# LINKING INTERACTIVE OPTIMIZATION FOR URBAN PLANNING WITH A SEMANTIC 3D CITY MODEL

N. Schüler[1,*], G. Agugiaro[2], S. Cajot[1], F. Maréchal[1]

[1] Industrial Process and Energy Systems Engineering Group, École Polytechnique Fédérale de Lausanne (EPFL),
Rue de l’Industrie 17, 1950 Sion, Switzerland - (nils.schueler, sebastien.cajot, francois.marechal)@epfl.ch
[2] Smart and Resilient Cities Unit, Austrian Institute of Technology (AIT),
Giefinggasse 4, 1210 Vienna, Austria - giorgio.agugiaro@ait.ac.at

**Commission IV, WG IV/10**

**KEY WORDS:** urban planning, decision support, interactive optimization, data model, CityGML, Scenario ADE, URB$^{io}$

## ABSTRACT:

The cities in which we live are constantly evolving. The active management of this evolution is referred to as urban planning. The according development process could go in many directions resulting in a large number of potential future scenarios of a city. The planning support system URB$^{io}$ adopts interactive optimization to assist urban planners in generating and exploring those various scenarios. As a computer-based system it needs to be able to efficiently handle all underlying data of this exploration process, which includes both methodology-specific and context-specific information. This article describes the work carried out to link URB$^{io}$ with a semantic city model. Therefore, two key requirements were identified and implemented: (a) the extension of the CityGML data model to cope with many scenarios by the proposition of the Scenario Application Domain Extension (ADE) and (b) the definition of a data model for interactive optimization. Classes and features of the developed data models are motivated, depicted and explained. Their usability is demonstrated by walking through a typical workflow of URB$^{io}$ and laying out the induced data flows. The article is concluded with stating further potential applications of both the Scenario ADE and the data model for interactive optimization.

## 1. INTRODUCTION

Urban planning deals with the development of new and existing parts of a city. Due to its intrinsic multidisciplinary approach, urban planning affects a multitude of domains and spans various scales ranging from the single building or even their floors up to a district or the whole city. The outcomes of an urban planning project generally impact various stakeholders. Consequently, many far reaching decisions have to be taken during early phases of urban planning. The resulting large decision space implies that, in theory, many possible configurations of the future city should be explored. However, due to the sheer quantity of these configurations together with limited financial and time resources, often only a few configurations - or scenarios - are explored in current planning practice (Balling et al., 1999). Computer-based approaches can provide useful assistance by exploring more thoroughly and quickly the decision space of planners and proposing optimized plans to them.

Since the decision space in urban planning is very large, it would also take a large amount of computational resources to capture it entirely. In addition, because many stakeholders are involved and many conflicting values are at play on the urban scene, accurately framing the problem and preferences before knowing the consequences of decisions is very challenging (Rittel and Webber, 1973). To overcome this, human-computer interaction and multi-objective optimization allow to use the experience and intuition of planners to guide and narrow the search towards promising alternatives, while at the same time providing a learning opportunity to refine the intuition and knowledge of planners.

The idea of employing such an interactive optimization (IO) approach (Branke, 2008) for early-stage urban planning has led to the development of the web-based planning support system URB$^{io}$ [1]. It employs multiparametric Mixed Integer Linear Programing (Gal, 1995) and an interface based on parallel coordinates to generate and compare a large number of plans. An example of such a parallel coordinates plot can be seen in the application section of this article (figure 4). The parallel coordinates are used in URB$^{io}$ not only to display simultaneously a large number of attributes but also to let the user steer the optimization interactively. Since a detailed description of URB$^{io}$ is beyond the scope of this paper the reader is referred to the following articles containing: (a) an explanation of both the general workflow and this steering process, (Cajot et al., 2017), (b) a description of the underlying optimization method and models (Schüler et al., 2018). The focus here is to highlight the requirements for a corresponding data model. These requirements can in principle be assigned to either the methodology of interactive optimization or the context of urban planning.

The methodology of interactive optimization requires a data model that allows to depict the changes made by a user which can be transformed into input information to the algorithms. The calculated results have to be stored again in order to be visualized in the interface.

The context of urban planning requires a sufficiently detailed urban model. Such a model needs to contain all representative entities implied in a planning process together with their spatial and non-spatial characteristics and their mutual relations and dependencies. However, collecting, harmonising and integrating huge quantities of urban data for such a model can be a tedious and resources-intensive task. This has led to the establishment

*Corresponding author

[1] `www.urbio.ch`

of open, and extensively documented standards like CityGML[2] (Gröger et al., 2012). Thanks to these standards, the availability of semantic 3D city models has increased in the last years (CityGML, 2018). This again is a good argument for adopting the CityGML standard, as it facilitates the application of URB$^{io}$ to different planning projects and cities. A further advantage of CityGML is the existence of a number of tools which cover most of related needs in terms of ETL (Extract, Transform, Load) operations, database implementations for data management like the open-source 3D City Database[3] (Kolbe et al., 2016), web-based solution for data access (Web Feature Services), and visualisation solutions like Google Earth or Cesium WebGlobe (Yao et al., 2018). In particular, the possibility to access and explore urban data by means of maps allows to better understand the results, and consequently to define further tasks and analyses. An extensive search of the decision space, which is a key strength of the interactive optimization approach, implies that it is required not only to store one instance of the city but a large number of alternative scenarios, which need to be managed. However, with the current data model infrastructure this is not easily possible.

