# Towards Modeling and Analysis of Power Consumption of Self-Adaptive Software Systems in Palladio

Christian Stier, Henning Groenda
FZI Research Center for Information Technology
Karlsruhe, Germany
{stier|groenda}@fzi.de

Anne Koziolek
Karlsruhe Institute of Technology
Karlsruhe, Germany
koziolek@kit.edu

**Abstract:** Architecture-level evaluations of Palladio currently lack support for the analysis of the power efficiency of software systems and the effect of power management techniques on other quality characteristics. This neglects that the power consumption of software systems constitutes a substantial proportion of their total cost of ownership. Currently, reasoning on the influence of design decisions on power consumption and making trade-off decisions with other Quality of Service (QoS) characteristics is deferred until a system is in operation. Reasoning approaches that evaluate a system's energy efficiency have not reached a suitable abstraction for architecture-level analyses. Palladio and its extension SimuLizar for self-adaptive systems lack support for specifying and reasoning on power efficiency under changing user load. In this paper, we (i) show our ideas on how power efficiency and trade-off decisions with other QoS characteristics can be evaluated for static and self-adaptive systems and (ii) propose additions to the Palladio Component Model (PCM) taking into account the power provisioning infrastructure and constraints.

## 1 Introduction

Palladio [BKR09] enables the evaluation of quality characteristics such as performance, cost or reliability for component-based software systems at early design stages. By predicting the performance and reliability of software systems it is possible to reason on design alternatives and infer whether agreed upon Service Level Agreements (SLAs) can be maintained. While Palladio accounts for the execution environment of software, its focus is on software-centric design decisions. Currently Palladio has limited support for data center design and sizing decisions. In particular, power consumption or provisioning are not considered although they are a major cost-factor in data centers. Power consumption accounts for roughly $15\%$ of a data center's Total Cost of Ownership (TCO) [GHMP08]. A data center needs to be equipped with suitably sized power provisioning infrastructure to avoid risking blackouts under peak load. Cost of an additional Watt of provisioned peak power is estimated around $\$11$ to $\$12.5$ [FWB07, GHLK$^+$12]. When power provisioning

Proc. SOSP 2014, Nov. 26–28, 2014, Stuttgart, Germany

and cooling costs are considered, the cost associated with power consumption is responsible for $40\%$ of a data center's TCO [GHMP08].

Even though power consumption decisively determines the TCO of a software system it is currently not sufficiently considered on an architectural level. Tradeoff decisions between power consumption and other quality dimensions are deferred to deployment time. The impact of design decisions on the energy efficiency of a software system can only be determined by accounting for the power consumption characteristics of the deployment environment.

Previous work on energy consumption analysis of software architectures focuses on specific architectural styles [SEMM08]. As it focuses on power consumption induced by communication it cannot be applied to predict the effect of individual design decisions on power consumption. Other approaches make limiting assumptions regarding the usage context [GWCA12] or application characteristics [MBAG10] which restrict their applicability outside of their specific problem domain. Brunnert et al. [BWK14] introduce the specification and evaluation of power consumption characteristics to Palladio. The authors assume power consumption to follow the same pattern for all physical machines. Even though a fixed model may accurately capture power consumption of current machines, it likely will become inaccurate in the future when considering recent trends in energy proportionality of physical machines [HP13]. Brunnert et al. perform an average case power consumption analysis. Consequently, it is not possible to identify phases in which the power consumption surpasses critical limits. As electricity pricing schemes usually factor in both peak power consumption and total energy consumption it is critical to take into account both average and peak power consumption for the modeled system [ZWW12].

This paper proposes (i) an approach enabling trade-off decisions between multiple quality characteristics of static and self-adaptive software systems. Our approach accounts for the operation of software systems in a data center. Our second contribution is (ii) an explicit model of the power consumption of software systems and the power provisioning infrastructure integrated into Palladio. The paper describes the *Power Consumption Analyzer (PCA)* approach leveraging the model and allowing continuous power consumption analysis. It supports reasoning on maintaining power consumption thresholds leveraging Palladio simulations and analysis of power-conscious *self-adaptation tactics* [PLL14] on an architectural level. Power consumption properties are evaluated in a post-simulation analysis that requires no modification of the simulation. In order to enable the analysis of power-conscious self-adaptation tactics, we extend the Palladio-based SimuLizar approach [BLB13] to support power consumption evaluations at intra-simulation using our PCA.

We show that our approach can be applied to evaluate whether limits in power consumption are adhered to for a given usage context. We are able to identify peaks in power consumption and violations not only for individual physical machines but also for Power Distribution Units (PDUs). By accounting for peak power consumption we improve the prediction of a data center's TCO when compared to previous work [BWK14].

This paper is structured as follows. Section 2 outlines foundations of our approach. Section 3 discusses related work. Section 4 introduces our model extensions to PCM. Section 5 sketches how the PCA interprets these model extensions to support power consumption

analysis. Finally, Section 6 summarizes the work presented in this paper and outlines our plans for future work.

## 2 Foundations

The *Palladio* [BKR09] approach enables software architects to predict quality characteristics of a component-based software architecture. The architecture is specified in the *Palladio Component Model (PCM)*. Software systems specified in PCM are assumed to be static: PCM does not include a specification of architectural runtime adaptations. Quality characteristics of a system defined in PCM can be predicted using analytical solvers [KBH07, KR08] or simulators [BKR09, MH11, BBM13].

