Proceedings of BS2015:
14th Conference of International Building Performance Simulation Association, Hyderabad, India, Dec. 7-9, 2015.

# PROTOTYPING THE NEXT GENERATION ENERGYPLUS SIMULATION ENGINE

Michael Wetter[1], Thierry S. Nouidui[1], David Lorenzetti[1],
Edward A. Lee[2] and Amir Roth[3]
[1]Lawrence Berkeley National Laboratory, Berkeley, CA
[2]University of California at Berkeley, Berkeley, CA
[3]US Department of Energy, Washington, DC

## ABSTRACT

We describe the prototype of a next-generation implementation of EnergyPlus, DOE's whole-building energy simulation engine. This new implementation breaks EnergyPlus into a set of component models with clearly defined input and output ports. It instantiates these components and their connections from the EnergyPlus input file – thereby not disrupting applications that use EnergyPlus – and then simulates them using a discrete event simulator. This new structure should allow EnergyPlus to evolve more rapidly and robustly by decoupling component modules from the numerical solver. It also allows models to be exported for integration with building control systems.

We prototyped this new implementation using the open-source Ptolemy II framework. We encapsulated the computing modules as Functional Mockup Units (FMUs) for Model Exchange. The system of equations defined by the connection of the FMUs is integrated using Quantized State System (QSS) simulation, a novel method that partitions systems of differential equations and integrates them asynchronously, using step sizes that are based on the time rate of change of the individual state variables.

We present a numerical example that illustrates the asynchronous integration and numerical benchmarks of a multizone building model with a radiant slab. We compare the computing time among our prototype, EnergyPlus version 8.2 and the Dymola 2015 FD01 Modelica simulation engine.

## INTRODUCTION

EnergyPlus is DOE's open-source whole building energy simulation program (Crawley et al., 2001). It is a state-of-the-art building energy simulation program with a significant user base that serves as the basis for both energy-efficiency codes and a growing ecosystem of commercial software. It is also nearing its twentieth birthday, having originated from a union of DOE-2 (Winkelmann and Selkowitz, 1985) and BLAST (BLAST, 1999) in 1996.

As EnergyPlus has grown, its traditional monolithic, imperative structure – which intermingles governing equations of physics, numerical solution methods and idealized control schemes – has become more burdensome. It makes EnergyPlus difficult to maintain and extend as solvers for new models must be carefully integrated with the existing solver. It complicates modeling of modular HVAC systems as it is rigidly organized around traditional primary and secondary HVAC loops. And it is not suited for simulation of control schemes other than rule-based supervisory control sequences because its load-based models have inputs and outputs that are semantically different from actuator commands and sensor signals, respectively, and the numerical methods cannot handle fast dynamics, events, certain sampled systems and finite state machines. EnergyPlus' control language is also bespoke and meaningless outside of EnergyPlus itself. In short, EnergyPlus is a self-contained ecosystem that provides few opportunities to leverage or reuse outside components, platforms, technologies, and expertise.

Spawn-of-EnergyPlus (SOEP) is a prototype using a partially new implementation of EnergyPlus that addresses these structural issues. SOEP leverages two open standard simulation technologies – the Modelica language (Mattsson et al., 1999) and the Functional Mockup Interface (FMI) (Blochwitz et al., 2011) for co-simulation and model-exchange. Modelica is a declarative modeling language in which developers write the governing equations of the system and link them with an external, domain-indepdendent solver. By separating models from numerical solvers, Modelica allows domain experts to focus on their domain while leveraging outside expertise to develop high-performance simulation platforms. Modelica makes it easy to prototype new models, to share models between simulation environments, and even to repurpose models for other applications. In the specific case of buildings, Modelica control models can be directly translated into working controller code – unlike current models written in EnergyPlus Runtime Language (ERL).

SOEP exploits FMI to encapsulate existing EnergyPlus models for envelope heat transfer, lighting, and airflow as Functional Mockup Units (FMU) and simulate them together with new HVAC and control FMUs generated from Modelica.

SOEP selects, instantiates, and connects FMUs by interpreting EnergyPlus' existing input files and is therefore backward-compatible with EnergyPlus. FMU time-stepping, state variable integration, and solution of the algebraic loops formed by connecting FMU are performed by the open-source actor-based framework Ptolemy II. The specific time integration method used is Quantized State System (QSS) (Zeigler and Lee,

1998; Kofman and Junco, 2001; Cellier and Kofman, 2006; Kofman, 2003; Bliudze and Furic, 2014). QSS methods schedule a module's computation in proportion to the time rate of change of that modules state variables. They allow for explicit scheduling of state events, making them attractive for control simulation.
The remainder of the paper is structured as follows. After describing SOEP's use cases, we describe background work that we use for our implementation. Then, we describe our implementation and provide numerical benchmarks. We conclude with related work and open research questions.

