Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Providing Model-Extraction-as-a-Service for Architectural Performance Models

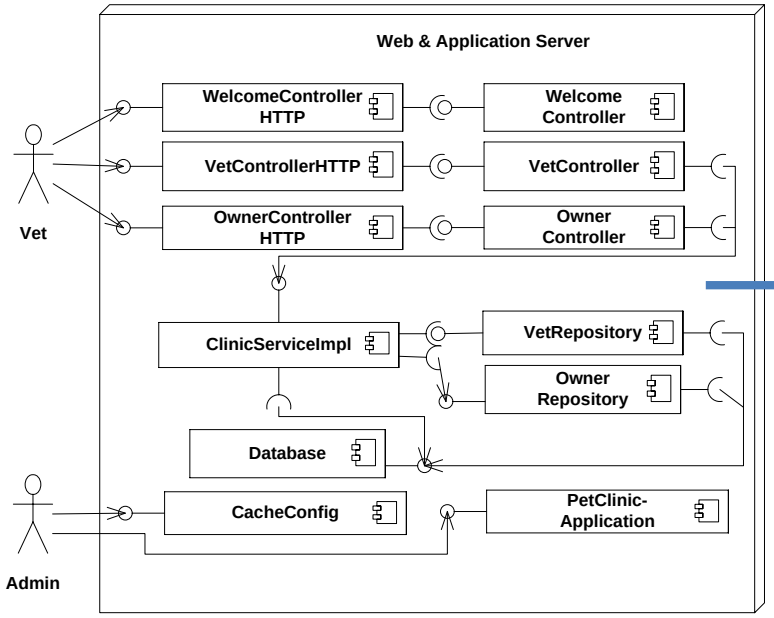
Jürgen Walter, Simon Eismann, Nikolai Reed, and Samuel Kounev

University of Würzburg

November 9-10, 2017
SSP 2017 Karlsruhe, Germany

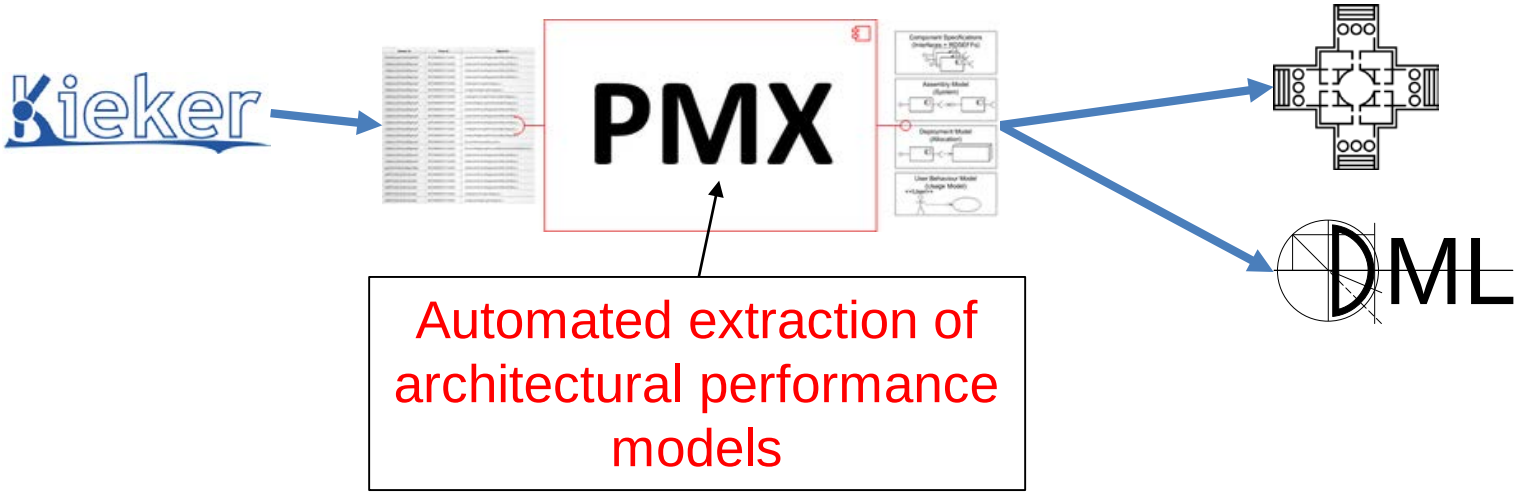
Motivation

- Architectural performance models …
  - can be applied to predict performance indices (response time, CPU utilization, throughput)
  - capture the semantics of the modeled system



