2020 Building Performance Modeling Conference and
SimBuild co-organized by ASHRAE and IBPSA-USA

URBANopt: AN OPEN-SOURCE SOFTWARE DEVELOPMENT KIT FOR COMMUNITY AND URBAN DISTRICT ENERGY MODELING

Rawad El Kontar, Ben Polly, Tanushree Charan,
Katherine Fleming, Nathan Moore, Nicholas Long, David Goldwasser

National Renewable Energy Laboratory, Golden, CO

ABSTRACT

Urban building modeling tools are developing rapidly; these tools use emerging simulation workflows for specific urban environmental design tasks, such as assessing the impacts of energy efficiency technologies at a district scale. However, with the emergence of new environmental design tasks, addressing all possible use cases and tasks is challenging and cannot be covered by a single tool. Urban-scale analysis at this level of complexity often requires linking multiple emerging tools, rather than using a single tool, to adequately evaluate a variety of possible fields in urban environmental design. To achieve this, flexible platforms are needed to support multiple input formats (e.g., geometric and non-geometric building properties), enabling the mapping of such inputs to underlying simulation engines.

This paper provides an overview of the open-source URBANopt Software Development Kit (SDK) for modeling high-performance buildings and energy systems at a district scale. URBANopt's flexible SDK is composed of several modules that can be customized to integrate with other tools and generate new workflows to perform urban environmental design tasks, such as capturing interactions between individual buildings, district energy systems, distributed energy resources, and the electric distribution grid.

We describe the functionality of the core SDK modules in URBANopt (called Core Gem, GeoJSON Gem, and Scenario Gem) and discuss the flexibility of these modules as a means of integration with a variety of tools. We also document and demonstrate technical details of writing and combining new modules to create customized workflows. Finally, we present a case study that uses the URBANopt SDK to model a hypothetical mixed-use urban project and simulate various scenarios to meet district energy performance goals.

INTRODUCTION

In the United States, buildings consume about 40% of U.S. primary energy and 75% of grid electricity (US-EIA 2018). Reducing energy use in buildings is important to achieving energy and environmental goals in the built environment. Many cities, utilities, states, and other entities in the United States are setting energy and emissions reductions goals. So, sustainable urban design that effectively integrates energy efficiency and renewable energy is of growing interest to a variety of stakeholders.

In the meantime, a large and growing amount of data is being collected about the built environment. Internet of Things (IoT)-enabled sensors can gather information specific to building energy behaviors, while advanced light detection and ranging (LIDAR) technologies provide 3D data of buildings with high resolution and accuracy. This unprecedented data availability provides significant opportunities for improved performance of buildings and their interactions with the broader energy system.

With more access to building data, energy modelers can develop highly detailed models and perform a wide range of analysis. Multiple open-source environmental simulation engines have been developed over the past few decades (Crawley et al. 2008), and analysts are combining data with simulations to investigate questions with increasing granularity and accuracy. These simulation engines target a variety of environmental issues. For example, building energy systems performance can be analyzed using tools like EnergyPlus™, ESP-r, IDA ICE, TRNSYS; lighting and solar radiation analysis can be performed using tools like the ray-tracing program Radiance; wind and water flow analysis can be performed using CFD softwares such as Fluent and OpenFoam; renewable energy systems can be modeled using tools such as Renewable Energy Integration & Optimization tool (REopt™); and grid network design and modernization can be conducted using the Open Distribution System Simulator (OpenDSS) (Allegrini et al. 2015) (Simpkins et al. 2014) (OpenDSS 2015).

Researchers are focused on leveraging collected data and simulation engines to better inform practices and energy interactions in the built environment. As a result, urban building energy models (UBEMs) were introduced to simulate various built environment performance measures and predict future evolution of districts and communities (Reinhart and Davila 2016). Using bottom-up modeling approaches, UBEMs can explore new technologies and

analyze interventions by running simulations characterized by the geometric and non-geometric information of buildings in an urban environment (Davila, Reinhart, and Bemis 2016).

Several UBEM software programs have been developed using back-end simulation engines and models: CitySim was developed as simulation software using a reduced order model (Robinson et al. 2009). Nouvel et al. used the ISO 13790 heat balance algorithm to simulate the energy behavior for more than 14,000 buildings in Germany (Nouvel et al. 2014). The Urban Modeling Interface (UMI) was developed to evaluate the environmental performance of neighborhoods and cities with respect to operational and embodied energy use, walkability, and daylighting potential (Reinhart et al. 2013). UMI relies on EnergyPlus and Radiance as underlying simulation engines. District Zero is being developed as a decision-making tool for communities using EnergyPlus and water/waste algorithms for simulation (Ellis 2018).

