SIM AUD 2015

Washington DC USA

2015 Proceedings of the

Symposium on Simulation for Architecture and Urban Design

Edited by

Holly Samuelson, Shajay Bhooshan, and Rhys Goldstein

Capturing an Architectural Knowledge Base Utilizing Rules Engine Integration for Energy and Environmental Simulations

Holly T. Ferguson, Charles F. Vardeman II, and Aimee P. C. Buccellato

University of Notre Dame
Notre Dame, IN, USA
{hfergus2, cvardema, abuccellato}@nd.edu

ABSTRACT

The era of "Big Data" presents new challenges and opportunities to impact how the built environment is designed and constructed. Modern design tools and material databases should be more scalable, reliable, and accessible to take full advantage of the quantity of available building data. New approaches providing well-structured information can lead to robust decision support for architectural simulations earlier in the design process; rule-based decision engines and knowledge bases are the link between current data and useful decision frameworks. Integrating distributed API-based systems means that material data silos existing in modern tools can become enriched and extensible for future use with additional data from building documents, other databases, and the minds of design professionals. The PyKE rules engine extension to the Green Scale (GS) Tool improves material searches, creates the opportunity for incorporating additional rules via a REST interface, and enables integration with the Semantic Web via Linked Data principles.

Author Keywords

Expert Systems; Sustainable Data; Linked Data; Big Data; Semantic Web; Ontological Knowledge Engine; PyKE; OWL; SWIRL; REST; SPIN; SPARAQL; RIF; Green Scale Tool; Knowledge Based Rules; Machine Learning.

ACM Classification Keywords

Algorithms, Design, Experimentation, HCI, Performance, Reliability, Standardization, Verification.

INTRODUCTION

Between the years of 2013 and 2025 U.S. electricity consumption from buildings is expected to increase from 72% to 75% [20], part of the 40% of domestic primary energy usage for which buildings account [12]. In addition, domestic carbon dioxide emissions that result from building services make up 40% of the total [8]. Annually, 26% of all non-industrial waste derives from construction, demolition, and renovations of our buildings [5]; this is a drastic figure representing hundreds of thousands of building projects [6].

As a result of these trends, strategies for achieving building sustainability are needed from the beginning of the architectural design process [4] because our design choices have long-lasting impacts on the environment. This research is aimed at bringing real solutions to the professional community that can be used readily and will have positive impacts in the early stages of production. However, full environmental analysis currently requires specialized expertise and software to quantify impacts, and even those programs that are available are limited in scope and accuracy. These shortcomings are due to a restricted understanding of computational models, but more so because of incomplete material property data used in the calculation processes, if available at all. Consequently, this limited data means that decisions are made by humans considering a smaller scope of parameters and criteria than could be possible by utilizing Big Data practices and modern computing power applied to that data.

This developing concern for the future of the built environment means that tools will absolutely need to be robust enough for multi-metric energy analysis to remain advantageous in the architecture and engineering fields. Single-metric analysis programs that are widely used today will not be sufficient in the near future as the quantity of available data increases to unparalleled amounts. Unfortunately, many of the commonly used AECO models and tools (Revit®[1], Ecotect®[2], Athena Impact Estimator®[3], Green Building Studio®[4], U. S. Department of Energy (DOE-2)[5], and the U. S. Department of Commerce (BEES 4.0)[6]) are limited to manual entry of building data and/or material properties values. Thus the data deficit which modern energy applications are subjected to requires attention and strategies to not only collect this data but to do so in a manner that is adaptive and scalable enough to keep pace with rapid advances in the field.

[1] Revit ®: www.autodesk.com/products/
[2] Ecotect ®: usa.autodesk.com/ecotect-analysis
[3] Athena ®: www.athenasmi.org/our-software-data
[4] GBS ®: www.autodesk.com/products
[5] DOE-2: http://energy.gov
[6] BEES: www.nist.gov/el/economics/BEESSoftware

SimAUD 2015, April 12 - 15, 2015, Alexandria, VA, USA

Most of the simulation tools in use today will usually only perform data analysis on a single design metric, but in the Big Data era, tools will have to be able to run efficiently enough to handle several metrics at once over a large quantity of design choices. To reach an optimal level of usefulness, modern multi-metric comparison tools will not only have to perform well making their calculations, but will now have to do so when tens of thousands of material options are available from an interdisciplinary set of data related to all aspects architectural material manufacturing and usage.