## USE CASES

SOEP must be able to simulate all physical phenomena that are typically encountered in building energy simulation, including heat and moisture transfer through the building fabric, daylight illuminance, natural ventilation, pressure driven flow distribution in ducts and pipe networks, HVAC systems, chilled water plants, electrical systems, water usage and occupant behavior. In addition, it must be able to model ideal and actual control sequences. These capabilities should support the following use cases:

1. Whole building annual simulation.
2. Equipment sizing.
3. Controls design.
4. Nonlinear optimization such as for design or model predictive control, input/output linearization for controls design, or system identification to support adaptive models.[1]
5. Extraction of component and subsystem models in order to execute them in isolation for given input and initial states, or to upload them to software that can communicate with building controllers, such as through Niagara Tridium.

## BACKGROUND

This section describes key underlying technologies.

### Actor-based modeling with Ptolemy II

SOEP uses the Ptolemy II simulation framework, whose module abstraction is that of *actors*, which are components that communicate through ports that send and receive data. These data are called *tokens* and can contain various objects such as a double value, a string or more complex data types. A Ptolemy II simulation model consists of instances of actors whose input and output ports are connected, and a director. The *director* is responsible for sending tokens from output ports to input ports and for invoking the actors' computation methods. The director implements the *model of computation*, such as discrete event domain, continuous time domain for solving ordinary differential equations (ODE), or algebraic loop solver. Most Ptolemy II actors are polymorphic, *i.e.*, they can be used with different models of computation.
Whereas EnergyPlus is a discrete time simulation, SOEP uses the discrete event domain of Ptolemy II, in which actors send each other time-stamped events, and the director processes these events in time-stamp order. This style of discrete event simulation is widely used for large, complex systems (Cassandras, 1993; Zeigler et al., 2000; Fishman, 2001). The particular implementation in Ptolemy II has a sound, deterministic semantics (Lee, 1999; Matsikoudis and Lee, 2013).

### Model integration using FMI

FMI is an open standard for *co-simulation* and *model-exchange*. In co-simulation, a model including its time integration algorithm is exchanged. Hence, given state variables, inputs and a communication time step, the FMU returns new state variables and outputs. In model-exchange, the time integration of the differential equations has to be done by a master algorithm. Hence, given state variables, inputs and time, the FMU returns the time derivatives and outputs. FMI defines a simulation application programming interface (API), uses XML to declare model properties and capabilities, and acts as a container for source code, binaries, and weather files. More than 60 simulation tools support the FMI standard, including EnergyPlus which uses it to support co-simulation with engines like CONTAM (Walton and Dols, 2013).
SOEP calls for re-factoring EnergyPlus *internally* into computing models encapsulated as FMUs. EnergyPlus' hand-crafted C++ models for envelope heat-transfer, lighting, and airflow – which use special numerical approaches that exploit the structure of the calculation – will be reused directly. HVAC and control models will be gradually migrated to Modelica implementations. The use of FMI internally will allow SOEP to incorporate external models as long as they conform to the FMI standard, the Modelica standard, or are Ptolemy II actors. Specifically, it will allow manufacturers to provide models that characterize their equipment in either source (*e.g.*, Modelica or C) or binary FMU forms.

### Modular modeling with Modelica

Modelica is an open-standard language for object-oriented equation-based modeling. In Modelica, developers build models by writing their differential, algebraic or discrete equations. Models can be hierarchically composed into larger ones, either textually or graphically in a schematic editor. The system of equations is analyzed and manipulated using computer algebra (Cellier and Kofman, 2006), translated to C and then linked with and solved by an external, domain-independent solver.
Originating in the automotive and aerospace industries, Modelica is a robust standard with significant support across multiple sectors and a growing ecosystem of tools capable of solving large real-world prob-

[1] These applications have in common that they all require repeated simulations with given initial states for any simulation period, and that outputs and state trajectories needs to be differentiable with respect to input signals.