As more information is collected, new environmental design tasks are emerging. This has driven efforts to add new capabilities to existing UBEMs and develop new tools and software. For example, the micro-climate simulation software ENVI-met was developed to assess the impact of buildings and vegetation on micro-climatic conditions such as outdoor thermal comfort, wind flow, air pollution, and solar energy gains (Bruse 2004). CitySim capabilities are being extended to model water and transportation systems. A new plugin is being developed for UMI, called HARVEST, to inform building-integrated agriculture implementation in an attempt to minimize water and energy consumption while maximizing crop yields (Benis, Reinhart, and Ferrão 2017). Regarding district systems, SynCity introduced an integrated tool for holistic urban energy system modeling (Keirstead et al. 2010), while KULeuven IDEAS lib and LBNL District lib added district-level simulation libraries to Modelica (Wetter et al. 2014). With the development of advanced modeling to capture more interactions in the built environment, a new taxonomy for UBEMs was defined based on the building-to-building and building-to-other-system interactions (Reyna et al. 2018). Each of these interactions is expected to be addressed with a unique set of analysis workflows and applications.

To reiterate, urban modeling tools are developing rapidly. Many of these tools have relied on back-end simulation engines to perform specific design tasks. However, with the emergence of new design tasks and simulation workflows, no single tool has, nor is intended to have, the capability of performing all possible tasks and use cases. Analysis of problems at this scale and complexity often requires linking together multiple tools, rather than the use of a single tool, to adequately evaluate a variety of possible fields in urban environmental design.

To help address these needs, flexible platforms can be developed to support the connection of multiple simulation tools. For example, simulation engines and software development kits—such as EnergyPlus™ and OpenStudio®—can be made more accessible to urban modeling if they are packaged with additional open-source workflows and analytical capabilities that enable multibuilding analysis and facilitate integration with other UBEM tools. Such platforms can support multiple formats of building-related information and enable the mapping of such inputs to various simulation engines. URBANopt™ (Polly et al. 2016) is one such platform, and is the focus of this paper.

URBANopt is currently being developed as an open-source software development kit (SDK). URBANopt is a simulation platform to perform environmental performance analysis within a geographically cohesive area smaller than a city (e.g., city block, district, or community). Instead of a monolithic piece of software, URBANopt is a flexible SDK composed of several modules that can be customized to integrate with other tools and generate new workflows to perform urban environmental design tasks, such as capturing interactions between individual buildings, district energy systems, distributed energy resources, and the electric distribution grid.

The paper gives an overview of the URBANopt SDK, focusing on the base structure of the SDK that enables multibuilding efficiency scenario analysis. We describe the functionality of the core SDK modules in URBANopt (which are called Core Gem, GeoJSON Gem, and Scenario Gem) and discuss the flexibility of these modules as a means of integration with a variety of tools. We also document and demonstrate technical details of writing and combining new modules to create customized workflows. Finally, we present an example project that uses the URBANopt SDK to model a hypothetical mixed-use urban project and simulate various scenarios.

The paper is organized as follows: Section 2 provides an overview of URBANopt. Section 3 discusses the software architecture. Section 4 describes the GeoJSON Gem functionality. Section 5 describes the Scenario Gem functionality. Then the connection of these modules to create and run an example project is demonstrated through the example workflow described in Section 6. Finally, Section 7 discusses the released example project and ongoing work.

URBANopt OVERVIEW

URBANopt is not a front-end user interface, but rather an SDK focused on underlying calculations and workflows for district-scale energy analysis. The URBANopt SDK provides modules to map different data formats to inputs for multiple simulation engines. Commercial software de-

Table 1: URBANopt Terms and Definitions

Feature	A Feature is a single object in a district-scale energy analysis, such as a building, district system, or transformer. Each Feature has an associated geometry definition of some type (e.g., position, 2D footprint, 3D mass). Each Feature has an associated Feature ID and Feature Type (e.g., Building, District System) as well as other properties related to the Feature Type (e.g., District System Type, Building Type(s), Number of Stories, Square Footage)
FeatureFile	A FeatureFile includes data for all Features in a given Scenario. The same FeatureFile may be used by multiple Scenarios, but a FeatureFile does not include information about multiple Scenarios. The FeatureFile could be written by a third-party application or user interface. URBANopt initially supports the GeoJSON format via the urbanopt-geojson-gem.
Simulation Mapper	A Simulation Mapper translates the information in a Feature to simulation input. This Mapper creates a simulation input using a template that bundles a set of OpenStudio Measures. It uses these measures to create an input defining specific energy model characteristics.
Scenario	A Scenario represents one potential realization of a district for performance analysis purposes. Each Scenario references a FeatureFile and assigns a Simulation Mapper Class to each Feature in that file. In this way, all Features in the FeatureFile can be translated to simulation input files.
Project	A Project includes multiple Scenarios for analysis. Each Scenario in the Project may reference different FeatureFiles with different Features. This allows Scenarios to have varying geometry across the Project. Results comparisons done at the Project level are typically concerned with comparing one Scenario to another.

velopers could leverage the URBANopt SDK to develop end-user-facing tools, applications, user interfaces, and web services for building energy modeling and analysis. These tools can then be employed in real-life applications to better inform the built environment stakeholders on their decisions, where urban master planning firms, architecture firms, energy consultancies are examples of potential users.