Individual modeling concerns in PCM are separated into specialized submodels or views. PCM encompasses an explicit structural view on the hardware deployment environment onto which software components are deployed. The deployment environment is specified in the *Resource Environment* model. The *Resource Environment* model consists of a set of nested *Resource Containers*. Outer containers represent physical parts of a data center such as racks and compute nodes, whereas inner containers serve as operating system or virtualization layers. Resource Containers host a set of *ProcessingResourceSpecifications* where each specification represents a resource to be used by components, e.g. CPU or HDD.

SimuLizar by Becker et al. [BBM13, BLB13] extends Palladio to enable a systematic design of self-adaptive software systems. Self-adaptive software systems adapt their configuration at run-time to cope with changing environmental properties such as varying user load. One example for such a reconfiguration is the migration of a virtual machine from an over- to an under-utilized node. Reconfigurations are the outcome of self-adaptation tactics. Becker et al. subdivide tactics into "a condition (input) and a self-adaptation action (output)" [BLB13]. The condition specifies when a self-adaptation action is triggered depending on measurements taken via system probes. Probes are attached to PCM in the Palladio Measurement Specification (PMS) model. Currently, the PMS only allows to capture performance-centric measurements such as the response time or utilization. Analysis of the self-adaptive system is carried out with the SimuLizar simulator. The simulator allows predicting quality characteristics of a self-adaptive software system specified according to the SimuLizar approach. Software architects can use the provided simulative analysis to reason on the impact of self-adaptation mechanisms on quality characteristics of the simulated system.

*Energy-conscious self-adaptation tactics* form a subset of self-adaptation tactics that "modify runtime software configuration for the specific purpose of lowering energy consumption" [PLL14]. They are aimed at reducing consumption over a period of time. *Power-conscious self-adaptation tactics* perform reconfigurations to reduce the power draw of a software system at a specific point time. Both energy- and power-conscious self-adaptation tactics rely on consumption measurements or estimates when reasoning on the benefit of performing an adaptation.

*Power models* are used to estimate power consumption in absence of actual power measurements [RRK08]. A *power model* describes power consumption of hardware and software components, individual services or complete server nodes based on a set of system metrics. The fundamental assumption of power models is that power consumption correlates with the values of a set of system metrics, e.g. CPU utilization or I/O throughput. Examples include linear regression models correlating CPU utilization [FWB07] or software performance counters [ERKR06] with power consumption. Power models evaluate power consumption at a stationary point. In order to determine energy consumption over a time-frame, numerical integration algorithms can be applied.

## 3 Related Work

Seo et al. investigate the energy consumption of different architectural communication styles [SEMM08]. The authors present a set of evaluation models for specific styles. Among the analyzed communication styles are client-server and publish-subscribe. Their approach focuses on energy consumption caused by communication and disregards all other aspects of a software system. While their approach allows to compare energy efficiency of employing specific communication styles against each other it can not be applied to reason on power consumption of other architectural design decisions.

An approach for multi-objective architecture optimization for embedded systems is proposed by Meedenyia et al. [MBAG10]. In their paper the authors focus on the tradeoff between reliability and energy consumption. A model that evaluates the energy consumed by service calls is applied to predict workload-dependent energy consumption. In order to account for idle consumption the authors additionally include a static consumption offset. All services offered by a component are assumed to cause the same energy consumption. This is an acceptable abstraction for the embedded system domain where one component handles a homogeneous task, e.g. controlling the brakes based on sensor signals. The parameter space of these sensors are restricted by physical constraints such as the maximum speed of a car. This is not the case for other other domains like enterprise software. Business components offer an array of services with a large input parameter space. Depending on parameter values, business components require largely varying resource demands. Assuming that all services of a component consume the same amount of energy consequently would result in imprecise consumption predictions.

The design of energy efficient self-adaptive software systems on an architectural level has already been investigated by Götz et al. [GWCA12, GWR+13]. Main goal of the proposed architectural design framework is to find optimal system configurations for single users. This is achieved by adapting the runtime configuration of the system to the QoS requirements of a single user. Multi-user scenarios are not considered by the approach. Hence it cannot be applied to the design of self-adaptive systems that are used by multiple users. Götz argues that QoS requirements could be checked against an architectural model of the self-adaptive systems using simulation [GÏ3, p. 103f.]. The developer leaves the development of a concept for this evaluation open to be addressed in future work.

Brunnert et al. [BWK14] outline an approach for capturing application profiles that subsume the core characteristics of a static software system. These profiles are used to make QoS-driven deployment decisions for different software architectures. Runtime reconfigurations are not considered by the authors. Brunnert et al. enhance power consumption characteristics as part of PCM's Resource Environment specifications to enable design-time power consumption analysis for software systems. Reasoning on power consumption is limited to an average-case analysis of static software systems. It is thus not possible to identify whether restrictions on peak power consumption are violated at specific points in time. Consequently, sizing decisions for the power provisioning infrastructure are therefore limited.

*CloudSim* by Calheiros et al. [CRB+11] is a simulator for Infrastructure-as-a-Service (IaaS) data centers. It focuses on the operation and optimization of running virtual machines (VMs) in a data center. All information is provided as Java source code extending the CloudSim simulation environment. There is no model abstraction of system entities and their relation. Unlike Palladio, CloudSim does not consider parametric dependencies and has no explicit usage model. Rather it simulates VMs as isolated entities. Power consumption predictions per node are carried out based on the CPU utilization aggregated over all VMs that are deployed on the same node. VM load is modeled directly as an utilization function over time. Reasoning on the power consumption of a software system requires accurate utilization descriptions.

# 4 Modeling the Power Consumption Characteristics of Software Systems

In order to reason on the power consumption of a software system on architectural level its consumption properties need to be captured as part of its architectural description. This section outlines our extension of PCM with power consumption characteristics.