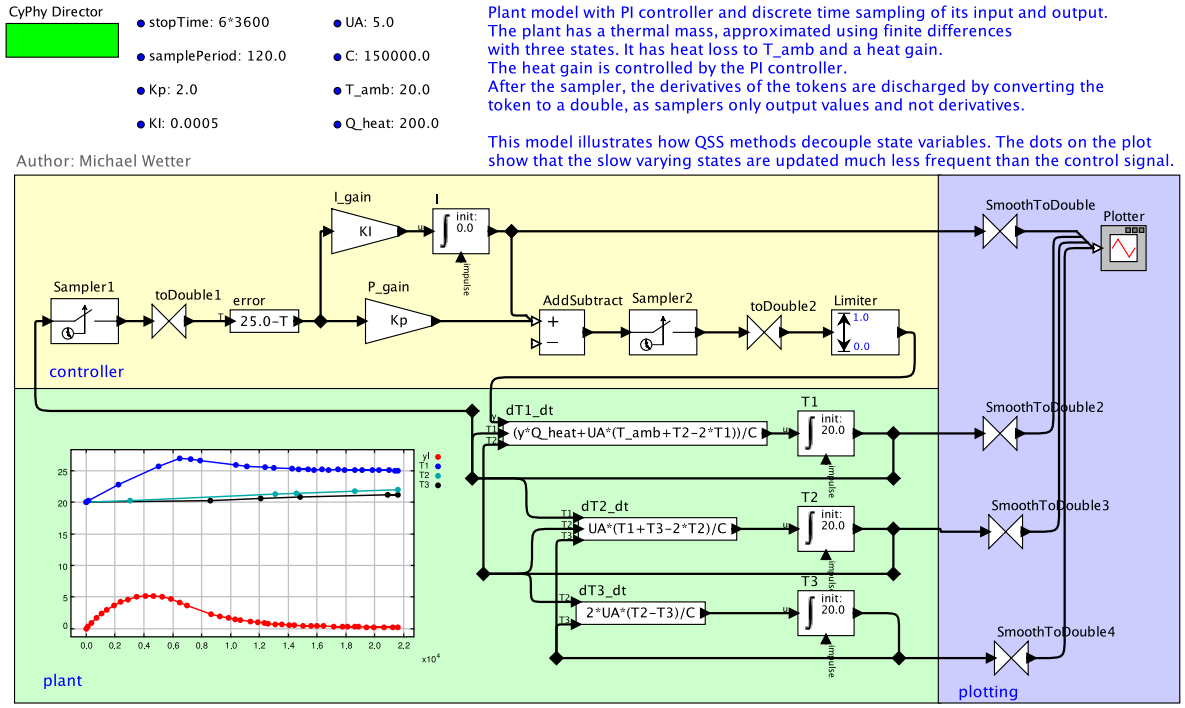



*Figure 1: Plant with feedback control loop and QSS solvers that solves* (1) *for six hours.*

lems. For building energy simulation, several open-source Modelica libraries are in active development, namely *AixLib* (Lauster et al., 2013), *Buildings* (Wetter et al., 2014), *BuildingSystems* (Nytsch-Geusen et al., 2012), and *IDEAS* (Baetens et al., 2012). These have been harmonized and refactored to extend from a common core Modelica library within the International Energy Agency project Annex 60 "New generation computational tools for building and community energy systems based on the Modelica and Functional Mockup Interface standards" conducted under the Buildings and Communities (EBC) Programme. SOEP will use HVAC and control models from these libraries. Eventually, Modelica will displace ERL as SOEP's control language.

## Quantized-State Systems (QSS)

SOEP uses recently-developed QSS methods to integrate ODEs. In a classical ODE simulator, a step-size control algorithm determines sample times, and a sample value is computed at those times for all states in the model. In constrast, in a QSS simulation, each state has its own sample times, and samples are processed using a discrete event engine in time-stamp order. The "next" sample time of each state is the time at which that state's value changes by a pre-determined tolerance called the quantum and is predicted using the state's time derivative. Higher-order QSS variants incorporate knowledge of higher-order derivatives for more refined predictions(Migoni et al., 2013).

For example, consider an ODE of the form $\dot{x}(t) = f(x(t), u(t))$, with initial conditions $x(0) = x_0$, where $x(\cdot) \in \mathbb{R}$ and $u(\cdot) \in \mathbb{R}^m$, for some $m \in \mathbb{N}$. The step size is computed based on the time rate of change of $\dot{x}(\cdot)$ – and possibly higher order derivatives – and the time when input signals $u(\cdot)$ change their value. This naturally decouples systems of ODEs, assigning short steps for fast dynamics like an integrator of a PI-controller, and large steps for slow dynamics such as the thermal mass of a storage tank.

For some systems, QSS yields efficient simulation by producing samples only when predicted state trajectories exceed the quantum. Moreover, state events can be scheduled using an explicit equation, avoiding iteration in time, which makes them very promising for HVAC and control simulation. Unlike classical ODE solvers, QSS solvers never require backtracking, greatly simplifying simulation.