What if analysis
architecture optimization
…

Motivation



[Walter2017] J. Walter et al. “An Expandable Extraction Framework for Architectural Performance Models”. In: Proceedings of the 3rd International Workshop on Quality-Aware DevOps (QUDOS’17). l’Aquila,Italy: ACM, Apr. 2017.

Problem

- Existing tools are cumbersome to setup and run
- Existing tools are not applied externally, due to cumbersome setup
- Existing approach does not scale

Idea

Provide web services for performance model extraction

Related Work

- Software-as-a-Service
  - Simulation-as-a-Service [Shekhar2016]
  - Modeling tools as web services [Cayirci2013].
- Performance Model Creation
  - Manual/semi-automated model extraction [Huber2010, Lehrig2016]
  - Automated model extraction [Brebner2016, Willnecker2016, Brunnert2017]
  - Subparts of model learning [Spinner2014, Spinner2015]

PMX: Learning of generic aspects

- PMX internally uses a pipes and filter architecture
- PMX reuses existing libraries where possible

- Operation call graph
- Resource landscape
- Deployment
- Job arrival rates

- Resource demands

PMX: Two step model learning

1. learning of generic aspects

Algorithm 1 Model Extraction Using Generic Builder

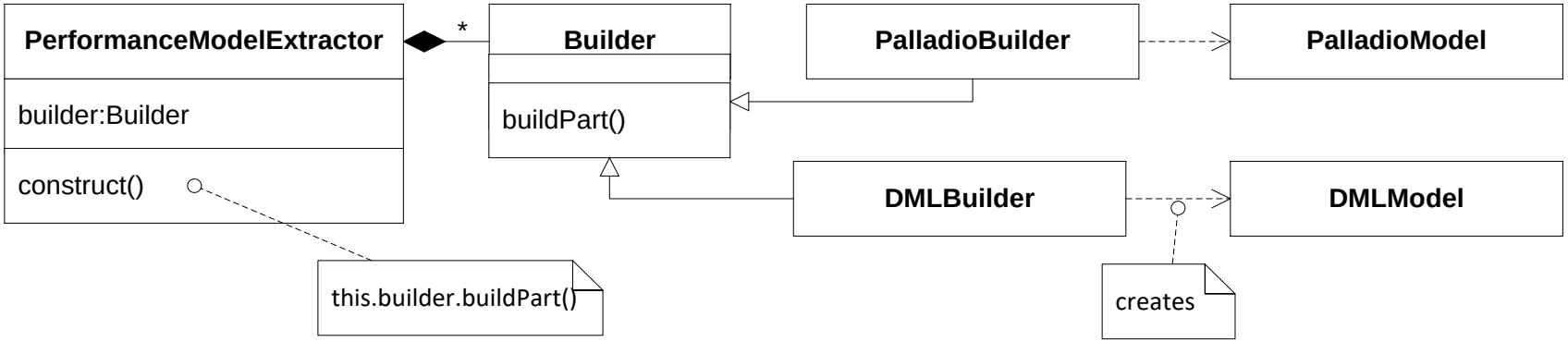
```
1: function CONSTRUCT(Path path, Builder builder)
2:     logs ← readLogFiles(path)
3:     analyzer ← compose analysis filters
4:     analyzer.analyze(logs)
5:     operationGraph ← analyzer.getOperationGraph()
6:     rds ← analyzer.getResourceDemands()
7:     workload ← analyzer.getWorkload()
8:     buildModel(operationGraph, rds, workload, builder);
9:     builder.save()
10: end function
```