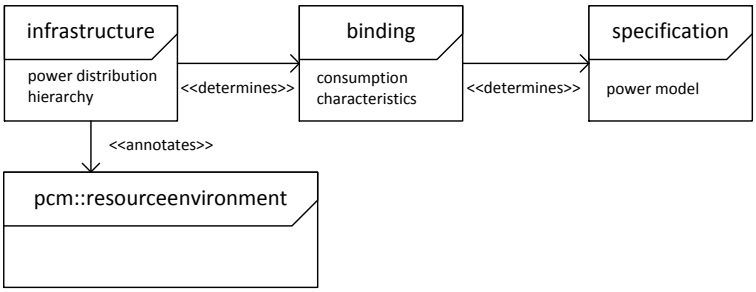


Figure 1: Relation between models used for specifying power consumption characteristics of a software system

PCM's Resource Environment models the hardware environment strictly by their performance and reliability characteristics. Our model annotates the hardware components in the

Resource Environment with their power consumption characteristics. Figure 1 depicts the relation between the model extensions and PCM. Our model introduces a model abstraction of the system's power provisioning infrastructure (*Power Infrastructure*). *Power Specification* describes models for evaluating the power consumption of hardware components and the power provisioning infrastructure. The *Power Binding* model links both models by binding each element in the infrastructure to the model used for evaluating its power consumption.

This section is organized as follows. Section 4.1 discusses advantages and disadvantages of different extension mechanisms that were considered for introducing power consumption characteristics to PCM. Section 4.2 presents the power provisioning model. In section 4.3 an overview is given on the model for specifying power models. Section 4.4 outlines the model used to specify consumption characteristics of hardware components.

### 4.1 Extending the Palladio Component Model

There are three approaches to extend PCM by another quality dimension. They are presented in the following including their advantages and drawbacks.

First, PCM can be invasively modified to include additional quality characteristics. This approach was taken for reliability [BKBR12] and is proposed by Brunnert et al. for power consumption predictions [BWK14]. The main disadvantage of an invasive extension is that it breaks support of existing tooling.

Second, PCM's profile extension mechanism [KDH+12] or EMF's child creation extenders [Mer08] can be used to introduce new properties and elements to PCM. These approaches are best taken when invasive changes to PCM are required that should not be propagated to all PCM-based tooling.

Third, it is possible to introduce a new meta-model which annotates or references existing PCM elements. As this alternative annotates existing model elements, it should only be chosen when no backwards navigation is needed or indicated.

We chose the last option where a separate model represents power consumption characteristics of the modeled software system. Model elements that have a counterpart in PCM's Resource Environment are annotated with their consumption characteristics. The reasons for choosing this extension mechanism are as follows. Wile power consumption is an important quality characteristic, it is not essential to other characteristics such as performance. Hence a non-invasive extension is indicated. Power consumption properties of hardware have no direct implications on their performance. It consequently suffices to introduce power consumption characteristics to PCM in an annotation model.

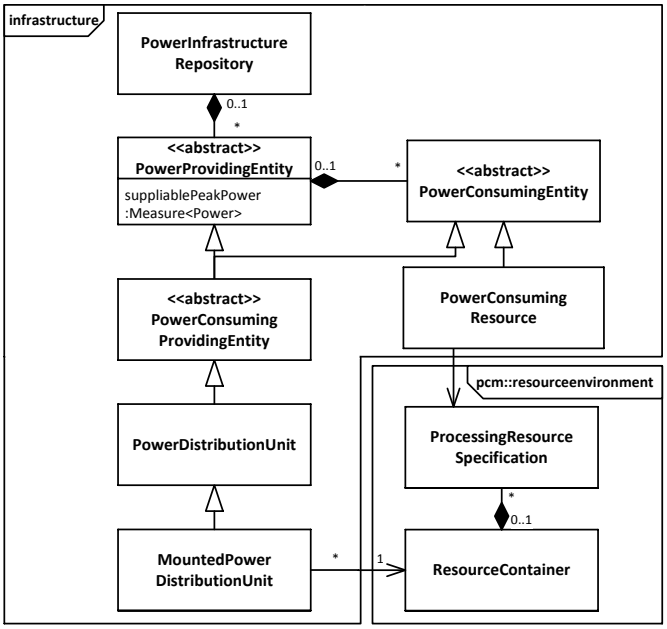


Figure 2: Power Infrastructure meta-model

### 4.2 Power Provisioning Infrastructure

The power provisioning infrastructure of data centers is typically structured hierarchically [FWB07]. It consists of a hierarchy of entities that provide and distribute power, e.g. PDUs, and entities that consume power, such as the physical components of a node. Figure 2 depicts our proposed model for the power provisioning infrastructure.

*PowerInfrastructureRepository* hosts a set of power provisioning infrastructure model descriptions. Typically only one infrastructure model is considered. However, the repository enables an easier modeling of alternative power provisioning infrastructures.

*PowerConsumingEntity* subsumes all components in the system that consume power.

*PowerConsumingResource* represents a physical resource that consumes power. It annotates a *ProcessingResourceSpecification* in a PCM instance with its consumption properties. Every PowerConsumingResource is a PowerConsumingEntity. An example for PowerConsumingEntity is the CPU of a compute node.

A *PowerProvidingEntity* distributes power to a set of *PowerConsumingEntities* nested below them. *suppliablePeakPower* defines the peak power that all PowerConsumingEntities connected to it can draw in total at any point in time.

Aside from entities that only provide or consume power there are also components that take on both providing and consuming roles, e.g. PDUs. *PowerConsumingProvidingEntity*

subsumes these entities.