*Example: Closed-loop plant control*

Figure 1 shows a simple model of a plant with a PI controller. The temperatures of the plant evolve as

$$C\,I\,\frac{dT(t)}{dt} = \begin{pmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & -2 & -2 \end{pmatrix} UA\;T(t) + \begin{pmatrix} UA\;T_{amb} + \dot{Q}\,y(t) \\ 0 \\ 0 \end{pmatrix}, \quad (1)$$

with initial temperatures $T(0) = (20, 20, 20)°\mathrm{C}$, where $C$ is the heat capacity, $I$ is the identity matrix, $UA$ is the heat conductance times area, $T_{amb}$ is the ambient temperature, $\dot{Q}$ is the heater capacity and $y(t)$

is the control signal. The PI controller equation is

$$y = K_p\, e(t) + K_I \int_0^t e(s)\, ds \qquad (2)$$

with output limiter, where $e(\cdot)$ is the difference between $T_{set} = 25°\mathrm{C}$ and the sampled value of $T_1$. The controller is sampled every 2 minutes. The plot in the lower left corner shows the trajectories of the temperatures $T(t)$ and the integral action $y_I(t)$. Each point at which the respective differential equation is evaluated is indicated by a dot. We used the QSS2 algorithm, and selected a large absolute and relative quantum of $10^{-2}$, which makes for better visualization of state variable recomputation events.

The plot shows that the QSS method evaluates the fast changing $y_I(t)$ 43 times, $T_1(t)$ 26 times and $T_2(t)$ and $T_3(t)$ each only 6 times as their time rate of change is smaller. These numbers of state updates are remarkable as the controller is sampled 181 times, but the QSS algorithm does not integrate its state if the input change is smaller than the quantum, and therefore it updates the states considerably less often. By comparison, simulating the same model using the classical Runge-Kutta 2(3) ODE solver requires 362 integration steps. With Runge-Kutta, the number of time steps is sensitive to the sampling interval. Such behavior is also commonly observed with solvers such as DASSL and Radau that are often used in Dymola (Wetter, 2009). In contrast, the number of QSS integration steps is not sensitive to the sampling interval, promising efficient simulation of control sequences.

## SPAWN-OF-ENERGYPLUS PROTOTYPE

This section describes the SOEP prototype and its implementation in Ptolemy II. It also evaluates SOEP using numerical benchmarks.

### Software architecture

Figure 2 shows the SOEP software architecture. Client applications – such as OpenStudio – write an EnergyPlus input data file (IDF) and consume EnergyPlus results. A translator – which has not yet been implemented – interprets the IDF to produce a list of modules, a listing of module output-to-input port connections, simulation start and stop time, and optional data such as a algorithms for time integration and solution of nonlinear equations, including their tolerances.

The SOEP master algorithm is implemented in Ptolemy II and available through the *CyPhySim* configuration (Brooks et al., 2015). The master algorithm instantiates the FMUs, evolves time in the FMUs, integrates the state variables, solves algebraic loops and synchronizes FMU inputs and outputs.

SOEP supports building operations in a number of ways. SOEP can participate in the Internet of Things by using Ptolemy II actors that implement a JavaScript execution environment in which scripts can safely be executed to communicate with sensors, actuators and services. Such JavaScripts can also run a service with a RESTful API to provide other devices with information from SOEP (Latronico et al., 2015). Also, FMUs simulated within SOEP can be imported into building automation systems such as Tridium Niagara (Nouidui and Wetter, 2014) for control or monitoring. Finally, the `BACnetReader` and `BACnetWriter` from the BCVTB (Wetter, 2011) could be ported to Ptolemy II to allow direct two-way run-time data exchange between SOEP and BACnet.

### Mathematical requirements

Building simulation leads to systems of ordinary and partial differential equations that are coupled to algebraic and discrete equations, and that have a wide range of numerical structure. Control and HVAC simulation has time constants in the order of seconds and are governed by equations that are often sparse and form an irregular solution graph. Controllers can trigger events if input signals cross a threshold. Heat conduction in solids requires solving ODEs with a band-diagonal structure that arised from the spatial discretization of the partial differential equations. Time constants are typically on the order of minutes to hours for walls, floor and ceiling slabs, but can be weeks to years for ground coupled heat transfer. It is not clear *a priori* what time integration algorithm works best for this variety of problems. Therefore, for SOEP we require that different time integration algorithms can be combined.

In SOEP, QSS methods can be combined with conventional ODE solvers and discrete time integration algorithms because Ptolemy II allows to hierarchically compose different models of computations within a single model. This hierarchical composition is done by instantiating an actor that contains its own director. For example, the discrete event domain can contain an actor that orchestrates the execution of its own actors according to the continuous time semantics. See (Lee and Zheng, 2007) for examples.

To simulate such systems efficiently and in a numerically robust way, SOEP require the following mathematical properties.