URBANopt SDK open-source modules can operate as part of the larger ecosystem of urban design and modeling tools. Commercial software developers can use existing URBANopt modules, customize URBANopt modules, or create new ones to help implement the desired workflows for their end-user tools. Other users of the SDK could include researchers looking to create customized workflows to perform specific environmental design tasks.

To support modular reuse and collaboration, URBANopt modules have clear inputs and outputs. Any module can be replaced by another module with the same inputs and outputs. URBANopt modules can also be written in a variety of software languages (Ruby, Python, C++, etc.) while ensuring the interoperability between modules.

The initial URBANopt SDK modules focus on modeling high-performance buildings and energy systems. These modules are developed in Ruby scripting language and leverage OpenStudio SDK modules (Weaver et al. 2012). However, URBANopt includes a Command Line Interface (CLI), shown in Figure 1, which supports developers whose tools may not have access to Ruby. Specific URBANopt terms documented in Table 1 are consistently used because they describe main elements in URBANopt modules.

URBANopt SOFTWARE ARCHITECTURE

Due to high variability in district-scale energy analysis, the user is responsible for combining the URBANopt modules needed to implement the desired workflow for their project. This section describes an example project, combining existing OpenStudio modules with the developed URBANopt modules (Core Gem, GeoJSON Gem, and Scenario Gem).

Figure 1 illustrates an example project where modules are combined to implement a district-scale energy analysis workflow. Each block shown in Figure 1 represents a different module, each developed and managed in separate source code repositories. The dashed connection represents an example of a new module that can be developed and integrated within this platform to expand its capabilities.

The Core Gem hosts the FeatureFile Class. Conceptually, the FeatureFile could be in different formats (e.g., GeoJSON, CityGML) and should describe properties of each Feature, such as location, floor area, number of stories, building type, cooling source, etc. Developed modules have to use the FeatureFile class; because of this, this class is defined in a separate module—the Core Gem. The Gem introduces flexibility in the SDK architecture, as it allows

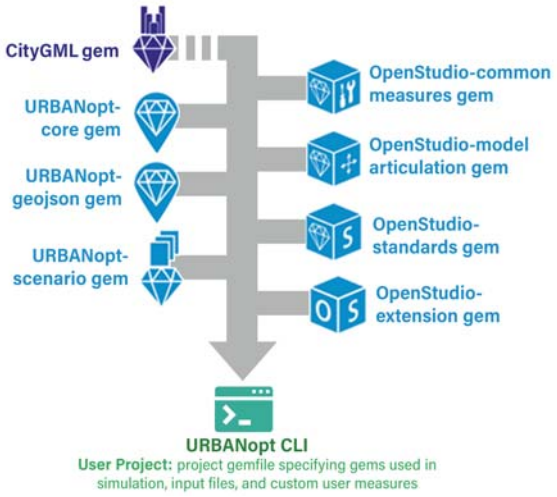

Figure 1: Software architecture for an example URBANopt project

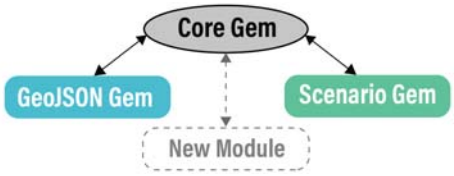

Figure 2: Core Gem functionality

the development of new modules that are independent of other modules. Figure 2 illustrates this functionality.

The FeatureFile format currently supported is GeoJSON. The GeoJSON Gem is developed to translate information in a GeoJSON format to simulation inputs. Third-party modules can be leveraged to support other FeatureFile formats. For example, a new CityGML module could be developed and used to map data from CityGML files to simulation inputs (potential connection shown by dashed line in Figure 1). The URBANopt Scenario Gem module allows the user to specify, run, and compare multiple district-scale energy scenarios. The OpenStudio Common Measures Gem, OpenStudio Model Articulation Gem, OpenStudio Standards Gem, and OpenStudio Extension Gem are part of the OpenStudio SDK modules and provide the users with a set of OpenStudio measures that they can assign to Features.

When working on a project, the user can specify the exact version of each module to use in their project. These modules can be executed using the URBANopt Command Line Interface (CLI), which combines all the modules and runs them using defined commands. The URBANopt CLI enables the execution of URBANopt calls from programs that do not directly support Ruby.

The following sections describe the functionality of the GeoJSON Gem and Scenario Gem in more detail. For more information on OpenStudio SDK modules, users can refer to OpenStudio documentation (`https://nrel.github.io/OpenStudio-user-documentation/`).