*PowerDistributionUnit* extends *PowerConsumingProvidingEntity*. A PDU is connected to a power source (PowerProvidingEntity) from which it draws power. The PDU then further distributes power to connected PowerConsumingEntities.

*MountedPowerDistributionUnit* further specializes PowerDistributionUnit. It links the PDU to a ResourceContainer in PCM's Resource Environment. In essence, a mounted PDU corresponds with a Power Supply Unit of a node or a rack-mounted PDU. The relationship between a mounted PDU and its ResourceContainer is explicitly modeled so that consumption properties of nodes and racks can individually be traced.

### 4.3 Power Model Specifications

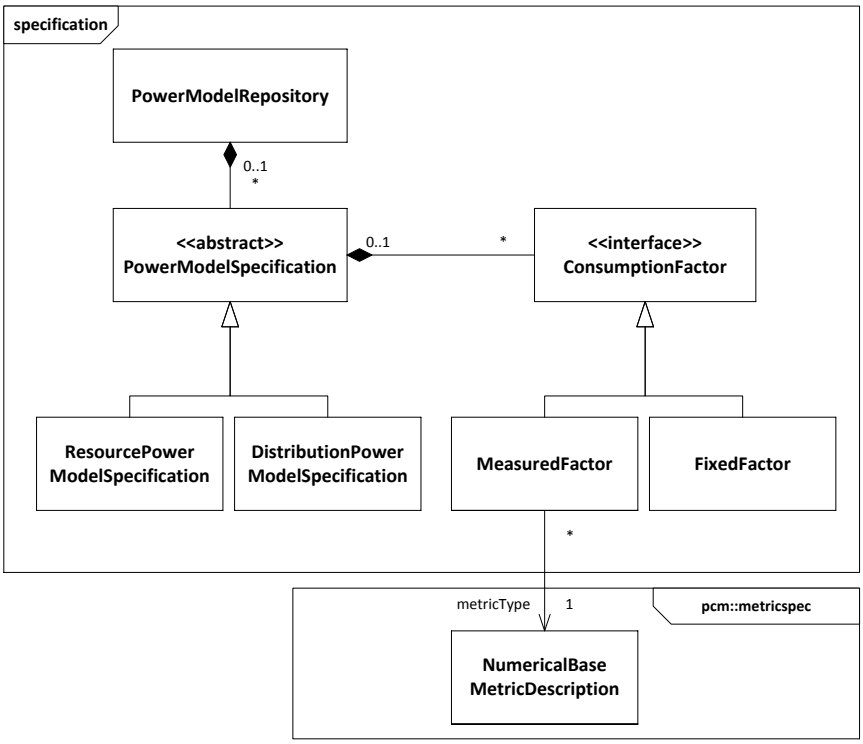


Figure 3: Power Specification meta-model

The *Power Specification* model shown in Figure 3 enables an explicit specification of power models and their input parameters. The calculation method is not specified in the Power Specification model instance. Rather, a calculator is implemented for each instance as part

of PCA as is explained further in Section 5.

The Power Specification Model is designed for unit support, e.g watt and ampere. Palladio's stochastical expression language *StoEx* [Koz08] was considered for specifying power models but ruled out due to lacking support for metric units.

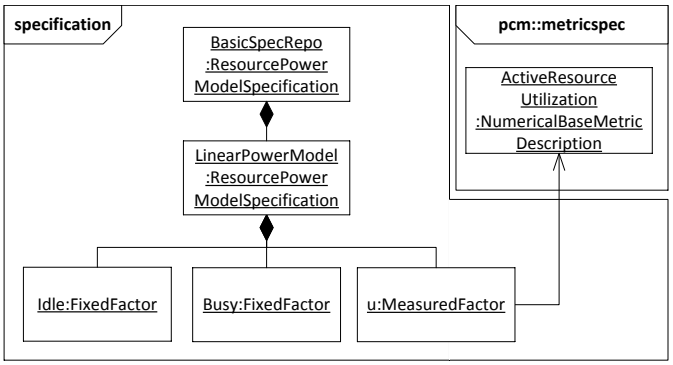


Figure 4: Linear power model represented as an instance of the PowerModelSpecification

A *PowerModelRepository* contains a set of *PowerModelSpecifications*. PowerModelSpecification represents a certain type of power consumption model (c.f. Figure 3). In the following, we will explain the concepts of the model based on the exemplary *LinearPowerModel* instance displayed in Figure 4. The depicted *LinearPowerModel* represents a simple linear regression-based power model for compute nodes as proposed by Fan et al [FWB07]. Fan et al. identify a close-to-linear relation between power consumption of a whole node and its CPU utilization . The linear power model predicts the power consumption $P(u)$ under utilization $u$ by interpolating between the node's power consumption in an *Idle* and *Busy* state: $P(u) = P_{Idle} + (P_{Busy} - P_{Idle}) \cdot u$.

A *ConsumptionFactor* specifies an input parameter of a power model. In the example, *Idle*, *Busy* and *u* are consumption factors. ConsumptionFactors are further distinguished into *Fixed* and *Measured Factors*.

*Fixed Factors* are independent from measurements and describe static power consumption characteristics of a system. For the linear power consumption model the fixed factors are made up of the CPU's power consumption under *Idle* ($P_{Idle}$) and *Busy* ($P_{Busy}$) load.

As their name implies, *Measured Factors* are extracted from the simulated system through measurements. A Measured Factor's metric is specified in direct reference to Palladio's *Metric Specification Framework* [Leh14]. For the linear power model the measured CPU utilization *u* is included as a Measured Factor.

