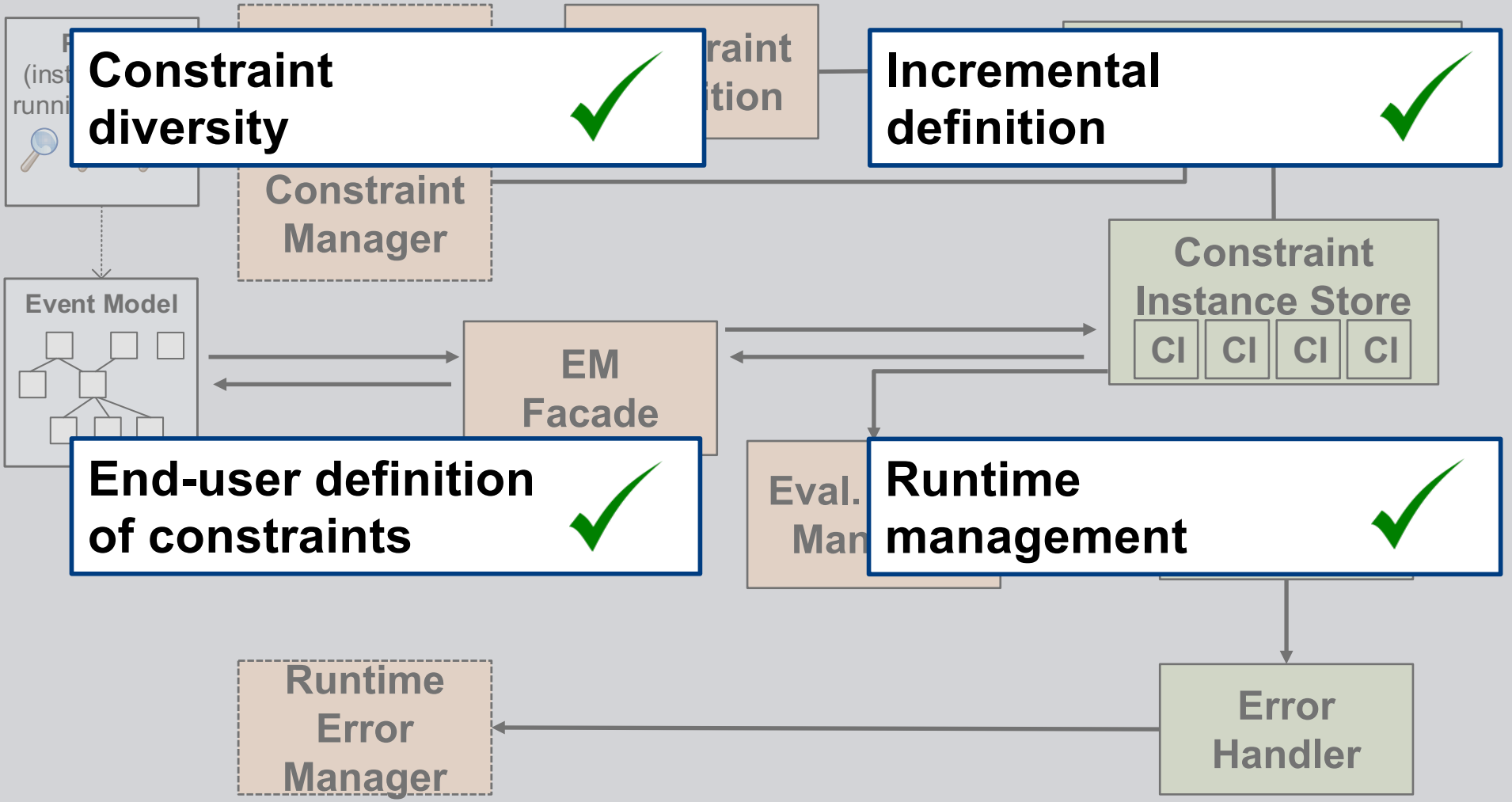
DSL and Incremental Constraint Checking

JOHANNES KEPLER UNIVERSITY LINZ



DSL and Incremental Constraint Checking

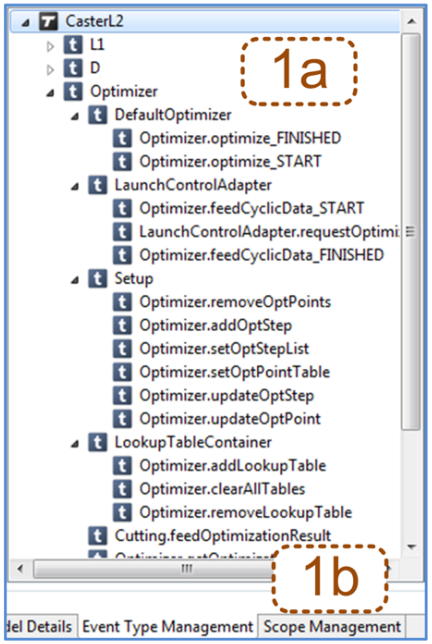
JOHANNES KEPLER UNIVERSITY LINZ



Typical Usage Scenario

JOHANNES KEPLER UNIVERSITY LINZ