Green Scale (GS) Research [19], and by extension the GS Tool[7] efforts, work to discover efficient methods to aggregate and process multitudes of material data without harming the performance of a multi-metric application, explore how these methods could communicate beneficially with the larger semantic web - partially via a PyKE Rules Engine, and ultimately seek solutions that will positively impact the way that the built environment is conceived and executed. Furthermore, the GS tool is an excellent example of a multi-metric energy analysis tool that can be used to implement experimental methods of rules communication needed for handling the quantities of data used in proper multi-criteria decision analysis while gaining access to the larger semantic web utilizing linked open data principles.[8] The following work analyzes the GS Tool with and without the implementation of a PyKE based Rules Engine (for architectural material choices, geometry discrepancies, and best practice "rules of thumb"). The structure of the paper is as follows: motivations for creating a Rules Engine Layer, the architecture of the GS Tool with the rules layer implementation, methodology of the project and creation of the rule types, results of the rules layer implementation, implications for future computer science, future research, related work, and concluding thoughts.

RELATED WORK

Projects have worked towards similar visions of knowledge-enhanced applications. SAFEgress [7] is a similar type of rule-based approach for egress simulations instead of material data leading to decision support. Design Elevator [16] was an approach to integrating knowledge into an application for influencing early stages of the design process; however, new technologies make this process more adaptable and extensible. Solibri Model Viewer[9] is a tool developed to address a vision of easier data sharing and model viewing possibilities with BIM and IFC data types. These are widely used and relevant data types as well but our research works to become reliant on a data methodology that supports all building data, regardless of source and structure.

Other relevant work is more specifically in the area of building material name matching related to energy analysis and includes projects where an IFCXML is the chosen file format for handling data to decrease erroneous input [14] or to create software implementations for shareable data between several design fields related to building systems [22]. While currently there are several fragmented computer science areas of research related to the architecture and engineering professions including other knowledge sharing system for sustainability sciences [15], the goal is to have a complete and interactive decision system for architecture applications similar to the UFIT ontology-driven knowledge-based system for enhancing physical fitness [18]. At that point, architecture-related ontology patterns can be applied to the data as well, such as those that represent building material transformations [21] for more accurate lifecycle inventory analysis [1] and perhaps advances can be made to the GS database with known methods of modeling semantic data [2] [3].

MOTIVATION TO IMPLEMENT A RULES ENGINE LAYER

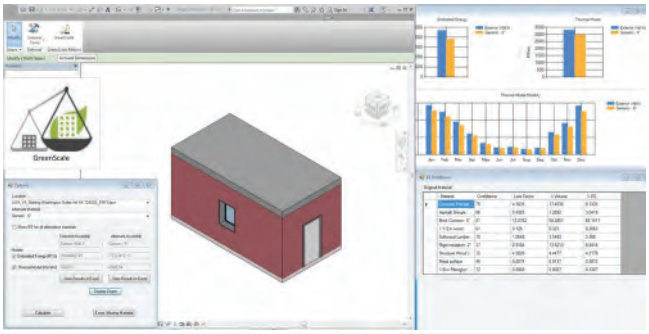

Figure 1: Green Scale Interface from the Revit[1] Plug-In

The Green Scale Tool (see Figure 1) was built to be integrated with the workflow of an architect and provide a broader perspective of the environmental impacts caused by the built environment through a multi-metric comparative analysis including BEAM [23] thermal heat flux model and Embodied Energy life cycle inventory model [4]. Development of the tool has been done in close consultation with architect practitioners and architecture students to integrate rules into a computational design environment that can be utilized on a daily basis. The tool heavily utilizes the existing Revit interface, and thus is largely tied to whatever usability advantages and disadvantages Revit provides. Further iterations of the research would include testing with a wider community outside the original development collaboration team.

In this paper, the GS tool is used as the base framework for applying a PyKE[10] based rules engine enabling Decision Theory [13] models to be incorporated such that architects will make better material choices for the built environment. Within certain existing simulation tool databases, there is

[7] Green Scale: www.greenscale.org
[8] Linked Data: www.w3.org/standards/semanticweb/data
[9] Solibri: http://www.solibri.com.au/products/

[10] PyKE ®: pyke.sourceforge.net

syntactical and semantic ambiguity that is not adequately addressed by application of regular expressions or even generating the correct nominal values for material data. By extension, adjustments in energy calculations are often overlooked due to the misinterpretations because of the structural nature of Open Green Building XML (gbXML)[11] based schemas as implemented in common architectural design tools and the limited explicitness to interpret XML tag structured data. A PyKE rules engine, however, means rules can be adaptively constructed to fix discrepancies and be extrapolated for use with other rule engines and applications.