Similar to Palladio's Resource Repository, instances of the Power Specification model are intended for reuse. Linear and other power models allow to predict the power consumption of a wide range of systems. Hence they are defined separately from the Power Infrastructure model.

This section discussed the specification of power models as a set of input parameters.

Consumption parameters alone do not suffice in evaluating the power consumption. It is necessary to define how the predictions are calculated from these parameters. We opted for a code-based realization of the calculation methods of power models. Section 5 provides details on the calculation.

### 4.4 Specifying Consumption Characteristics of the Infrastructure

Power consumption characteristics of the infrastructure are not directly specified in the infrastructure model. They are maintained in the separate *Binding* model and referenced from the Infrastructure model. The main rationale behind separating these two models lies in increased reuse and ease of variation compared to a direct specification of consumption characteristics in the infrastructure model. E.g., power consumption characteristics of an IBM System X3350M3 server can be described in terms of its consumption in idle (230 W) and busy states (410 W) as is done by Brunnert et al. [BWK14].

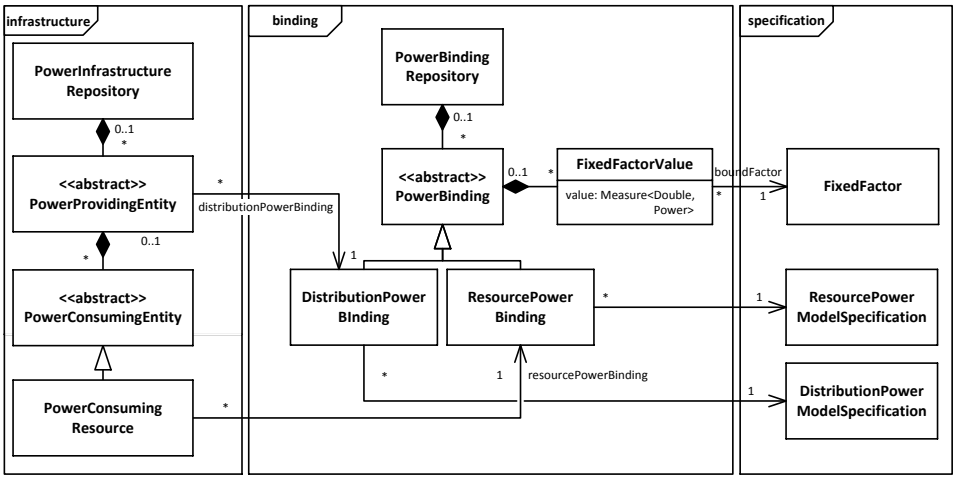


Figure 5: Power Binding meta-model

Figure 5 depicts our proposed binding model. It specifies the relation between the entities in the provisioning infrastructure and power models. Both *Resource* and *Distribution-PowerBinding* specialize *PowerBinding*.

A PowerBinding comes with a set of *FixedFactorValues*. Every FixedFactorValue determines the value of a FixedFactor. In case a server's power consumption can be described by a linear power model, the FixedFactorValues of the respective PowerBinding would specify the consumption of the node in a busy and idle state in Watt.

A *ResourcePowerBinding* defines the consumption characteristics of a type of PowerConsumingResource. Continuing with the previous example, all X3350M3 nodes in a system can reference the same binding. Should a CPU-based linear power consumption model not

sufficiently capture consumption characteristics of a node, e.g. because it is exclusively used as a file server, the PowerConsumingResource can reference a more suitable binding.

Every PowerProvidingEntity references a *DistributionPowerBinding*, which defines with what values the input parameters of a DistributionPowerModelSpecification are instantiated. As for ResourcePowerBinding, a DistributionPowerBinding can be reused across multiple PowerProvidingEntities.

## 5 Power Consumption Analyzer

The *Power Consumption Analyzer (PCA)* evaluates the power consumption of a software system. The analysis requires the specification of the system as a PCM instance and corresponding consumption annotations in the shape of a Power Consumption model (c.f. Section 4) instance. As part of a needs analysis for architecture-level power consumption, we identified two use cases covering static and self-adaptive software systems. PCA support both use cases, which are as follows: The *Power Consumption Analyzer (PCA)* evaluates the power consumption of a software system. The analysis requires the specification of the system as a PCM instance and corresponding consumption annotations in the shape of a Power Consumption model (c.f. Section 4) instance. As part of a needs analysis for architecture-level power consumption, we identified two use cases covering static and self-adaptive software systems. PCA support both use cases, which are as follows:

In the first use case, a software architect wants to predict power consumption of the system in a specific usage context. The software architect intends to compare the impact of design decisions on the systems' power consumption. In this case, the structure and consumption characteristics of hardware components have no impact on the system performance. Consequently, power consumption is analyzed as part of a *post-simulation* step. The advantage of analyzing power consumption separate from the simulation is that different power provisioning infrastructures can be compared without requiring additional time-intensive simulation runs. A new simulation is only necessary when both performance and power consumption characteristics have changed, e.g. because new servers are added to the resource environment.

In the second case, the software architect is interested in evaluating the impact a power-conscious self-adaptation tactic has on multiple quality characteristics. Power-conscious self-adaptation tactics use power consumption measurements to reason on the power efficiency of the system and issue system reconfigurations. The mutual dependency between current system architecture and power consumption requires an intra-simulation analysis.

The implementation of PCA is designed for both intra- as well as post-simulation power consumption analysis. Figure 6 sketches the integration of the PCA with the Quality Analysis Lab developed by Lehrig [Leh14] that is part of the upcoming Palladio release 3.5. The PCA evaluates the power consumption for a given *EvaluationContext*. This context consists of an instance of our Power Consumption model, the PCM instance it annotates, and a set of relevant measurements taken from the simulated system-under-analysis.