1. Models that describe physical processes must be formulated such that output and state trajectories are once continuously differentiable in the parameters of the model and the input signals. This is a basic requirement for Newton solvers, ODE solvers and to establish existence and uniqueness of a solution to the ODE.
2. Numerical solvers must expose their tolerances. This is required for use of the model within an optimization, or whenever nested solvers are present.
3. Computing modules must declare which outputs and state derivatives directly depend on which inputs.[2] This is required to discover the sparsity of the

[2] We say that an output $y$ depends directly on an input $u$ if the output is an algebraic function of the input $u$. If $y$ depends only on

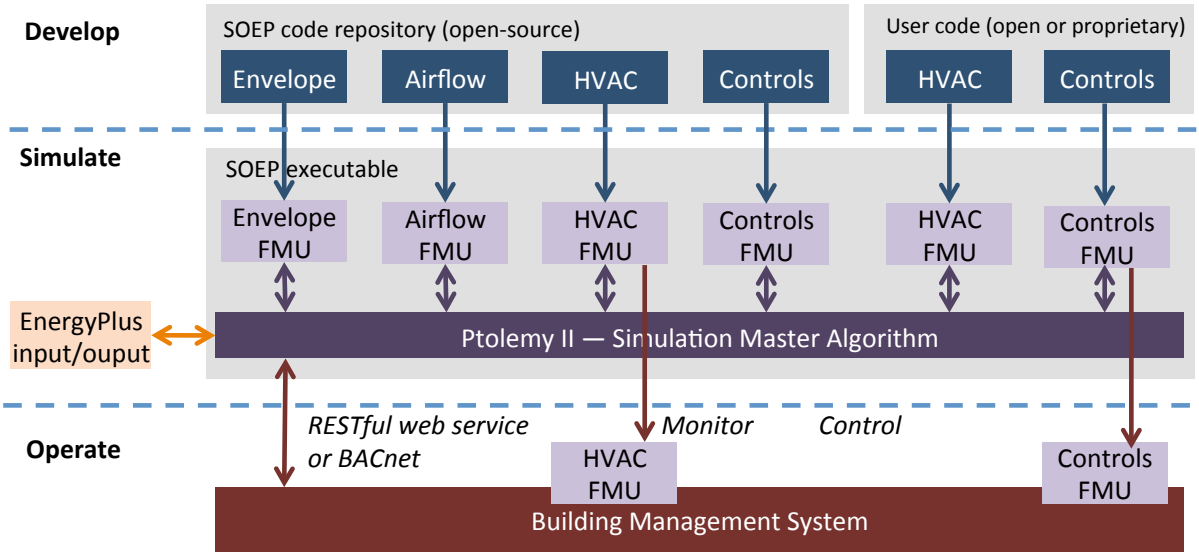



*Figure 2: Architecture of SOEP and its interaction with model libraries and building automation systems.*

model, to detect algebraic loops and to efficiently use QSS time integration algorithms.

4. All computing modules must support saving and reinitialization of state variables. This is required to solve optimal control problems, model linearizations and to implement certain parallel time integration algorithms.
5. All FMUs must be deterministic[3] and the master algorithm must guarantee a deterministic execution of any composition of deterministic FMUs.[4] This ensures that phenomena observed in a simulation are a result of the model structure and parameters, and not of arbitrary choices in the evaluation order of seemingly independent modules.

**Additions to Ptolemy II**

We added the following functionality to Ptolemy II for the SOEP prototype.

*FMI import*

We added FMI import capabilities in Ptolemy II. Models can be imported as FMUs for co-simulation or model-exchange. As FMUs for co-simulation provide their own time integration algorithm, this allows use of time integration methods such as the Conduction Transfer Functions that are currently used in EnergyPlus. FMUs for model-exchange do not contain a time integration algorithm. They can be imported for integration using either QSS methods or other ODE integrators that are provided through the continuous time domain of Ptolemy II. FMUs are encapsulated as Ptolemy II actors and can therefore be combined with other actors provided by Ptolemy II.

*QSS Solver*

The QSS solver is a standalone package which can be used to integrate the state derivatives of a system of initial-value ODEs. It contains the explicit QSS methods QSS1, QSS2 and QSS3, as well as the linearly-implicit methods LIQSS1 and LIQSS2 (Kofman, 2003; Migoni et al., 2013).

*QSS Director*

A new QSSDirector extends Ptolemy II's discrete-event model of computation to include the QSS solver. This director has additional fields for specifying the type of QSS solver and its quantum, which are used by actors that perform QSS integration.

*FMU QSS actor*

The new FMU QSS actor is used to import FMUs 2.0 for model-exchange. It unzips the FMU, parses the model description file, extracts the state derivatives which need to be integrated, creates input and outputs ports, integrates the state derivatives using a QSS solver, requests a time event for the next update of the state variables, and sends the state variables to the output ports.