GeoJSON GEM

GeoJSON is a commonly used format for describing geospatial data related to the built environment. Notably, Microsoft has published GeoJSON files, including all building footprints for each state in the United States, at `https://github.com/microsoft/USBuildingFootprints`. There are several existing tools that can read and write GeoJSON, making it an easy format to work with; see `http://geojson.io`. The URBANopt GeoJSON Gem translates information in GeoJSON format into OpenStudio models for simulation with EnergyPlus. To achieve this functionality, this module first checks the validity of the GeoJSON file, ensuring that all the required properties exist and are in the correct format. Then it runs methods to parse this information and translate it to the energy model input.

GeoJSON Gem Schemas

To be able to validate the FeatureFile, the GeoJSON gem defines two main schemas:

(a) *A standard JSON schema for the GeoJSON format FeatureFile:* This schema ensures that each Feature in the GeoJSON is associated with geometry that has a valid geospatial coordinate system. This helps to ensure that the spatial requirements of each Feature are considered in district design.

(b) *An URBANopt GeoJSON schema:* This schema places additional restrictions on Feature geometry and requires non-geometric properties that are not presented in the standard JSON schema (e.g., Feature type). Because URBANopt requires additional properties to describe information about buildings, district systems, and other energy-related assets, each Feature is assigned a type property (e.g., "Building," "District System," "Region"). Sub-schemas exist for each of these Feature types to define the valid properties that describe them.

A FeatureFile is compared to both schemas to check its validity and ensure that all the required Feature properties are properly defined. Figure 3 shows an example of a valid URBANopt GeoJSON FeatureFile that includes the required properties of a building feature such as building type, floor area, and ID.

GeoJSON Gem methods

In addition to the schemas, the GeoJSON Gem includes methods used to achieve the following key capabilities:

(1) *Validate a GeoJSON file:* Given an input GeoJSON file, this gem has methods to check each Feature and verify that it meets additional requirements for data sup-

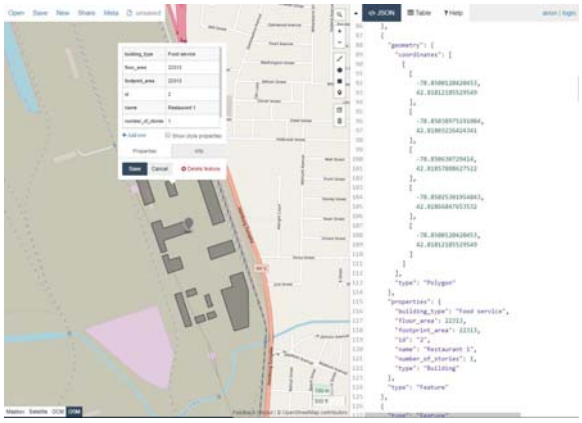

Figure 3: Valid URBANopt GeoJSON FeatureFile

ported or required by the appropriate URBANopt GeoJSON Gem sub-schema.

(2) *Identify nearby Features for shading calculations:* Given a Feature of interest, this method finds all other Features that could potentially shade the Feature of interest during an annual simulation.

(3) *Translate a "Building" Feature to OpenStudio Shading Objects:* This method creates OSM shading objects to represent GeoJSON Features of certain types (e.g., "Building").

(4) *Translate a "Building" Feature to an OpenStudio Model:* This method reads each Feature from the FeatureFile, converts geospatial coordinates to cartesian coordinates, performs auto-zoning to establish perimeter and core zones as needed, and extrudes geometry from 2D footprints to 3D surfaces. The properties from each Feature are then translated to OpenStudio model (OSM file) inputs leveraging methods in the OpenStudio Model Articulation Gem, Common Measures Gem, Standards Gem, and Extension Gem.

SCENARIO GEM

The Scenario Gem is an essential part of the URBANopt SDK because it has the functionality to construct multiple district-scale energy scenarios, run the simulation models in each Scenario, and post process results. To achieve this functionality, this Gem is composed of four main classes: (1) Simulation Mapper, (2) Scenario CSV, (3) Scenario Runner, and (4) Scenario Post Processor. In addition, this Gem includes a Reports schema that defines the output results of a Scenario. The connection of these components is described in the general workflow below and followed by subsections that describe each of the main components in more details.