(1) Defining events

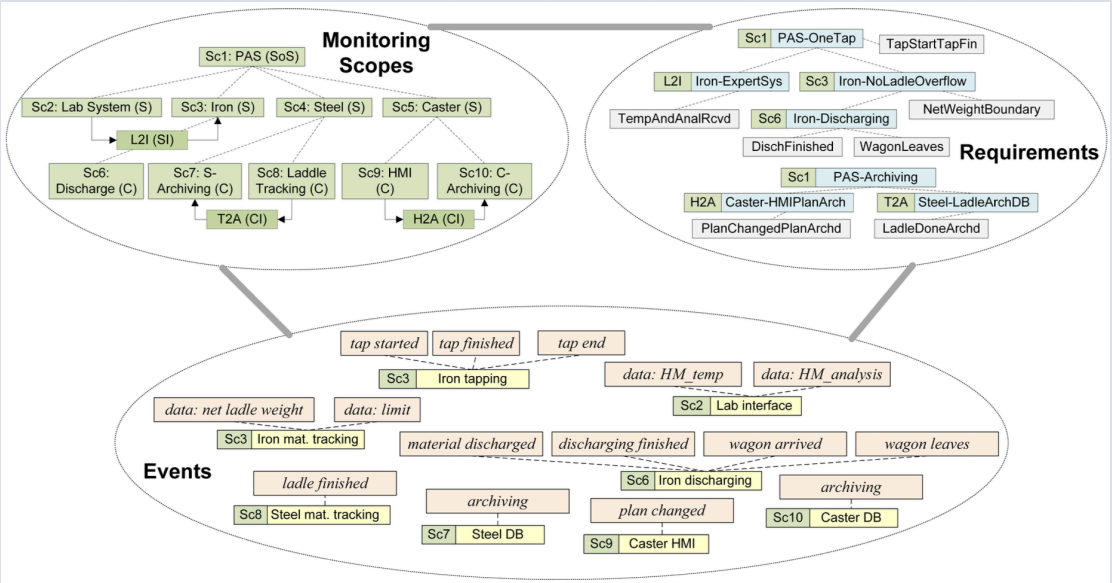


Evaluation of DSL Expressiveness

JOHANNES KEPLER UNIVERSITY LINZ

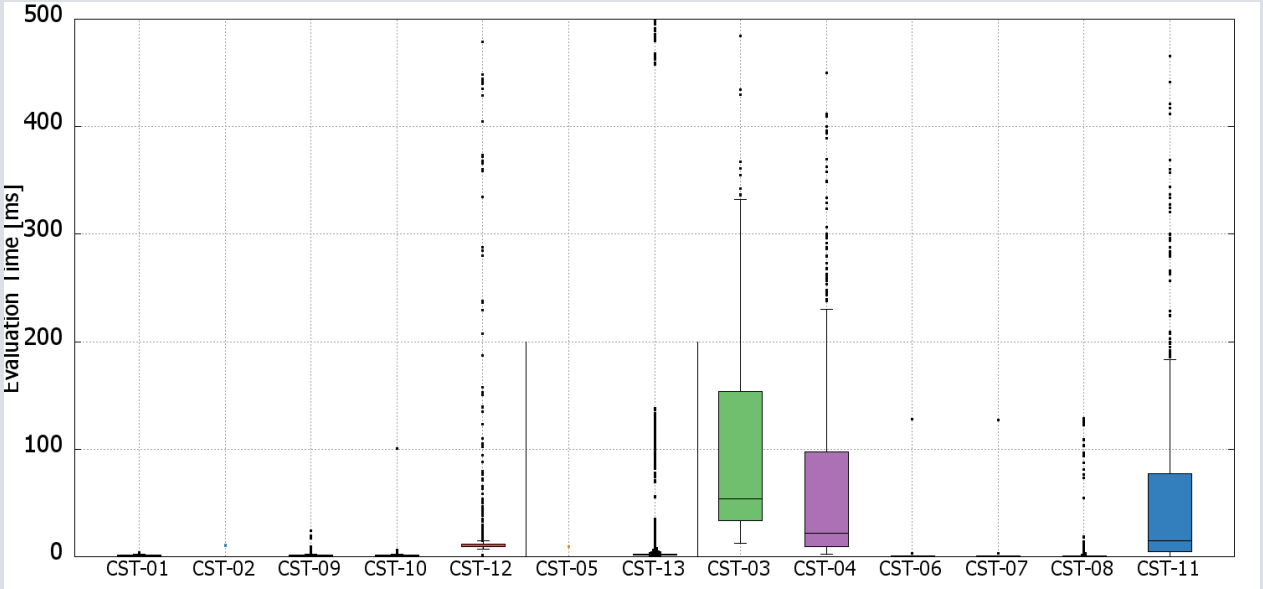
- Workshops & interviews with Primetals architects and engineers
- Analyzed technical specification documents