PCA computes the power consumption of individual PowerConsumingResources using

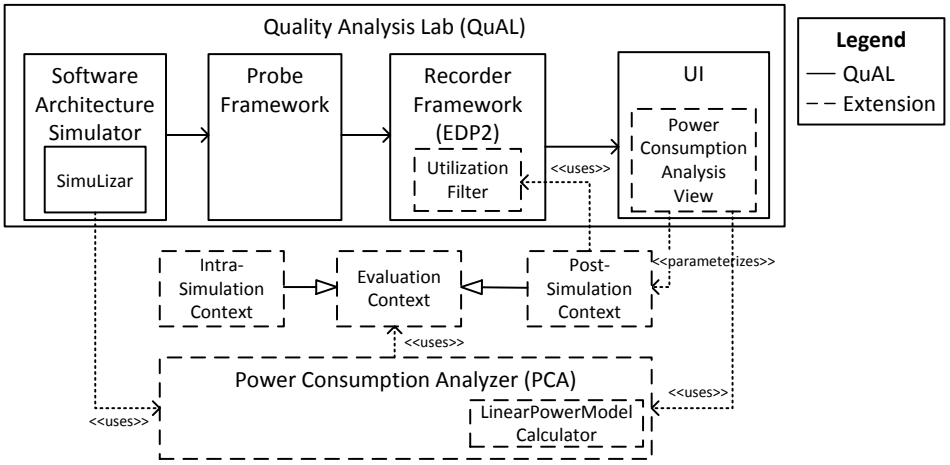


Figure 6: Integration of the Power Consumption Analyzer with Palladio's Quality Analysis Lab (QuAL) [Leh14]

*calculators*. A calculator programmatically evaluates power consumption of infrastructure elements. For example, the *LinearPowerModelCalculator* for the linear power model depicted in Figure 6 determines the power consumption of the node based on its current utilization by evaluating the factors of $P(u) = P_{Idle} + (P_{Busy} - P_{Idle}) \cdot u$. Every calculator offers a `calculate` method that takes exactly one measurement for every *metricType* of a MeasuredFactor specified in the corresponding PowerModelSpecification instance. FixedFactors do not change and are thus passed as part of the calculator's constructor parameters.

After determining power consumption of all nested PowerConsumingEntities the analyzer aggregates the power consumption of a PowerProvidingEntity. We apply the power model concept to individual elements in the power provisioning infrastructure. This design enables modeling of arbitrary conversion losses [PKA07] for each individual element in the provisioning infrastructure hierarchy.

The following sections briefly sketch how the PCA is used as part of post- and intra-simulation power consumption analysis to support software architects in designing power-efficient software systems.

### 5.1 Post-Simulation Power Consumption Analysis

Post-simulation power consumption analysis allows software architects to analyze the power consumption of a software system after a previous simulation run. It is possible to analyze the power consumption of any element in the power provisioning infrastructure at any given point in time. Consumption characteristics of the power provisioning infrastructure

can be changed without requiring repeated simulation. Post-simulation analysis leverages measurements that were extracted from the simulation for a set of system metrics, e.g. resource utilization for the linear power model presented in Section 4.3. Multiple power provisioning systems can be compared on the basis of the same simulation run as changes in the power consumption characteristics do not induce changes in the simulation logic. Software architects can configure the analysis, e.g. by setting the size of the interval for utilization-based metrics.

PCA calculates power consumption measurements of a PowerProvidingEntity by aggregating the power consumption of all connected elements. In case of post-simulation analysis the measurements are taken from the Recorder Framework (c.f. Figure 6), which stores all measurements of a simulation run. Power consumption is always calculated at a discrete point in time. Yet, measured data from the experiment does not necessarily contain measurements for said specific point in time. This is resolved by passing a derived calculated value from the post-simulation context to the analyzer.

### 5.2 Intra-Simulation Power Consumption Analysis

Power-conscious self-adaptation tactics as proposed in [JHJ+10, DSG+12, VAN08, RRT+08] adapt the system based on power consumption measurements. While each of the tactics has been evaluated for a set of specific software systems it is difficult to estimate how they would impact quality characteristics of other systems. Estimating the effect of using multiple energy-conscious self-adaptation tactics concurrently is even more difficult. Furthermore, power-conscious and other self-adaptation tactics can influence each other depending on specific thresholds for activation. By making power consumption measurements available at intra-simulation time software architects are enabled to evaluate combinations of all tactics and reason on QoS implications on an architectural level.

Intra-simulation power consumption analysis uses the same technical infrastructure as post-simulation power consumption analysis. The PCA derives power consumption metrics from measurements collected during the simulation by aggregating the power consumption of individual elements in the Power Infrastructure model instance. The only difference is that the PCA processes constantly updated measurements from a running simulation instead of pre-recorded measurements. Calculators provide power consumption estimates of hardware components to power consumption probes, which in turn provide power consumption measurements as a basis for decisions of power-conscious self-adaptation tactics.

We are currently implementing the presented concepts as an extension of SimuLizar.

## 6 Conclusion and Outlook

This paper presented a model-driven approach for analyzing the power consumption of static and self-adaptive software systems. By accounting for different power models and temporary peaks our analysis improves the precision of power consumption estimations on

an architectural level when compared to previous work [BWK14, MBAG10].