Each user-defined URBANopt Project can include multiple URBANopt Scenarios. Each Scenario takes as input a FeatureFile defining multiple district Features (e.g.,

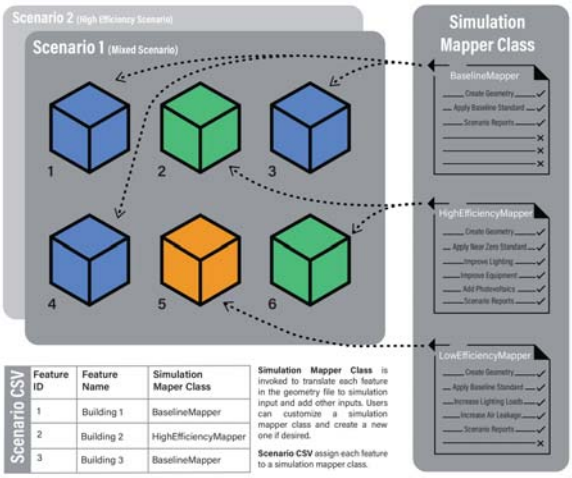

Figure 4: Mapping a Simulation Mapper Class to Features in a Scenario

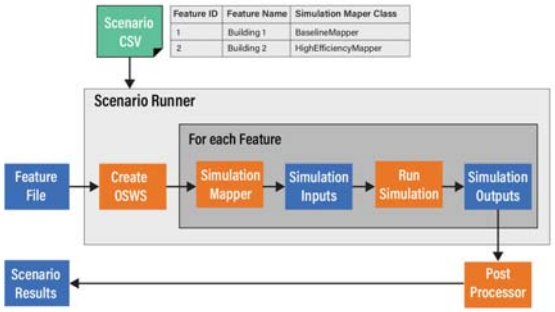

Figure 5: A simplified outline of the Simulation Runner workflow

building, district system) and a Scenario CSV file describing the mapping of those Features to simulation inputs, which are packaged in a Simulation Mapper. Each row in the Scenario CSV file assigns one Feature to a Simulation Mapper. Multiple Simulation Mappers can be created and assigned to Features in a Scenario. This process is illustrated in Figure 4.

After constructing a Scenario via a Scenario CSV, the Scenario Runner creates an OpenStudio workflow file (OSW) for each Feature using the specified Simulaton Mapper. The Scenario Runner then runs each OSW using the OpenStudio CLI to generate Feature results. After all the Features have been simulated, the Scenario Post Processor accesses the simulation outputs for all Features in a Scenario to generate Scenario results. This process is illustrated in Figure 5.

Simulation Mapper

A Simulation Mapper is an instance of a Simulation Mapper Class. The Simulation Mapper translates each Feature in a FeatureFile to simulation input leveraging measures from the GeoJSON Gem and other OpenStudio Gems. The first type of simulation input format that URBANopt supports is the OpenStudio Workflow (OSW) format, which can be run using the OpenStudio CLI. This OSW file refers to a set of measures that are used to characterize a building model.

Each Simulation Mapper is first defined by one OSW file that includes GeoJSON measures and other OpenStudio measures. These measures are applied to create an OpenStudio Model. The Mapper Class has the ability to enable or disable any measure within its OSW file and can set arguments to the measures. Finally, each Simulation Mapper Class supporting an OSW file implements a method that translate all Features in a scenario to OSW files.

Simulation Mapper Classes can be defined at the user-specific Project level. This gives great control to define custom simulation behavior for specific Projects. For example, a Simulation Mapper may decide that all Building Features over three stories tall must have photovoltaics and high-efficiency equipment. Users can create a completely different Simulation Mapper that defines the characteristics of another type of building.

Scenario CSV

A Scenario is defined by assigning a Simulation Mapper Class to each Feature in a FeatureFile. Users can create more than one Scenario by assigning different Simulation Mapper Classes to Features in the Scenario. Users can also create multiple FeatureFiles, which also vary between Scenarios. Therefore, a ScenarioBase Class has been developed to assign Mapper Classes to Features. The Scenario CSV Class inherits from a ScenarioBase Class to provide an easy way to perform these differing assignments via a simple comma-separated values (CSV) file. This Scenario CSV file format can be written manually using tools like Microsoft Excel or programmatically if the Scenario generation is done via software. Users can also write a new Class using ScenarioBase to achieve this functionality via a different file format other than the CSV file.

For illustration, imagine a FeatureFile that has three Features. The user can construct two Simulation Mapper Classes: a Baseline Mapper that translates a Feature to simulation inputs using baseline assumptions, and a High Efficiency Mapper that translates a Feature to simulation inputs using high-efficiency assumptions. Then users can use these two Mapper Classes to specify different Scenarios using the Scenario CSV file. For example, in the first scenario, all buildings are assigned to the Baseline Mapper; second scenario has all buildings assigned to the High Efficiency Mapper; and the third scenario, demonstrated in Figure 4, can have a mix of baseline and high-efficiency buildings using both Mappers.