Algorithm 2 Application of builder for Performance Model Generation

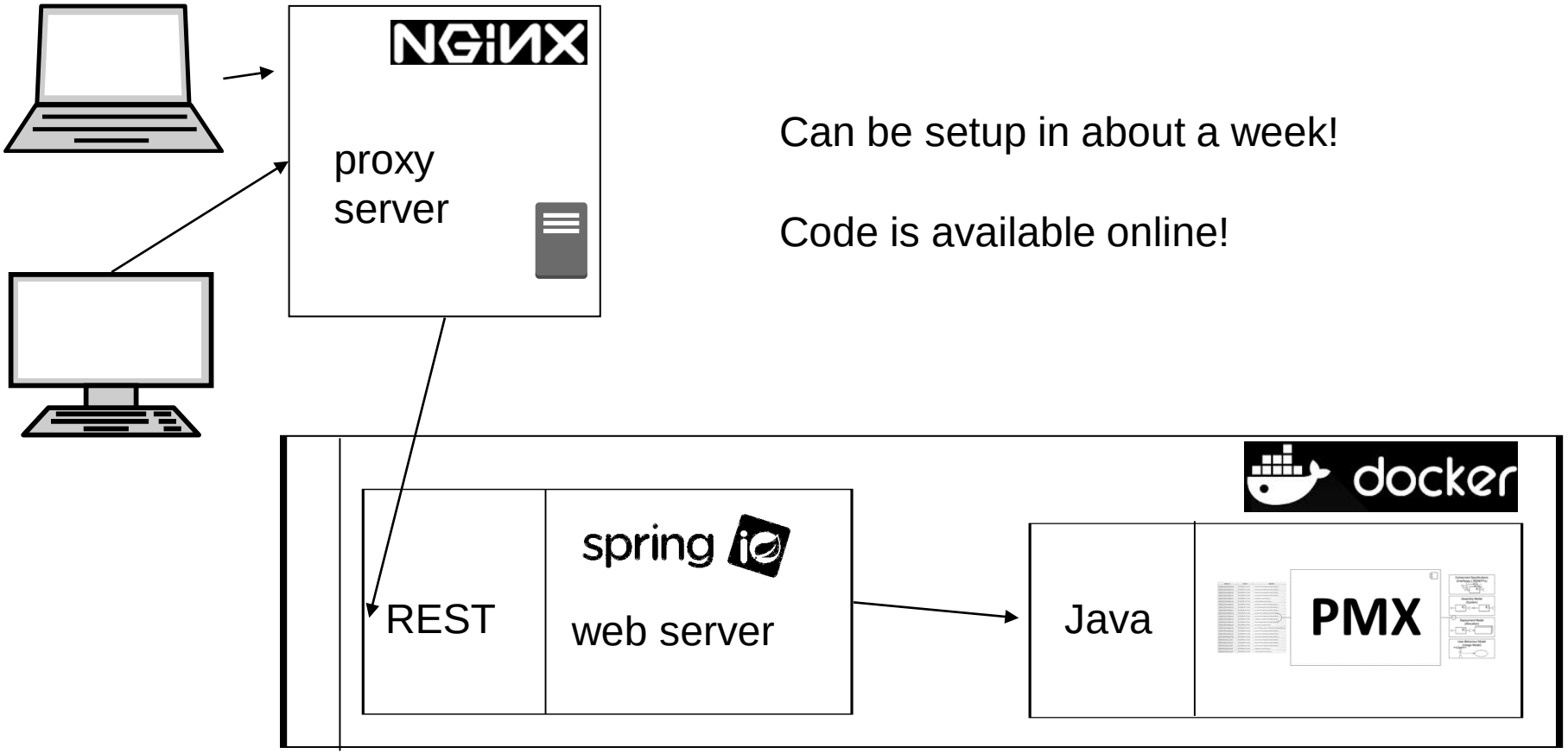
```
1: function BUILDMODEL(systemModel, operationGraph, resourceDemands, workload, builder)
2:     createHosts(systemModel, builder);
3:     createComponents(systemModel, builder);
4:     createInterfaces(systemModel, builder);
5:     createAllocations(systemModel, builder);
6:     for all source : operationGraph.vertices do
7:         component ← source.component.name
8:         host ← source.host.name
9:         assembly ← component + host
10:        builder.addAssembly(assembly);
11:        builder.assign(assembly, component);
12:        for all edge : source.outgoingEdges do
13:            target ← edge.getTargetVertice;
14:            tComponent ← target.component.name
15:            tHost ← target.host.name
16:            tAssembly ← tComponent + tHost
17:            builder.assign(tAssembly, tComponent);
18:            builder.connect(assembly, tAssembly);
19:            calls ← outgoing.getExternalCalls();
20:        end for
21:        rd ← resourceDemands.get(signature)
22:        builder.addBehavior(component, signature, calls, host, rd);
23:    end for
24: end function
```