Past and ongoing research related to rule-based expert systems include the development of the Python based rules engine presented at PyCon in 2008 [9], which was the foundation of this project. PyKE is a knowledge-based inference engine inspired by Prolog[12] but implemented completely in Python. This method of logic programming allows the free exchange of Python expressions and customized expert system rules while improving the availability and quantity of code reuse. A distributed API-based system can be added after the rules are implemented as traditional python functions via PyKE. Stored in a designated location away from pre-existing code, the knowledge base is now portable, easily editable, and accessible for adding new rules over time.

The proposed rules layer resolves problematic or often missing XML tags and, combined with regular expressions, these XML files can now be efficiently and more accurately parsed within the architectural simulation tool (with minimal interventions aside from deliberately adding new rules to the knowledge bases). Additionally, rules capturing "rules of thumb" tacit knowledge known to the architecture practitioner can be triggered based on conditions detected in XML data structures. This PyKE implementation builds on the Green Scale Tool using a flexible framework for mapping regular expressions to conditions and finally to an associated action for several types of architectural decisions (material choice, geometry discrepancies, and architectural best-practice rules), all of which are encountered through using this example application. This framework constructs a mapping between the architectural approximation rules and the GS model data, which usually has computational processes and code sequences that require more analysis than the standard simulation tools allow for accuracy in the calculations. These methods often include strings, regular expression matching, and other numerical data.

Varying architectural perspectives and experience can be leveraged in order to gather a broader set of rules and expand the existing collection. The resulting knowledge base can then be used to benefit the larger community and be applied as Linked Data via SPIN[13], SPARQL[14], and the PyKE rules engine. Using REST[15] type interfacing and web services, the accessibility on wireless devices can be increased and the system can be effective in cloud-computing environments. This means the final implementation would be able to interface with other applications thus more usefully connecting all the system elements. The rules layer and ultimately the knowledge base communicate with the PyKE rules engine; once it is moved to the Cloud, it can be extended and improved by Machine Learning [11]. Additionally, SPIN rules can be integrated with SPARQL for generating inference based connections with relevant methods in the expanding knowledge base. Additionally, SPIN could provide mechanisms for data consistency and integrity checks within the knowledge base.

TOOL ARCHITECTURE AND RULES ENGINE

The Green Scale Tool Architecture

The GS Tool is implemented as a Revit 2014 Plug-In and has an architectural design presented in Figure 1. After the Plug-In launches, the user is then required to choose surfaces from the architectural model of at least one construction type. Next, the GS user interface is presented from which the Python energy models are called. This is a windows application that handles locating all the construction types for a particular surface, generating a gbXML for each type found, presenting a list of these possibilities to the user, and processing the Python models once for each choice selected so as to create an iterative and extensible simulation tool. This processing loop can be repeated as desired by the user. From the Windows Application layer, the Python based analysis models are launched individually enabling potential parallelism and extensibility and currently include modules for computation of life cycle inventory based Embodied Energy and Thermal Heat Flux metrics for a given architectural design [19]. Individual modules also require connections to additional data sets and external entities; for example, the thermal model would need access to Energy Plus climatology data. A SQLite3[16] database called the "GS Material Property Database" is queried from within the running Python models via a Django[17] based RESTful web service running on a remote Linux server.

Once each one of the Python modules has completed execution for all assembly versions selected for comparative analysis, the collected calculation results are transferred back to the Revit Plugin via standard output. These results are reported and visualized using text tables,

[11] gbXML: http://gbxml.org
[12] Prolog: http://www.deransart.fr//prolog/docs.html
[13] SPIN: http://spinrdf.org
[14] SPARQL: http://www.w3.org/2009/sparql/wiki
[15] REST: www.ibm.com/developerworks/library/ws-restful
[16] SQLite3: http://sqlite.org
[17] Django: https://www.djangoproject.com

charts, graphs, PDFs, etc. As a result, the application is an ideal modular framework for implementing computational rules because it already facilitates multi-metric calculations and comparisons to be analyzed simultaneously. As modules are added and the database expands, the application will be able to connect to other databases that conform to a RESTful API, collecting new material data on-the-fly as needed by the models. The data aggregation can even extend to cloud-based or crowd-sourced datasets as the community expands. The inherent nature of all of these goals will benefit long-term from rule-based processing of properly structured data (see Figure 2).