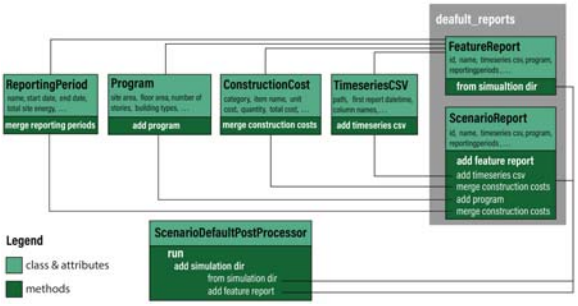

Figure 6: Scenario Post Processor Architecture

Scenario Runner

The Scenario Runner provides functionality to run a Scenario. The initial URBANopt class is a Scenario Runner OSW, which uses an OpenStudio Extension Gem to run all the OSW files for each scenario. This class is initialized with a root directory containing a Gemfile, which specifies the OpenStudio Extensions and other gems to use during simulation.

The Simulation Mapper Class decouples the Scenario specification from the FeatureFile format. The specified OpenStudio Extension Gems include FeatureFile readers, which can be used by the Simulation Mapper Classes, as well as OpenStudio Measures, which can be used in the simulation workflow. This allows the same Scenario Runner functionality to run simulations for multiple FeatureFile formats.

The Scenario Runner also has functionality to run Features in stages; results from one stage are available when the next stage is run. In cases where different Feature types exist in the FeatureFile (e.g., Building, District System, Transformer Features), a staged runner is needed. For example, when District System Feature simulations are run, they are able to access simulation results for all Building Feature simulations.

The initial implementation of Scenario Runner supports running simulations on the local computer with the functionality to run simulations in parallel. This way, users can specify the number of cores they want to utilize based on the number of Features and availability of cores for computation.

Scenario Post Processor and Reports

The Scenario Gem hosts schemas that define the output results for each Feature, defined as Feature Reports, and the aggregated results for the whole Scenario, defined as Scenario Reports. These schemas include specific at-

tributes that are derived from the simulation outputs such as program, construction costs, energy consumption, energy generation and other metrics related to building energy performance. Users can extend the schemas to include additional results related to building and district energy performance.

To generate Feature Reports, an OpenStudio Reporting Measure is used to query and report specific output data from an OpenStudio simulation of each Feature. The default URBANopt reporting measure is the default_feature_reports. This Default Reporting Measure will generate Feature-level results needed by the Default Scenario Post Processor. This will require writing result files with similar attributes as reported in the Scenario Reports. This aspect enables the Default Scenario Post Processor to easily sum results across Features. Users can create their own OpenStudio Reporting Measure and add them to the Simulation Mapper to generate customized simulation reports when the Scenario Runner OSW is invoked. Users can also request results for different reporting frequencies or query and report additional outputs that are important for their own projects (e.g., reporting thermal comfort results).

The Scenario Post Processor aggregates Feature reports into Scenario-level results by leveraging methods of the Feature report classes. Each of these classes corresponds to an attribute in the defined results schema. It is important to note that each attribute of the Feature report should have a specific aggregation method. For example, if users decide to customize the reporting measure to report results for a new attribute(e.g., construction costs), additional methods should be developed to allow the Post-Processor to aggregate the results for the added attribute.

Figure 6 describes the architecture of this Post Processor and illustrates the class hierarchy and the main methods currently leveraged by this Post Processor to aggregate Feature reports into a Scenario report. Users can edit these methods or add new methods that extend or customize the Post Processor functionality to report and aggregate new properties of interest.

WORKFLOW

A developed workflow can be run using URBANopt CLI commands or Ruby Rake tasks. In the below example, Rake tasks are developed to run and post process each of the defined Scenarios. Figure 7 illustrates the process of running the workflow using the run_all and post_process_all tasks.

A GeoJSON FeatureFile is used and includes multiple Features. The user specifies a Simulation Mapper Class for each Feature in the FeatureFile using a Scenario CSV file. The Scenario Runner creates an OSW file for each Feature using the specified Simulation Mapper Class. The

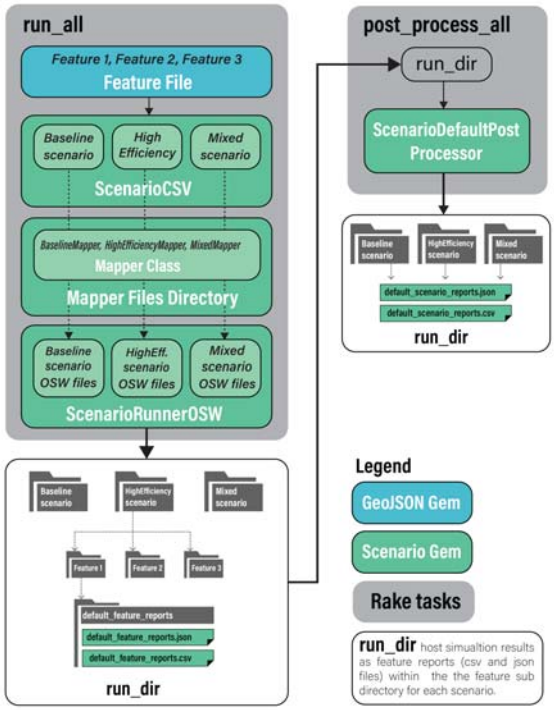

Figure 7: Main workflow

Scenario Runner then runs each OSW using the OpenStudio CLI and saves default Feature reports to sub directories in the scenario run directory. After all of the Features have been simulated, a ScenarioDefaultPostProcessor is used to aggregate the results. This post processor has access to simulation results for all Features in the Scenario. The ScenarioDefaultPostProcessor writes Scenario reports that contain Scenario-level results (e.g., total energy use, total energy cost) at the top of the scenario run directory.