2. model element creation

PMX: Builder Pattern



How to pack PMX into a web service



Can be setup in about a week!

Code is available online!

Spring I/O web server

- A web server allows to execute model extraction within a browser. The web interface provides an interaction layer to the user to interact with the model extraction layer (PMX) using a graphical interface.
- The web server contains
  - REST service implementation
  - HTTP pages user interface
  - software to trigger model extraction service
  - software to return results (file download)
- We decided for the Spring Framework, as it allows for easy implementation of the REST API that is used to upload files, trigger the model extraction, and download.

Docker container

- The docker container simplifies and speeds up deployment by packing web server and java application into a single container.
- The docker container allows deploying the application and access it locally within a web browser.
- We decided for Docker technology, as it provides light-weight easy to deploy software images

NGINX proxy server

- The proxy provides an interface between external users and docker container, and organizes communication.
  - For example, it provides a reverse proxy to do port forwarding and a secure layer for https.
- We decided for NGINX because it is light-weight, easy to use, and very popular. By also providing load balancing functionality, NGINX allows for an increased PMX user base in the future.

Benefits of MEaaS

- Simplified usage for all kinds of users
- Maintain software only for a single setup
- Potential collection of monitoring traces to further improve model extraction mechanisms

- MEaaS == Model-Extraction-as-a-Service

Future Work Web Service

- Software-as-a-Service
  - The model extraction might take longer than HTTP timeouts (especially if input traces are huge) ➔ asynchronous communication
  - There is a vulnerability for denial-of-service attacks. The significant resources required by a single model extraction request makes the system prone to such attacks ➔ login mechanisms.
- Privacy for monitoring data could be ensured running a separate service instance behind the firewall of a company