Addition of a Rules Engine and API Framework

We have implemented an extended architecture that allows the execution of a rule set and knowledge base to be used with design evaluation models, potentially in distributed computing environments as design complexity increases. The focus of this work is the addition of the PyKE based Rules Engine for the GS Tool (see Figure 2). Figure 2 demonstrates the existing GS Tool, the new additions made as a result of the implementation of the PyKE rules engine, and the elements to be added through future research. The rules are incorporated as an extension of the python modules lying within the previously described main application architecture. This works in conjunction with a set of .kfb, .krb, and .kqb extension files that represent a fact base, forward and backward chaining rules and a "question base" respectively, that have Prolog inspired syntax. Because PyKE is a hybrid of python and a formal logic language that provides a knowledge-based inference engine, it allows communication from any existing module for easy intermingling between Python expressions and the expert system rules [9]. This also means that the decoupling of rules from the main body of source code is now a functional possibility.

By integrating a rules engine directly into a multi-metric application architecture, we have created a bridge that is extensible in terms of encapsulating and transferring rules and knowledge bases between GS platform instances. (see Figure 2). Mechanisms for crowd-sourcing can be constructed within the GS tool user interface allowing data and tacit knowledge to be gathered from a variety of sources not only to build up the data in the GS Material Property Database itself, but to add rules to the PyKE base files for the rules engine to utilize. We envision the GS tool will be able to use additional logic and machine learning to automatically incorporate these rules and infer decisions using semantic web based technologies creating a truly distributed computational architecture. At this point, a SPARQL endpoint can be added that to facilitate distributed queries and inferencing from a web of Linked Data, increasing the variety of data from architects, manufacturers, construction contractors, and governmental organizations to be incorporated in a rules based decision support system. The PyKE rules engine can potentially be connected to formal Ontologies and Ontology Patterns implemented in the Web Ontology Language (OWL) that allow larger expressivity based on first order description logics creating an architectural design knowledge base for isolated calculations such as robust and highly accurate multi-criteria decision analysis.

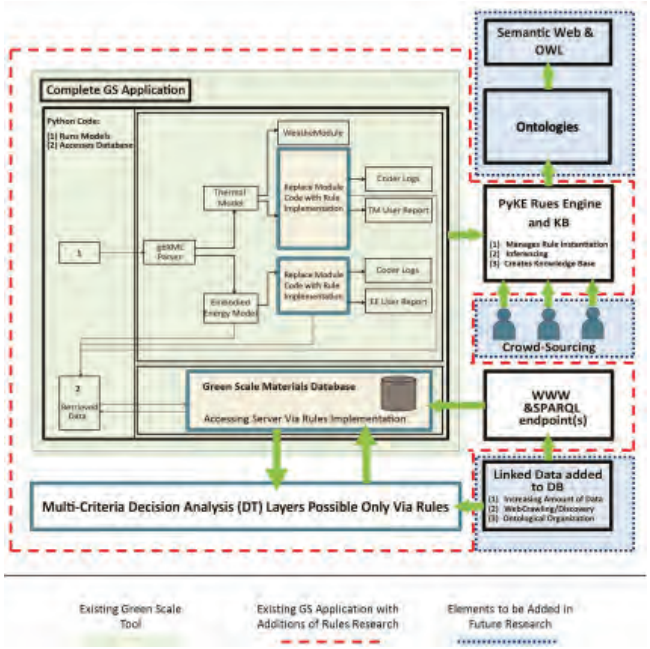

Figure 2. Existing and Future Application Architecture

METHODOLOGY

Application Restructuring for a Rules Engine

Current tools are using a very limited amount of data to make calculations or are reliant upon inaccurate user input for results. This means that all of the source code (often if-else or switch statements) is able to handle potential data combinations; in the Big Data era, continuously updating source code to add new cases or storing all potential logic in the source code files is impractical for performance in the very least.

The rules layer created for the GS activates only once within the GS tools main routine and from this point the individual rules are called as needed based upon the type of rule needed and what associated pieces of the knowledge base are required. This process introduces some initial overhead, but the benefits of having decoupled rule sets far outweigh the change in processing times (For more see: Rules Aiding Decision Theory Frameworks). Throughout the source code, entire sections of nested if-else logics were replaced with only one or two rules to achieve the same model outputs for all case studies.