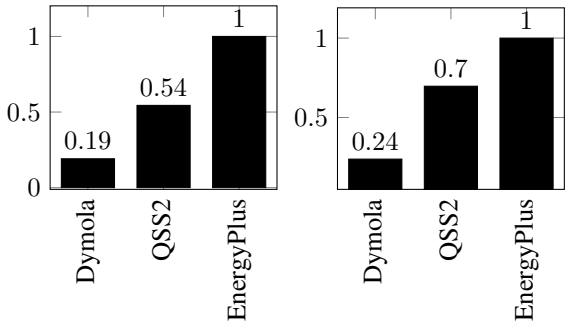
*Token that contains value and derivatives*

FMU QSS actors communicate via a new SmoothToken data type which contains the value and optionally the derivatives of a discrete time signal. The derivatives are used by second and third order QSS methods. They are also used to align the values of tokens that are produced at different times. For example, adding

a state variable $x$ (and, possibly, on a different input $\hat{u}$), then we say that $y$ does not depend directly on $u$, even if the time rate of change $dx/dt$ directly depends on $u$.

[3] A *deterministic FMU* is an FMU where the output values and states are uniquely defined given initial conditions, input values, and communication points.

[4] A *deterministic composition of deterministic FMUs* is one where for a valid sequence of communication points, given initial conditions and inputs from outside the composition, the values of outputs of the deterministic FMUs are uniquely defined. See Lee and Zheng (2005) for a rigorous definition of determinism.




*(a) One room.* *(b) Nine rooms.*

*Figure 3: Normalized run-time for annual simulation. Absolute runtime was 68 seconds for the one room model, and 488 seconds for the nine room model.*

the tokens $(x_1, \dot{x}_1) = (0,\, 2)$ produced at $t = 0$ and $x_2 = 0.2$ produced at $t = 1$ yields at $t = 1$ a token with value and derivative $(2.2,\, 2)$.

*Algebraic loop solver*

We also added a mechanism for specifying algebraic loops, and implemented algebraic loop solvers using successive substitution, a Newton-Raphson solver, and a homotopy method (Allgower and Georg, 2003). The SOEP model builder is given explicit control over the solution method and initial guesses.

## EVALUATION

To evaluate the SOEP protoype, we conducted annual simulations of i) a room with a window and a radiant heating system that is embedded in the concrete slab, and b) a 3-by-3 nine room configuration that couples common walls and floor/ceiling slabs. For comparison, we conducted the same simulations in Dymola 2015 FD01 and in EnergyPlus 8.2. The room geometry is identical to ASHRAE Standard 140 case 600FF (ASHRAE, 2007).

As we have not yet factored the envelope, lighting, and airflow modules out of EnergyPlus, we used Modelica Buildings Library models. We exported the models as five FMUs for the one-room case, and 45 FMUs for the nine-room case. We also built a version of the model in EnergyPlus 8.2 and a version that uses Modelica without any FMUs. As the Buildings library uses finite differences for the heat conduction, we configured EnergyPlus to use finite differences, with a time step of 3 minutes, which is the largest time step possible before the EnergyPlus simulation became unstable. We configured the number of states to be similar between the two implementations: Modelica had 88 states and EnergyPlus had 64 states. For the QSS simulation, we used QSS2 with a quantum of $10^{-3}$. To simulate the Modelica model, we used Dymola 2015 FD01 with the DASSL solver with its default tolerance of $10^{-4}$, as these values led to good agreement between the QSS2 and Dymola simulations.

In these benchmarks, FMUs with QSS was 30% and 46% faster than EnergyPlus, but slower than Dymola. Profiling showed that for the nine-room example, 33% of the computing time was spent in Ptolemy II, and 67% inside the FMUs. The 67% includes the Java-to-C overhead, which in some experiments made up 1.4% of run-time. Note that QSS2 called the FMUs twice at each step to approximate high order derivatives. Reducing the number of FMU calls, for example by using higher order methods, may further reduce run-time as fewer data tokens have to be sent to ports. Furthermore, parallelism in calling FMUs for simulation and derivative approximation has not yet been exploited.

## RELATED WORK

The idea for equation-based simulation is not new, even in the building space. In the 1980's and 1990's, researchers experimented with systems like SPARK (Simulation Problem Analysis and Research Kernel, (Sowell et al., 1986)) and NMF (Neutral Model Format (Sahlin and Sowell, 1989)).

Related work includes an FMU-based co-simulation environment for buildings developed by TU Dresden. Broman et al. (2013) present a co-simulation algorithm and required extensions for FMI for determinate composition of FMUs for co-simulation. Broman et al. (2015) define a suite of requirements for a future hybrid co-simulation version of FMI that allows mixing continuous time and discrete event signals.

Gunay et al. (2014) also investigated a discrete event simulation for whole building simulation. They report that fixed time step solvers used with adaptive occupant models can cause differences of $40\%$ in cooling energy if the time step is varied between 1 and 30 minutes, and $15\%$ in peak cooling demand if varied between $4$ and $60$ minutes. They concluded that proper event handling is important for simulation of adaptive occupancy models. Their numerical method can also be selected in the prototype that we present here.