Our proposed Power Consumption model captures the consumption characteristics of a software system and its power provisioning infrastructure. It is designed to support design-time reasoning on the power consumption of power-conscious software systems. We presented how the three major aspects provisioning infrastructure (Power Infrastructure), consumption characteristics of hardware components (Power Binding) and the power models used to evaluate consumption characteristics of the components (Power Specification) can be abstracted. We showed the advantages of integrating them using a loose coupling concept. It allows software architects to separately define and extend a software system's power provisioning infrastructure and the power models used to predict the consumption of different types of hardware components.

The paper outlines a methodology and implementation for post- and intra-simulation power consumption analysis of software systems. Post-simulation analysis allows software architects to evaluate the power consumption of a software system. Thereby, multiple design alternatives can be compared against each other for their effect on power consumption. Not only is it possible to track peaks in power consumption for individual nodes but also for all other elements in the power provisioning infrastructure, e.g. PDUs. As a significant portion of large-scale software systems' TCO is determined by their power efficiency, this ultimately allows software architects to more accurately reason on the TCO. No repeated simulations are required when power consumption properties of the system-under-analysis are changed. Our approach further supports software architects in selecting a set of energy-conscious self-adaptation tactics that are best suited to meet QoS goals in multiple quality dimensions. We showed the integration of intra-simulation power consumption analysis with the SimuLizar approach and how our extension supports reasoning on energy-conscious self-adaptation tactics.

In future work we will increase the precision of power consumption predictions and enable architecture-level cost projections based on power consumption.

The Power Infrastructure model includes power sources as explicit entities. As of now, constant restrictions on the peak power provided by these PowerProvidingEntities are captured. We plan to introduce temporal constraints on power consumption that are driven by electricity price and availability. We expect a significant improvement of operational cost projections for software systems that are operated in dynamic environments.

Power efficiency of software systems can be controlled through hardware and middleware techniques such as ACPI [Hew13]. ACPI allows to control power efficiency by transitioning resources between states with different power consumption and performance properties. Palladio currently does not distinguish operational states of hardware. By introducing the modeling of operational states to Palladio's hardware resources, the evaluation of techniques such as power capping [RRT+08] will be supported. We expect this to improve accuracy of both power consumption and performance predictions.

We plan to account for reconfigurations to the execution environment of software systems, e.g. turning nodes off and on. This allows focusing on the optimization for whole data centres instead of a set of applications. Their inclusion should contribute further in achieving more precise power consumption predictions for self-adaptive software systems.

Further ideas for future work include an explicit modeling of the power consumption caused by network equipment, as well as cooling and ventilation. Our approach currently does not explicitly account for cooling and ventilation, as both are strongly correlated with power consumption of the computing infrastructure [FWB07]. When modeling a system's total power consumption we thus follow Fan et al.'s proposition to include them into our power models "as a fixed tax over the critical power" [FWB07]. However, the approach presented in this paper allows to include detailed power models for cooling by including them as PowerConsumingEntities. Dependencies between the heat generated by the compute hardware and their power consumption can be represented as MeasuredFactors to consider load variations in a data center. We currently account for network power consumption as a fixed factor since it is reported to be mostly static [FWB07, NPI+08]. Our model can be extended to include network power consumption in a similar way as for cooling and ventilation. The necessary steps comprise the introduction of network equipment as a specialized PowerConsumingEntity and the specification of their consumption properties in the Power Binding and Power Specification model.

## 7 Acknowledgements

This work is funded by the European Union's Seventh Framework Programme under grant agreement 610711.

## References

[BBM13] Matthias Becker, Steffen Becker, and Joachim Meyer. SimuLizar: Design-Time Modelling and Performance Analysis of Self-Adaptive Systems. In *Proceedings of Software Engineering 2013*, SE2013, February 2013.

[BKBR12] F. Brosch, H. Koziolek, B. Buhnova, and R. Reussner. Architecture-Based Reliability Prediction with the Palladio Component Model. *IEEE Transactions on Software Engineering*, 38(6):1319–1339, Nov 2012.

[BKR09] Steffen Becker, Heiko Koziolek, and Ralf Reussner. The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3 – 22, 2009. Special Issue: Software Performance - Modeling and Analysis.

[BLB13] Matthias Becker, Markus Luckey, and Steffen Becker. Performance Analysis of Self-Adaptive Systems for Requirements Validation at Design-Time. In *Proceedings of the 9th ACM SigSoft International Conference on Quality of Software Architectures*, QoSA '13. ACM, June 2013.

[BWK14] Andreas Brunnert, Kilian Wischer, and Helmut Krcmar. Using Architecture-level Performance Models As Resource Profiles for Enterprise Applications. In *Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures*, QoSA '14, pages 53–62, New York, NY, USA, 2014. ACM.

[CRB+11] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud

Computing Environments and Evaluation of Resource Provisioning Algorithms. *Softw. Pract. Exper.*, 41(1):23–50, January 2011.

[DSG+12] Corentin Dupont, Thomas Schulze, Giovanni Giuliani, Andrey Somov, and Fabien Hermenier. An Energy Aware Framework for Virtual Machine Placement in Cloud Federated Data Centres. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 4:1–4:10, New York, NY, USA, 2012. ACM.

[ERKR06] Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, and Partha Ranganathan. Full-System Power Analysis and Modeling for Server Environments. In *Workshop on Modeling Benchmarking and Simulation (MOBS)*, 2006.

[FWB07] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power Provisioning for a Warehouse-sized Computer. *SIGARCH Computer Architecture News*, 35(2):13–23, June 2007.

[GÏ3] Sebastian Götz. *Multi-Quality Auto-Tuning by Contract Negotiation*. PhD thesis, Technische Universität Dresden, Dresden, Germany, February 2013.