PAS RMM	
Element	#
Scopes	21
Rqts/Constraints	40
Event types	109
Probes	22



Evaluation of Checking Scalability

JOHANNES KEPLER UNIVERSITY LINZ



Evaluation Runs		
Element	6h Run	168h Run
Active Scopes	5	5
Checks	10,572	262,979
Events Captured	12,484	363,491
Active Probes	14	14

	Type	[#]	[ms]		Type	[#]	[ms]
CST-01	FUTURE	128	1.38	CST-08	DATA	9,700	0.37
CST-02	FUTURE	2	10.55	CST-09	FUTURE	687	1.20
CST-03	DATA	747	54.06	CST-10	FUTURE	584	1.37
CST-04	DATA	753	22.00	CST-11	DATA	548	14.64
CST-05	PAST	1	9.74	CST-12	FUTURE	584	9.71
CST-06	DATA	16	0.66	CST-13	PAST	55,029	1.55
CST-07	DATA	16	0.60				

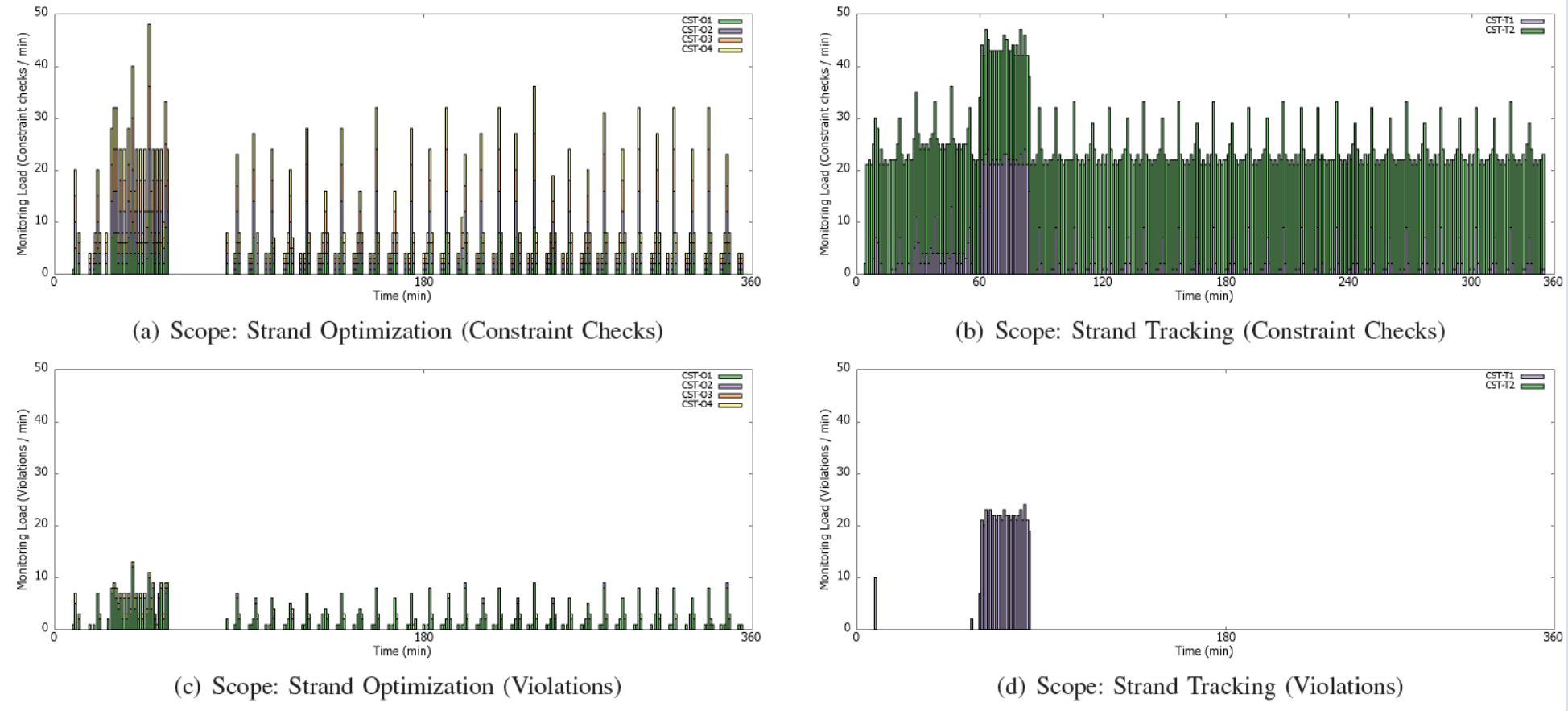
Simulation of Emergencies

JOHANNES KEPLER UNIVERSITY LINZ

Temporarily disabling StrandOptimization

Number of constraint checks and violations per minute

StrandOptimization *StrandTracking*



(a) Scope: Strand Optimization (Constraint Checks)

(b) Scope: Strand Tracking (Constraint Checks)

(c) Scope: Strand Optimization (Violations)

(d) Scope: Strand Tracking (Violations)

DSL and Constraint Checking: Lessons Learned

JOHANNES KEPLER UNIVERSITY LINZ

Diversity of requirements in SoS

Use an iterative language design + dynamic constraint management

The DSL will change and evolve over time

Simplify and automate extending the DSL

Too many features confuse

Keep the YAGNI* principle in mind

*"You Aren't Gonna Need It"

Avoid unnecessary dependencies

Keep the mapping of the DSL to the constraint checker flexible

ReMinds Benefits

JOHANNES KEPLER UNIVERSITY LINZ

- Monitoring at different layers and levels of granularity
- Monitoring across different systems and technologies