Multiple customization and additions can be implemented to modify this workflow. In this example, the ScenarioDefautPostProcessor is used to write a JSON and CSV file. However, other Scenario Post Processors can be developed and used. For example, the URBANopt GeoJSON Gem can define a Scenario Post Processor that puts simulation results back into GeoJSON format for visualization. Also, this post processor—along with the Reporting Measure—can be customized to report and aggregate additional results (e.g., construction cost, number of occupants)

Regarding the Mapper Class, new measures can be added to an OSW file and then modified, and the arguments for these measures can be set in the Mapper Class. Below is an example of some measures defined in the Default Mapper Class:

1. set_run_period: An OpenStudio Measure used to define the number of timesteps per hour and specify the start and end date for running the simulation.

2. ChangeBuildingLocation: An OpenStudio Measure used to specify and load the EnergyPlus Weather file.

3. create_typical_building_from_model: An OpenStudio Model Articulation Measure that takes a custom building from user-defined geometry and assigns constructions, schedules, internal loads, HVAC, and other loads (such as exterior lights and service water heating) based on the space and sub space types.

4. urban_geometry_creation: An URBANopt GeoJSON measure used to create geometry along with spaces for a particular building, accounting for shading from surrounding buildings.

5. IncreaseInsulationRValueforExteriorWalls: An OpenStudio measure used to increase the R-Value of insulation for exterior walls by a specific value.

6. default_feature_reports: An URBANopt Scenario Measure that generates feature reports (JSON and CSV files that include feature results), which are used by the ScenarioDefaultPostProcessor.

Users can create new Mapper Classes or extend the Default Mapper Class by adding new measures to the OSW file and modifying corresponding arguments. For example, a new measure can be added to reduce lighting loads, and a reduction percentage argument can be set to 20% to define the efficiency level.

RESULTS AND ONGOING WORK

URBANopt workflows are tested and released through example projects. The workflow described in the previous section (and illustrated in Figure 7) was tested on a hypothetical project that can be found on the following GitHub repository: `https://github.com/urbanopt/urbanopt-example-geojson-project`. A GeoJSON file was created to include 13 building Features. The geometric properties and building types described in the GeoJSON file are illustrated in Figure 8 below. Users can clone this repository or use URBANopt CLI to run the example project and generate results on their local machine. Users can also refer to the URBANopt documentation pages (`https://docs.urbanopt.net/`) to learn more about the URBANopt SDK and execution process.

Ongoing work related to the URBANopt SDK to support connections with other engines and tools includes:

(1) Development of an URBANopt-REopt module that enables users to analyze and optimize distributed energy resources such as solar and behind-the-meter batteries within the URBANopt platform.

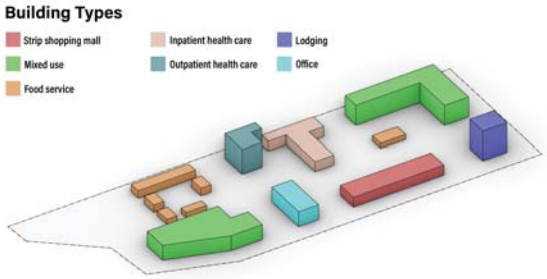

Figure 8: GeoJSON example project

(2) Development of an URBANopt-OpenDSS module enabling modeling electrical distribution networks and how they interact with buildings and distributed energy resources at a district scale.

(3) Development of district energy system modules to translate to and construct detailed Modelica-based models of district thermal energy systems (e.g., shared underground heating and cooling loops serving multiple buildings), leveraging Spawn of EnergyPlus (Wetter et al. 2015) and the Modelica Buildings Library (Wetter et al. 2014).

CONCLUSION

URBANopt is an open-source SDK. It is characterized by a modular and flexible structure that enables the integration of multiple tools, development of new workflows, and customization of existing workflows to help address a variety of environmental design tasks. The initial version of the URBANopt SDK provides basic capabilities for scenario analysis of energy-efficient districts. Because this is open-source software, software developers, academic researchers, and others can leverage components of the SDK in district-scale simulation tools and analyses.