As the architectural design documents increase in complexity or number of surfaces, it is likely more inconsistency in XML data will be encountered due to data quality [10] or inconsistent naming of materials,

assemblies, dimensions and novel design geometries that have not been accounted for in analysis modules. We have observed that gbXML files alone can have several types of errors that need the use of rules to be resolved so that the python models calculate accurately. Incorrect tags, values, strings, and/or missing versions of the same types of data are some of the most problematic XML problems encountered from the Plug-In export from Revit). For example, data may be arbitrarily concatenated into strings that are then output from Revit and available to use in certain cases instead of a structured set of material property data and values using an appropriate schema. Cases such as these result in a complex set of conditional statements that are difficult to manage in a purely procedural language implementation; this is why gbXML alignment is necessary. The rules engine is a more flexible and singular solution for these data structures.

Three Types of Architectural Rules Considered

After exploring a large set of architectural models that are typical of a standard architectural workflow and studying the tendencies of the Revit outputs, it was observed that most of the code that would benefit from being executed utilizing a rule language could be divided into three distinct rule categories: material choice, geometry discrepancies, and architectural best-practice rules-"rules-of-thumb." Although some rules need to be used in combination, given a particular analysis model calculation type, the concepts are maintained throughout the source code and the knowledge base files. The belief is that as the rule base grows, the more comprehensive the rule answers will be since they will derive from a greater body of knowledge.

While outside of the scope of this work, the addition of new rules over time could potentially be achieved by automated processes where information is extracted from design documents and analysis during their creation. Rules can be created and extended due to the modularity of the GS Tool, including ones to capture historic, regional, resilience, and user preference data. Because the rules base is encoded in plain text files, rules can be readily exchanged. These rules could be aggregated through reliable methods such as properly sourced databases, experts and other standards (i.e. written by a domain expert via the correspondence of architects and their knowledge) to give assurance of rule quality. Lastly, application of natural language based rule entry using the existing GS tool interfaces, which could then use semantic technologies such as SPIN [17] to map to formal rules. In the following section, conceptual and data levels are outlined as the method for bridging the gap of various data description types. Natural Language Processing[18] (NLP) and SPIN would be able to take user input and translate entries into variables and then rules for automatic code generation as illustrated in Figure 3.

[18] NLP: https://www.coursera.org/course/nlp

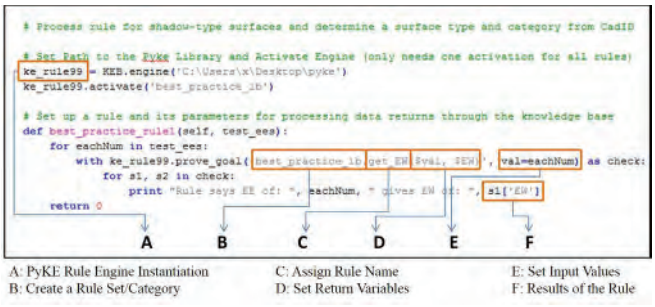

Figure 3: Example Rule Implementation

Material Choice Rules

As the name indicates, this category of rules is used when there is a question about what type of physical material identity is most appropriate for a calculation or what type of surface to assign within the python analysis models to produce the most accurate end results. Some of the rules that appear in this category for the GS application implementation include a sequence that determines a default door or window type with associated material data (density, thickness, etc.). There is also a rule set that determines a surface type and construction ID when only a descriptive string is provided through the gbXML serialization. A similar rule initializes missing data when columns are present in a space by determining between round and square shapes as well as cross-section areas and construction descriptors.

Geometry Discrepancy Rules

Apart from material choices, there is a large category of inconsistencies that are now resolved by rules where the geometries of a surface, material, or other object are assigned values for material properties from a Revit materials database through the gbXML transfer mechanism, but those values are incorrect, or more general that would be desired for a given analysis type. In certain common situations, the geometries are more easily exported by estimating instead of handling all possible geometric cases. This makes the Revit® export process easier and faster at times, but creates issues that rules can now appropriately resolve for the external energy simulation calculations. These rules include volumetric adjustments when certain strings are present including walls that have an alternating stud and insulation type of assembly, adjustments to insulation volumes and coefficients depending on insulation types, and certain structural elements.

Architectural Best-Practice Rules

Architectural best practice rules are representing a type of tacit knowledge that most architects and engineers have accumulated through education, experience and specialization best practices. We can now provide a possible mechanism by which some of this tacit knowledge can be captured and used analytical or decision support models. These rules can include factors for reducing the embodied energy to get embodied water, handling the calculation of the thermal conductance of a material (C-

value), and swapping back and forth between temperature units and conversions for the thermal model. Tacit knowledge "reported" back by constructed buildings via embedded sensor networks is another data source and knowledge base scenario that can contribute to other rules.