Future Work Model Extraction

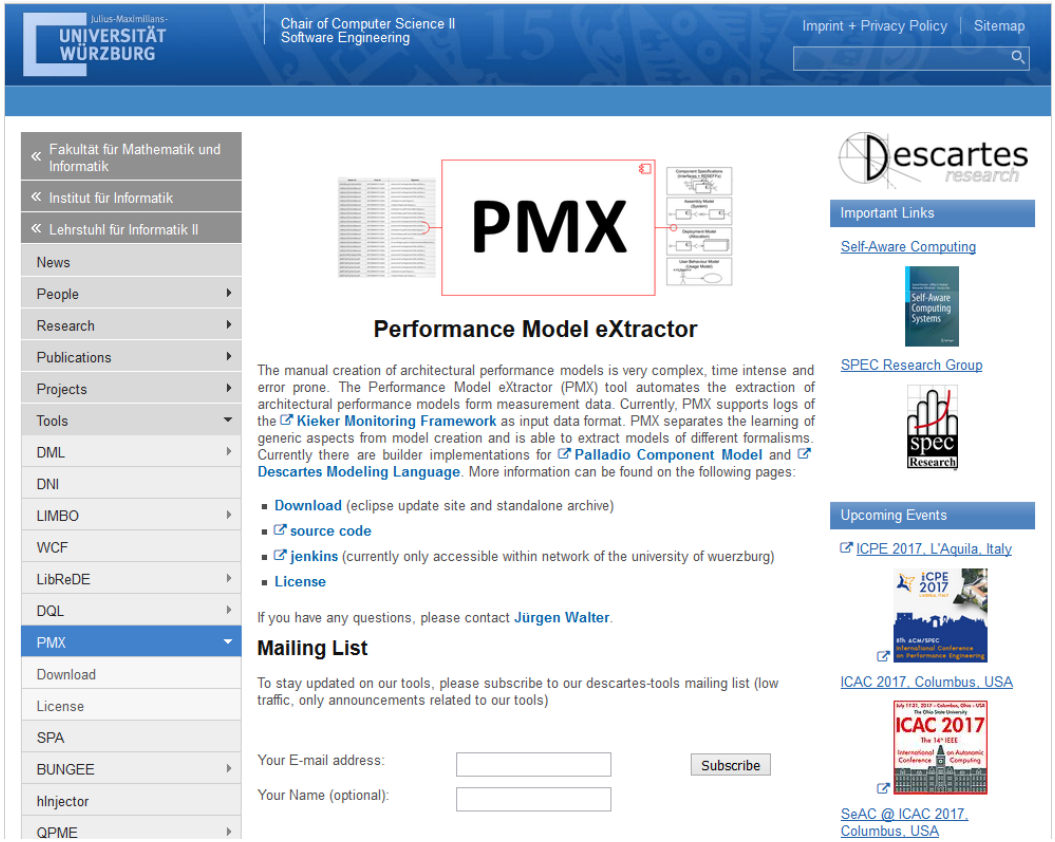
- Provide more builder implementations
- Conduct more case studies
- Allow for different monitoring tools and formats using OPEN.xtrace (formerly Common Trace API (CTA)) as input
- Use automatically derived performance models …
  - to integrate in load testing e.g., using a Jenkins plugin
  - for runtime resource management

Conclusion

- Based on an existing model extraction software, we presented additional software components of a web service to provide Model-Extraction-as-a-Service for architectural performance models.
- Provisioning as a web service reduces efforts to derive architectural performance models for all kinds of users.

Available online

- PMX core as well as builders and **web services** are available online http://descarte.tools/pmx/



Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Questions?

Jürgen Walter, Simon Eismann, Nikolai Reed, and Samuel Kounev

University of Würzburg

November 9-10, 2017
SSP 2017 Karlsruhe, Germany

References

[Cayirci2013] E. Cayirci. “Modeling and simulation as a cloud service: A survey”. In: 2013 Winter Simulations Conference (WSC). Dec. 2013, pp. 389–400

[Brebner2016] P. C. Brebner. “Automatic Performance Modelling from Application Performance Management (APM) Data: An Experience Report”. In: Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering, ICPE 2016, Delft, The Netherlands, March 12-16, 2016. 2016, pp. 55–61

[Brunnert2017] A. Brunnert and H. Krcmar. “Continuous performance evaluation and capacity planning using resource profiles for enterprise applications”. In: Journal of Systems and Software123 (2017), pp. 239–262

[Huber2010] N. Huber et al. “Performance Modeling in Industry: A Case Study on Storage Virtualization”. In: ACM/IEEE 32nd International Conference on Software Engineering (ICSE 2010), Software Engineering in Practice Track. Cape Town, South Africa:ACM, Mar. 2010, pp. 1–10.

[Lehrig2016] S. Lehrig and S. Becker. “Using Performance Models for Planning the Redeployment to Infrastructure-as-a-Service Environments: A Case Study”. In: 12th International ACM SIGSOFT Conference on Quality of Software Architectures, QoSA 2016, Venice, Italy, April 5-8, 2016. 2016, pp. 11–20

[Shekhar2016] S. Shekhar et al. “A simulation as a service cloud middleware”. In: Annals of Telecommunications 71.3 (Apr. 2016), pp. 93–108.

[Walter2017] J. Walter et al. “An Expandable Extraction Framework for Architectural Performance Models”. In: Proceedings of the 3rd International Workshop on Quality-Aware DevOps (QUDOS’17). l’Aquila,Italy: ACM, Apr. 2017.

[Willnecker2016] F. Willnecker and H. Krcmar. “Optimization of Deployment Topologies for Distributed Enterprise Applications”. In:2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA). Apr. 2016, pp. 106–115.