## CONCLUSIONS AND FUTURE WORK

We have demonstrated the feasibility of our proposed redesign of EnergyPlus. Although the properties of the QSS methods – in particular the time scale separation and explicit event handling – imply high performance for building simulation, additional experiments are needed to test this hypothesis. Such experiments are on-going. The question of which ODE solvers work best for the various equations encountered in building simulation – controls, HVAC equipment dynamics, envelope heat and moisture transfer, and ground heat transfer – is also open.

Providing individual HVAC equipment and control models as FMUs has the advantage that their code can be provided to the user in compiled form, thereby avoiding compilation prior to the simulation. However, this component-level packaging complicates assembling components to systems because components have causal rather than acausal interfaces. It also precludes certain optimizations – the use of alias vari-

ables (*e.g.*, for mass flow rates that are common across all equipment models of an air flow leg), common subexpression elimination (*e.g.*, for computing medium properties that are shared by multiple equipment models) and block lower triangularization – from being performed at the system level. Also, as the size of the FMUs decreases and their number increases, the computations required to synchronize the FMUs increase. There would seem to be an optimum size that allows sparse evaluation of FMUs while minimizing synchronization overhead. See also Wetter et al. (2015) for a discussion.

We have not yet explored the potential for parallel computation. This could be employed in evaluating higher order derivatives and within the discrete event simulation.

Further work also remains to be done on algebraic loop solving and on event handling if events are triggered inside an FMU.

Finally, much R&D remains in the implementation and optimization of FMUs and QSS methods for whole building simulation.

## ACKNOWLEDGMENTS

We thank Christopher Brooks from UC Berkeley for his contributions to the development of the required extensions of Ptolemy II, and Xiufeng Pang from LBNL for his assistance in creating the EnergyPlus model of the radiant system.

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

This work emerged from the Annex 60 project, an international project conducted under the umbrella of the International Energy Agency (IEA) within the Energy in Buildings and Communities (EBC) Programme. Annex 60 will develop and demonstrate new generation computational tools for building and community energy systems based on Modelica, Functional Mockup Interface and BIM standards.

## REFERENCES

Allgower, E. L. and Georg, K. 2003. *Introduction to Numerical Continuation Methods*, volume 45 of *Classics in Applied Mathematics*. SIAM.

ASHRAE 2007. ANSI/ASHRAE Standard 140-2007, Standard method of test for the evaluation of building energy analysis computer programs.

Baetens, R., De Coninck, R., Van Roy, J., Verbruggen, B., Driesen, J., Helsen, L., and Saelens, D. 2012. Assessing electrical bottlenecks at feeder level for residential net zero-energy buildings by integrated system simulation. *Applied Energy*, 96(Special issue on Smart Grids, Renewable Energy Integration, and Climate Change Mitigation - Future Electric Energy Systems):74–83.

BLAST 1999. *BLAST 3.0 Users Manual*. University of Illinois, Urbana-Champaign, IL, Department of Mechanical and Industrial Engineering, Building Systems Laboratory.

Bliudze, S. and Furic, S. 2014. An operational semantics for hybrid systems involving behavioral abstraction. In *Modelica Conference*, Lund, Sweden.

Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., Neumerkel, D., Olsson, H., Peetz, J.-V., and Wolf, S. 2011. The functional mockup interface for tool independent exchange of simulation models. In *Proc. of the 8-th International Modelica Conference*, Dresden, Germany. Modelica Association.

Broman, D., Brooks, C., Greenberg, L., Lee, E. A., Masin, M., Tripakis, S., and Wetter, M. 2013. Determinate composition of fmus for co-simulation. In *Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on*, pages 1–12.

Broman, D., Greenberg, L., Lee, E. A., Massin, M., Tripakis, S., and Wetter, M. 2015. Requirements for hybrid cosimulation standards. In *18th International Conference on Hybrid Systems: Computation and Control*. ACM Press.

Brooks, C., Lee, E. A., Lorenzetti, D., Nouidui, T. S., and Wetter, M. 2015. Demo: CyPhySim – a cyber-physical systems simulator. In *18th International Conference on Hybrid Systems: Computation and Control*. ACM Press.

Cassandras, C. G. 1993. *Discrete Event Systems, Modeling and Performance Analysis*. Irwin.

Cellier, F. E. and Kofman, E. 2006. *Continuous System Simulation*. Springer.

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., and Glazer, J. 2001. EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings*, 33(4):443–457.

Fishman, G. S. 2001. *Discrete-Event Simulation: Modeling, Programming, and Analysis*. Springer-Verlag.