Combinations of Rules

Interestingly, as seen in the previous section, there can be situations where rules seem to fit in more than one rule category. For these cases, there can be a best fit determination based upon how the rule is being used, or eventually there could be a pattern layer on top of the rule sets that flag them automatically into certain categories based upon what a user enters into either Revit and/or a Rules Entry User Interface. For example, the determination of volume reduction on a stud wall assembly can use the best practice rule of 15% structure and 85% insulation when the gbXML indicates both materials as taking up 100%. This would be one rule type that requires a material decision rule about the studs and insulation, followed by an adjustment in the volume of both of these materials. This second rule can be of a geometry or a best practice type, but since it is changing the volume for calculations, it is grouped with geometry discrepancy rules, for this situation.

The Use of a Rules Engine at a Larger Scope

A PyKE layer allows for further extensions to the existing analysis modules and beyond the application layer of the tool. PyKE specifically was chosen for this type of application because the source code is already implemented in the Python programming language. Using a rules engine that is coded in the same language makes this addition more interoperable and cohesive with the existing multi-metric application. There are also less problems communicating with the GS Material Property Database than there might be if another language were introduced, and this creates the effect of a less problematic bridge between the application and incorporating LD, DT, and/or the Semantic Web.

Without a separate knowledge base and rules, all of these nested, ever-growing statements would be completely stored in the source code, creating an overly complex logic that makes the python models slower and/or too complicated to truly capture architectural knowledge from the limited data available in the gbXML files. It would also keep the code in an impossible state for end-users to be able to understand where to enter their own additional conditions for materials, geometries, and best practices. If this logic were to remain in a purely procedural form, it would also make the models eventually very inefficient and prone to possible errors as multiple people want to add more conditions to the models. Therefore, the rules engine employed now provides a better solution to capturing this logic, processing for semantic decisions, and application extensibility. It also allows for the possibility of self-consistency checking and monitoring data quality.

RESULTS OF A RULES ENGINE WITH THE GS TOOL

One of the goals of this rules layer implementation was to understand the implications of a more expressive rules language to the overall run time of the program. Our team felt that there may be some advantages for several reasons: 1) the application will no longer need to filter though long lists of conditional string comparisons, 2) the creation of a simpler process of interaction as new rules are added means general users can add rules, and 3) potential interoperability with the semantic web exists as an extensible application feature since rules are now all stored in a single location and format. For the set of seven building models tested (see Figure 4), the full implementation of rules in the current version of the GS Tool was the first experiment conducted and shows a slight increase in the processing time (see Figure 5). This indicates that the simplicity of the architectural models and consistent materials used therein are not yet overcoming the overhead of activating the rules engine and processing the rules in this manner. This is not a deterrent to a rules based approach, however, because currently the rules are based upon a subset of the potential possibilities a fully developed database and surface set would provide (See the following section).

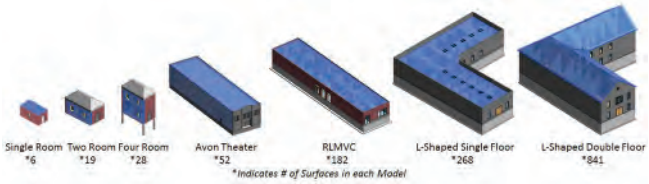

Figure 4: Test Set of Architectural Models

To study processing time relative to building model complexity, the aforementioned seven architectural models (see Figure 4) were run through each of the two python analysis models (embodied energy and thermal heat flux) both before and after the rules were implemented into the code (see Figure 5). The embodied energy model shows a modest increase in run time for the simpler architectural models (0.102 seconds overall for the single room model of 6 surfaces on ground level) and a slightly higher increase in run time for the larger ones (4.59 seconds overall for the largest tested model of 841 surfaces spanning two stories). This is due in part to certain types of rules (see Figure 6) requiring more execution time to run than others and due to the fact that the GS Material Property Database is sparse resulting in shorter processing time compared to having to search tens of thousands of entries at full capacity with fuller collections of data. The increase in raw run time for the thermal model is almost negligible 0.45 seconds for the single room model and 25.82 seconds for the largest tested model-which is still only a 5.91% increase over the original, and will become mode negligible as architectural model size increases and varies in complexity. For the thermal model, the percent of time increase is 6.49% for the single room model and only 5.91% for the largest model, suggesting that the increase in overhead time will be

overcome as the models and database become more complex and the architectural models become larger in scale. It should be noted that while it can be true that run time correlates to number of surfaces to process, often numbers of surfaces are not directly proportional to run times due to the variation in surface types and assembly components throughout a building.