[GHLK+12] B. Grot, D. Hardy, P. Lotfi-Kamran, B. Falsafi, C. Nicopoulos, and Y. Sazeides. Optimizing Data-Center TCO with Scale-Out Processors. *Micro, IEEE*, 32(5):52–63, Sept 2012.

[GHMP08] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, December 2008.

[GWCA12] Sebastian Götz, Claas Wilke, Sebastian Cech, and Uwe Aßmann. *Sustainable ICTs and Management Systems for Green Computing*, chapter Architecture and Mechanisms for Energy Auto Tuning, pages 45–73. IGI Global, June 2012.

[GWR+13] Sebastian Götz, Claas Wilke, Sebastian Richly, Christian Piechnick, Georg Püschel, and Uwe Aßmann. Model-driven Self-optimization Using Integer Linear Programming and Pseudo-Boolean Optimization. In *The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2013)*, pages 55–64. IARIA, 2013.

[Hew13] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation. Advanced Configuration and Power Interface Specification, 2013.

[HP13] Chung-Hsing Hsu and S.W. Poole. Revisiting Server Energy Proportionality. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 834–840, Oct 2013.

[JHJ+10] Gueyoung Jung, M.A Hiltunen, K.R. Joshi, R.D. Schlichting, and C. Pu. Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. In *2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, pages 62–73, June 2010.

[KBH07] Heiko Koziolek, Steffen Becker, and Jens Happe. Predicting the Performance of Component-Based Software Architectures with Different Usage Profiles. In Sven Overhage, Clemens A. Szyperski, Ralf Reussner, and Judith A. Stafford, editors, *Software Architectures, Components, and Applications*, volume 4880 of *Lecture Notes in Computer Science*, pages 145–163. Springer Berlin Heidelberg, 2007.

[KDH+12] Max E. Kramer, Zoya Durdik, Michael Hauck, Jörg Henss, Martin Küster, Philipp Merkle, and Andreas Rentschler. Extending the Palladio Component Model using Profiles and Stereotypes. In Steffen Becker, Jens Happe, Anne Koziolek, and Ralf Reussner, editors, *Palladio Days 2012 Proceedings (appeared as technical report)*, Karlsruhe Reports in Informatics ; 2012,21, pages 7–15, Karlsruhe, 2012. KIT, Faculty of Informatics.

[Koz08] Heiko Koziolek. *Parameter dependencies for reusable performance specifications of software components*. PhD thesis, Carl von Ossietzky University of Oldenburg, 2008.

[KR08] Heiko Koziolek and Ralf Reussner. A Model Transformation from the Palladio Component Model to Layered Queueing Networks. In *Proceedings of the SPEC International Workshop on Performance Evaluation: Metrics, Models and Benchmarks*, SIPEW '08, pages 58–78, Berlin, Heidelberg, 2008. Springer-Verlag.

[Leh14] Sebastian Lehrig. Quality Analysis Lab (QuAL): Software Design Description and Developer Guide Version 0.2. Technical report, Universität Paderborn, Faculty of Electrical Engineering - Computer Science - Mathematics, April 2014.

[MBAG10] Indika Meedeniya, Barbora Buhnova, Aldeida Aleti, and Lars Grunske. Architecture-Driven Reliability and Energy Optimization for Complex Embedded Systems. In George T. Heineman, Jan Kofron, and Frantisek Plasil, editors, *Research into Practice - Reality and Gaps*, volume 6093 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin Heidelberg, 2010.

[Mer08] Ed Merks. Creating Children You Didn't Know Existed. `http://ed-merks.blogspot.de/2008/01/creating-children-you-didnt-know.html`, January 2008. Online; accessed 31/10/2014.

[MH11] Philipp Merkle and Jörg Henss. EventSim – An Event-driven Palladio Software Architecture Simulator. In Steffen Becker, Jens Happe, and Ralf Reussner, editors, *Palladio Days 2011 Proceedings (appeared as technical report)*, Karlsruhe Reports in Informatics ; 2011,32, pages 15–22, Karlsruhe, 2011. KIT, Fakultät für Informatik.

[NPI+08] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing Network Energy Consumption via Sleeping and Rate-adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 323–336, Berkeley, CA, USA, 2008. USENIX Association.

[PKA07] A. Pratt, P. Kumar, and T.V. Aldridge. Evaluation of 400V DC distribution in telco and data centers to improve energy efficiency. In *Telecommunications Energy Conference, 2007. INTELEC 2007. 29th International*, pages 32–39, Sept 2007.

[PLL14] Giuseppe Procaccianti, Patricia Lago, and Grace A. Lewis. Green Architectural Tactics for the Cloud. In *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*, pages 41–44, April 2014.

[RRK08] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A Comparison of High-level Full-system Power Models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[RRT+08] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center. *SIGARCH Comput. Archit. News*, 36(1):48–59, March 2008.

[SEMM08] Chiyoung Seo, G. Edwards, S. Malek, and N. Medvidovic. A Framework for Estimating the Impact of a Distributed Software System's Architectural Style on its Energy Consumption. In *Seventh Working IEEE/IFIP Conference on Software Architecture*, WICSA 2008, pages 277–280, February 2008.

[VAN08] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. In Valrie Issarny and Richard Schantz, editors, *Middleware 2008*, volume 5346 of *Lecture Notes in Computer Science*, pages 243–264. Springer Berlin Heidelberg, 2008.

[ZWW12] Yanwei Zhang, Yefu Wang, and Xiaorui Wang. Electricity Bill Capping for Cloud-Scale Data Centers that Impact the Power Markets. In *Parallel Processing (ICPP), 2012 41st International Conference on*, pages 440–449, Sept 2012.