Gunay, H. B., O'Brien, W., Beausoleil-Morrison, I., Goldstein, R., Breslav, S., and Khan, A. 2014. Coupling stochastic occupant models to building performance simulation using the discrete event system specification formalism. *Journal of Building Performance Simulation*, 7(6):457–478.

Kofman, E. 2003. Quantization-based simulation of differential algebraic equation systems. *SIMULATION*, 79(7):363–376.

Kofman, E. and Junco, S. 2001. Quantized-state systems: A DEVS approach for continuous system simulation. *Transactions of The Society for Modeling and Simulation International*, 18(1):2–8.

Latronico, E., Lee, E. A., Lohstroh, M., Shaver, C., Wasicek, A., and Weber, M. 2015. A vision of swarmlets. *IEEE Internet Computing, Special Issue on Building Internet of Things Software*, 19(2):20–29.

Lauster, M., Fuchs, M., Teichmann, J., Streblow, R., and Müller, D. 2013. Energy simulation of a research campus with typical building setups. In Roux, J. J. and Woloszyn, M., editors, *Proc. of the 13-th IBPSA Conference*, pages 769–775.

Lee, E. A. 1999. Modeling concurrent real-time processes using discrete events. *Annals of Software Engineering*, 7:25–45.

Lee, E. A. and Zheng, H. 2005. Operational semantics of hybrid systems. In Morari, M. and Thiele, L., editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 25–53. Springer Berlin Heidelberg.

Lee, E. A. and Zheng, H. 2007. Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *Proceedings of the 7th ACM &Amp; IEEE International Conference on Embedded Software*, EMSOFT '07, pages 114–123, New York, NY, USA. ACM.

Matsikoudis, E. and Lee, E. A. 2013. On fixed points of strictly causal functions. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume LNCS 8053, pages 183–197, Buenos Aires, Argentina. Springer-Verlag.

Mattsson, S. E., Otter, M., and Elmqvist, H. 1999. Modelica hybrid modeling and efficient simulation. In *38th IEEE Conference on Decision and Control*, pages 3502–3507, Phoenix, AZ. IEEE.

Migoni, G., Bortolotto, M., Kofman, E., and Cellier, F. 2013. Linearly Implicit Quantization-Based Integration Methods for Stiff Ordinary Differential Equations. *Simulation Modelling Practice and Theory*, 35:118–136.

Nouidui, T. S. and Wetter, M. 2014. Linking simulation programs, advanced control and FDD algorithms with a building management system based on the Functional Mock-Up Interface and the building automation Java architecture standards. In *ASHRAE/IBPSA-USA Building Simulation Conference*, pages 49–55, Atlanta, GA. IBPSA-USA.

Nytsch-Geusen, C., Huber, J., and Ljubijankic, M. 2012. Building and plant simulation strategies for the design of energy efficient districts. In *Computational Design Modelling*, pages 171–180.

Sahlin, P. and Sowell, E. F. 1989. A neutral format for building simulation models. In *Proceedings of the Second International IBPSA Conference*, pages 147–154, Vancouver, BC, Canada.

Sowell, E. F., Buhl, W. F., Erdem, A. E., and Winkelmann, F. C. 1986. A prototype object-based system for HVAC simulation. Technical Report LBL-22106, Lawrence Berkeley National Laboratory.

Walton, G. N. and Dols, W. S. 2013. CONTAM 3.1, user guide and program documentation. Technical Report NISTIR 7251, National Institute of Standards and Technology.

Wetter, M. 2009. Modelica-based modeling and simulation to support research and development in building energy and control systems. *Journal of Building Performance Simulation*, 2(2):143–161.

Wetter, M. 2011. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 4(3):185–203.

Wetter, M., Fuchs, M., and Nouidui, T. S. 2015. Design choices for thermofluid flow component and systems that are exported as Functional Mockup Units. Accepted: *11-th International Modelica Conference*, Paris, France. Modelica Association.

Wetter, M., Zuo, W., Nouidui, T. S., and Pang, X. 2014. Modelica Buildings library. *Journal of Building Performance Simulation*, 7(4):253–270.

Winkelmann, F. C. and Selkowitz, S. 1985. Daylighting simulation in the DOE-2 building energy analysis program. *Energy and Buildings*, 8:271–286.

Zeigler, B. P. and Lee, J. S. 1998. Theory of quantized systems: Formal basis for DEVS/HLA distributed simulation environment. In *SPIE Conference on Enabling Technology for Simulation Science*, volume SPIE Vol. 3369, pages 49–58, Orlando.

Zeigler, B. P., Praehofer, H., and Kim, T. G. 2000. *Theory of Modeling and Simulation*. Academic Press, Inc., Orlando, FL, USA, 2nd edition.