Figure 5. Comparison of Traditional Models with Rules

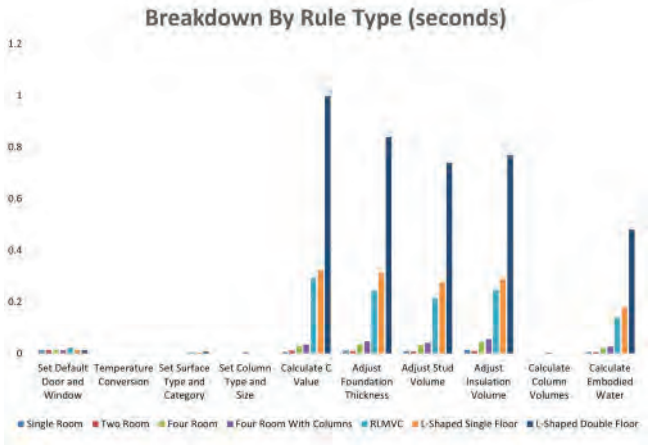

Figure 6. Breakdown between Types of Rules

Additionally, with the rules set into the different categories described above, it is possible to compare and see what types of rules are causing the most benefit and the most time increase in the overall program (see Figure 6). One possibility is that dynamic rule sets could be chosen given user criteria for accuracy vs architectural model complexity allowing rules that offer the most benefit for a given function to be instantiated.

BIGGER SCOPE BEYOND THE RULES ENGINE

Future Work and Additions to the Rules Layer

After the development of a working Rules Engine Layer, the next steps to improve energy simulation tools would involve applying an extension to data aggregation in the GS Material Property Database. This can be better achieved via crowd-sourcing and/or automated web-crawling since the amount of data does affect the decision making quality. To further these future improvements, there are extensions that would help to materialize these pursuits faster and easier. For example, a secondary interface used for the purposes of gathering database data and rules would help aggregations run more smoothly and reach a more diverse group of users. Thus, the simulation tool could accumulate a more diverse range of rules and knowledge via crowdsourcing, individual, and automated use.

Rules Aiding Decision Theory Frameworks

We believe as the GS Material Property Database grows, the potential efficiency will improve relative to a procedurally structured code and by creating de-coupled rules sets for more maintainable source code, improving interoperability, facilitating data collection via crowd-sourcing, maintaining further compatibility with Python, and setting up rule-type organization and optimization. Methods such as crowd-sourcing provide the basis for DT to become operational. Ideally, for proper multi-criteria decision analysis, all of these interconnected extensions of the application would aid in generating sets of decisions or choices and the most accurate application output.

The rules create a degree of flexibility for certain methods of decision analysis that if-else/switch statements simply are not capable of generating. For example, a knowledge base that is established at the time the program is instantiated provides data in one place such that rules can simultaneously select parameters based on a combination of minimizing cost, shipping, and energy consumption. In reference to the modest increase in run times mentioned in the previous section, it is obvious by this point that optimizing decision suggestions over such an extensive collection of data is a real concern. Since the change in run times of the tested models showed that there could be a decrease in time as the code and model complexity increased, a second experiment was conducted with a basic set of conditional statements to see at what point the rule structures could overcome the traditional conditional lines of code. For a set of 10 searches using a basic naming system (i.e. one string example is: ‘Brick, Common: 10 1/2" ’), it took between 6,400 and 12,800 conditional statements for the rules to perform better. At first glance, this may seem like a rare circumstance, but it should be noted that this is not only for the most simplistic structure for a basic string search in each of the 10 cases, but it also processed with a minimal version of our material data strings that would become quite complex in time as product manufacturer data is incorporated. In reality, the necessary conditional structures will be far more complex in terms of the number of lines of source code and therefore without a more sophisticated processing schema, the complexity will only increase with growing communication abilities across the semantic web. Additionally, 12,800 comparisons is not that unrealistic to determine rules to be a good option

because in a real database scenario, there could easily be tens or hundreds of thousands of options for each rule, making the 12,800 baseline well-worth the effort of PyKE rule implementation (as well as the other discussed benefits of rules). Rules structures also give a larger degree of expressivity through formal logics, do not have the complications of the old procedural approach, and have computational advantages.