Ongoing work includes integrating grid-interactive modeling tools such as REopt and OpenDSS, as well as developing district energy system modeling capabilities, through new URBANopt SDK modules, which will allow users to design and optimize grid-interactive efficient buildings, distributed energy resources, electric distribution systems, and district energy systems at a district scale.

ACKNOWLEDGMENT

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy. Funding provided by U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Building Technologies Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that

the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. The authors would like acknowledge and thank Daniel Macumber, formerly with NREL, who was instrumental in the developing the original vision for the URBANopt Software Development Kit. Also, the authors would like to acknowledge that additional URBANopt modules are being developed by a collaborative team that includes NREL, Lawrence Berkeley National Laboratory (LBNL), the University of Colorado Boulder, and several other partners; these additional capabilities will be the focus of follow-on publications.

REFERENCES

Allegrini, Jonas, Kristina Orehounig, Georgios Mavromatidis, Florian Ruesch, Viktor Dorer, and Ralph Evins. 2015. "A review of modelling approaches and tools for the simulation of district-scale energy systems." *Renewable and Sustainable Energy Reviews* 52:1391–1404.

Benis, Khadija, Christoph Reinhart, and Paulo Ferrão. 2017. "Development of a simulation-based decision support workflow for the implementation of Building-Integrated Agriculture (BIA) in urban contexts." *Journal of cleaner production* 147:589–602.

Bruse, Michael. 2004. "ENVI-met website." *Online: http://www. envimet. com*.

Crawley, Drury B, Jon W Hand, Michaël Kummert, and Brent T Griffith. 2008. "Contrasting the capabilities of building energy performance simulation programs." *Building and environment* 43 (4): 661–673.

Davila, Carlos Cerezo, Christoph F Reinhart, and Jamie L Bemis. 2016. "Modeling Boston: A workflow for the efficient generation and maintenance of urban building energy models from existing geospatial datasets." *Energy* 117:237–250.

Ellis, Peter. 2018. District Zero: A Decision-Making Tool for Net-Zero Communities. `https://www.districtenergy.org/HigherLogic/System/DownloadDocumentFile.ashx?DocumentFileKey=dc41ad96-8d82-089c-5b08-4a0f88333468`.

Keirstead, James, Nouri Samsatli, Nilay Shah, et al. 2010. "SynCity: an integrated tool kit for urban energy systems modelling." *Energy efficient cities: Assessment tools and benchmarking practices*, pp. 21–42.

Nouvel, R, M Zirak, H Dastageeri, V Coors, and U Eicker. 2014. "Urban energy analysis based on 3D city model for national scale applications." *IBPSA Germany conference*, Volume 8.

OpenDSS, EPRI. 2015. "Open distribution system simulator." *Sourceforge. net*.

Polly, Ben, Chuck Kutscher, Dan Macumber, Marjorie Schott, Shanti Pless, Bill Livingood, and Otto Van Geet. 2016. "From zero energy buildings to zero energy districts." *Proceedings of the 2016 American Council for an Energy Efficient Economy Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA, USA*, pp. 21–26.

Reinhart, Christoph, Timur Dogan, J Alstan Jakubiec, Tarek Rakha, and Andrew Sang. 2013. "Umi-an urban simulation environment for building energy use, daylighting and walkability." *13th Conference of IBPSA, Chambery, France*.

Reinhart, Christoph F, and Carlos Cerezo Davila. 2016. "Urban building energy modeling–A review of a nascent field." *Building and Environment* 97:196–202.

Reyna, Janet, Amir Roth, Andrew Burr, and Michael Specian. 2018. "How Can Cities Use Urban-Scale Building Energy Modeling?" Technical Report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

Robinson, Darren, Frédéric Haldi, Philippe Leroux, Diane Perez, Adil Rasheed, and Urs Wilke. 2009. "CitySim: Comprehensive micro-simulation of resource flows for sustainable urban planning." *Proceedings of the Eleventh International IBPSA Conference*. 1083–1090.

Simpkins, T, D Cutler, K Anderson, D Olis, E Elgqvist, M Callahan, and A Walker. 2014. "REopt: a platform for energy system integration and optimization." *ASME 2014 8th international conference on energy sustainability collocated with the ASME 2014 12th international conference on fuel cell science, engineering and technology*. American Society of Mechanical Engineers Digital Collection.

US-EIA. 2018. How much energy is consumed in U.S. residential and commercial buildings? `https://www.eia.gov/tools/faqs/faq.php?id=86&t=1`.

Wetter, Michael, Thierry S Nouidui, David Lorenzetti, Edward A Lee, and Amir Roth. 2015. "Prototyping the next generation energyplus simulation engine." *Accepted: 13-th IBPSA Conference. International Building Performance Simulation Association*.

Wetter, Michael, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang. 2014. "Modelica buildings library." *Journal of Building Performance Simulation* 7 (4): 253–270.