- Different speeds of systems
- Variability of system requirements and monitors
- Controlling the performance overhead of the monitoring solution

References

JOHANNES KEPLER UNIVERSITY LINZ

- William Robinson, A Roadmap for Comprehensive Requirements Modeling, IEEE Computer, May 2010 (vol. 43 no. 5), pp. 64-72
- Neil A. M. Maiden: Monitoring Our Requirements. IEEE Software 30(1): 16-17 (2013)
- M. Vierhauser, R. Rabiser, P. Grünbacher, K. Seyerlehner, S. Wallner, and H. Zeisel, "ReMinds: A Flexible Runtime **Monitoring Framework** for Systems of Systems," *Journal of Systems and Software, 2015*
- M. Vierhauser, R. Rabiser, P. Grünbacher, and A. Egyed, "Developing a **DSL**-Based Approach for Event-Based Monitoring of Systems of Systems: Experiences and Lessons Learned," 30th IEEE/ACM International Conference Automated Software Engineering, Lincoln, Nebraska, USA, 2015.
- M. Vierhauser, R. Rabiser, P. Grünbacher, and B. Aumayr, "A **Requirements Monitoring Model** for Systems of Systems," 23rd IEEE International Requirements Engineering Conference (RE'15), Ottawa, Canada, 2015.
- M. Vierhauser, R. Rabiser, and P. Grünbacher, "A **Case Study** on Testing, Commissioning, and Operation of Very-Large-Scale Software Systems," Proc. of the 36th International Conference on Software Engineering, ICSE Companion, Hyderabad, India, ACM, 2014, pp. 125-134.
- R. Rabiser, M. Vierhauser, P. Grünbacher, "Assessing the Usefulness of a Requirements Monitoring Tool: A Study Involving Industrial Software Engineers", *In: Proceedings 38th Int'l Conference on Software Engineering (ICSE 2016)*, Austin, USA, 2016