CONCLUSION

Regardless of data optimization preferences, building sustainability measures will become increasingly accurate as more data becomes available from which to make design choices. Architectural tools, which are currently relying on limited or local data collections, are suffering in the advent of Big Data possibilities. The larger the database grows and the more that Linked-Data principles are integrated into the data sets, the more efficient and accurate architectural PyKE rules will perform thus improving our multi-metric energy simulation tools. This is the future of energy simulation in a global world functioning within the semantic web environment and it will enable multi-criteria decision analysis in an unprecedented fashion.

ACKNOWLEDGMENTS

The GS Research Project has been made possible by Center for Research Computing at the University of Notre Dame, the University of Notre Dame Office of the VP of Research Faculty Research Support Program 2012 Regular Grant, and the University of Notre Dame Center for Sustainable Energy Sustainable Energy Initiative Program and undergraduate research support by the Slatt Foundation.

REFERENCES

1. Bertin, B. Scuturici, M., Pinon, J. M., Risler, E. Semantic modelling of dependency relations between Life Cycle Analysis processes. 2012 pp. 109–124.
2. Bertin, B., et al. A Semantic Approach to Life Cycle Assessment Applied on Energy Environmental Impact Data Management. ACM:EDBT. 2012.
3. Bertin, B., et al. Carbon DB: A semantic life cycle inventory database. 2012 pp. 2683–2685.
4. Buccellato, A.P.C., Duke, R.P., Veenstra, K. E. Material Matters. Smart and Sustainable Built Environments (SASBE) 2012. Sao Paolo, Brazil.
5. Buildings and their Impact on the Environment: A Statistical Summary. Technical report, April 2009.
6. C-Series Reports. Technical report, January 1995.
7. Chu, M.L. et al. SAFEgress: A Flexible Platform to Study the Effect of Human and Social Behaviors on Egress Performance. ACM: SimAUD. 2014.
8. Emissions of Greenhouse Gases in the US 2007. Technical Report DOE/ EIA-0573(2007). 2008.
9. Frederiksen, B. Applying Expert System Technology to Code Reuse with Pyke. PyCon: Chicago. 2008.
10. Furber, C., Hepp, M. Using SPARQL and SPIN for Data Quality Management on the Semantic Web. SpringerLink: BIS. Volume 47, 2010, pp 35-46.
11. Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics, 2nd Ed. 2009.
12. Independent Statistics and Analysis: U.S. Energy Information Administration. (2014) Available from: http://www.eia.gov/tools/faqs/faq.cfm?id=86&t=1.
13. Ishizaka, A., Nemery, P. Multi-criteria Decision Analysis: Methods and Software. John Wiley & Sons, Ltd. August, 2013.
14. Kim, K., et al. Semantic Material Name Matching System for Building Energy Analysis. Elsevier: Automation in Construction. 2012 pp. 242-255.
15. Kraines, S., Guo, W. A System for Ontology-Based Sharing of Expert Knowledge in Sustainability Science. Data Science Journal: Vol 9. 2011 pp. 107-123.
16. Oh, Y. et al. Intelligent Critiquing of Design Sketches. CDG, Carnegie Mellon University. AAAIFS. 2004.
17. Sander, M., et al. Ontology-Based Translation of Natural Language Queries to SPARQL. NLABD: AAAI Fall Symposium. 2014.
18. Su, C. J., et al. Ontological Knowledge Engine and Health Screening Data Enabled Ubiquitous Personalized Physical Fitness (UFIT). Sensors. 2014.
19. The Green Scale, Green Scale Research Project. (2015) Available from: http://www.greenscale.org.
20. U. S. Department of Energy and Annual Energy Review 2007. Technical Report DOE/ EIA-0384 (2007). 2008.
21. Vardeman, C. et al. An Ontology Design Pattern for Material Transformation. WOP2014 Patterns. 2014.
22. Yang, Q.Z., Zhang, Y. Semantic Interoperability in Building Design: Methods and Tools. Elsevier: Computer-Aided Design 38. 2006 pp. 1099-1112.
23. Yu, N., et al. Thermal Model of Green Scale Digital Design and Analysis Tool for Sustainable Buildings (BEAM). Aerospace and Mechanical Engineering Internal Report: University of Notre Dame. 2014.