Consequently, this paper focuses on the work carried out in order to link the interactive optimization framework for urban planning URB$^{io}$ with the richness of a semantic 3D city model based on CityGML. In order to achieve this, the two following contributions are proposed as a response to the identified gaps:

a) the extension of CityGML in order to cope with scenarios (section 2.2)

b) the definition of a dedicated data model for interactive optimization (section 2.3)

## 2. METHODOLOGY

### 2.1 City models and scenarios

A semantic 3D city model represents a sort of "digital twin" of a real city. When working with city models, a common functionality that many applications eventually tend to develop is the possibility to define (a) different urban configurations and scenarios, and (b) compute some results and Key Performance Indicators (KPI) to assess qualitatively and quantitatively the performance or the effects of selected measures carried out either on the whole city, or a district, or a group of buildings, or even a single object. Therefore, a city model should not only represent the current situation of a city, but could also include different scenario-derived urban configurations. Each urban configuration is obtained by means of a number of changes and modifications. As a matter of fact, there are only three basic operations that can be combined to derive a new urban configuration: (a) the addition of a new city object (e.g. a new building is built), (b) the removal of a city object (e.g. a building is demolished) or (c) the change of one characteristic of a city object (e.g. the number of inhabitants in a building). If different software applications are linked to one or more urban configurations, managing the respective set of results may become quite complex. At the moment, however, CityGML does not provide a standardized way of dealing with multiple urban configurations, scenarios, scenario-dependent results, or the description and documentation of the changes required to obtain an urban configuration B from an urban configuration A. However, it is possible to extend CityGML by means of so-called Application Domain Extensions (ADE) (van den Brink et al., 2013). An ADE allows to define and add additional entities and properties to the current ones. Alternatively new feature classes or city objects can be specified by derivation from the general GML class _Feature, the _CityObject base class, or any specific CityGML class.

Initial research work on extending CityGML to support multiple versions of a city model has been carried out by (Chaturvedi et al., 2017). Their focus is on adding support for the management of versions and history within semantic 3D city models in the next version of CityGML 3.0. However, the proposed approach is not meant to be backported to the current version 2.0 of CityGML, as it changes rather deeply the overall data model. The work presented in this paper focuses instead on adding support to the current version of CityGML, and it has partially taken inspiration from the work of (Sindram and Kolbe, 2014), where a systematic approach for modelling urban planning actions by means of complex transactions on semantic 3D city models is proposed. In the context of the development of the Energy ADE, (Benner, 2017) and (Agugiaro et al., 2018) have proposed how to store energy simulation inputs and results, but so far - according to the authors' knowledge - no generic solution has been proposed tackling the above-mentioned challenges. In order to overcome these shortcomings, further research work has been carried out to define a so-called Scenario ADE.

### 2.2 Scenario ADE

The data model of the Scenario ADE is briefly presented here. Rather than explaining it in every detail, the focus is put on the aspects of the ADE which are relevant for the proposed IO approach for urban planning. Further details are given in the documentation provided together with the software resources on GitHub[4]. A UML diagram (figure 1) is used as reference in order to better illustrate the Scenario ADE elements and where they connect to the existing CityGML classes.

In the current implementation, the Scenario ADE consists of two parts: the core module and the time series module, which is intended to model time-dependent variables. The latter is omitted for space reasons. The core module is built around the main class Scenario. In the Scenario ADE, a scenario is intended as a unique composition of:

- A set of physical objects, be it a whole city model, a single city object, or a group of city objects. The last are modeled by means of the CityObjectGroup class, itself derived from a _CityObject. For this purpose, the class Scenario is associated both to the CityGML CityModel and _CityObject classes. In other words, a scenario can be linked only to the objects considered for a specific planning project. Each scenario can be further characterized in temporal terms, i.e. as a time period or a fixed instant. A scenario can be itself derived from a previous scenario. This is made possible by means of the `isDerivedFrom` association. Thus, changes made within preceding scenarios can be traced back and aggregated if required;

- A number of optional scenario parameters, modelled by the class ScenarioParameter. These parameters can be used to

[2] `www.citygml.org`
[3] `www.3dcitydb.org`
[4] `https://github.com/gioagu/3dcitydb_ade`

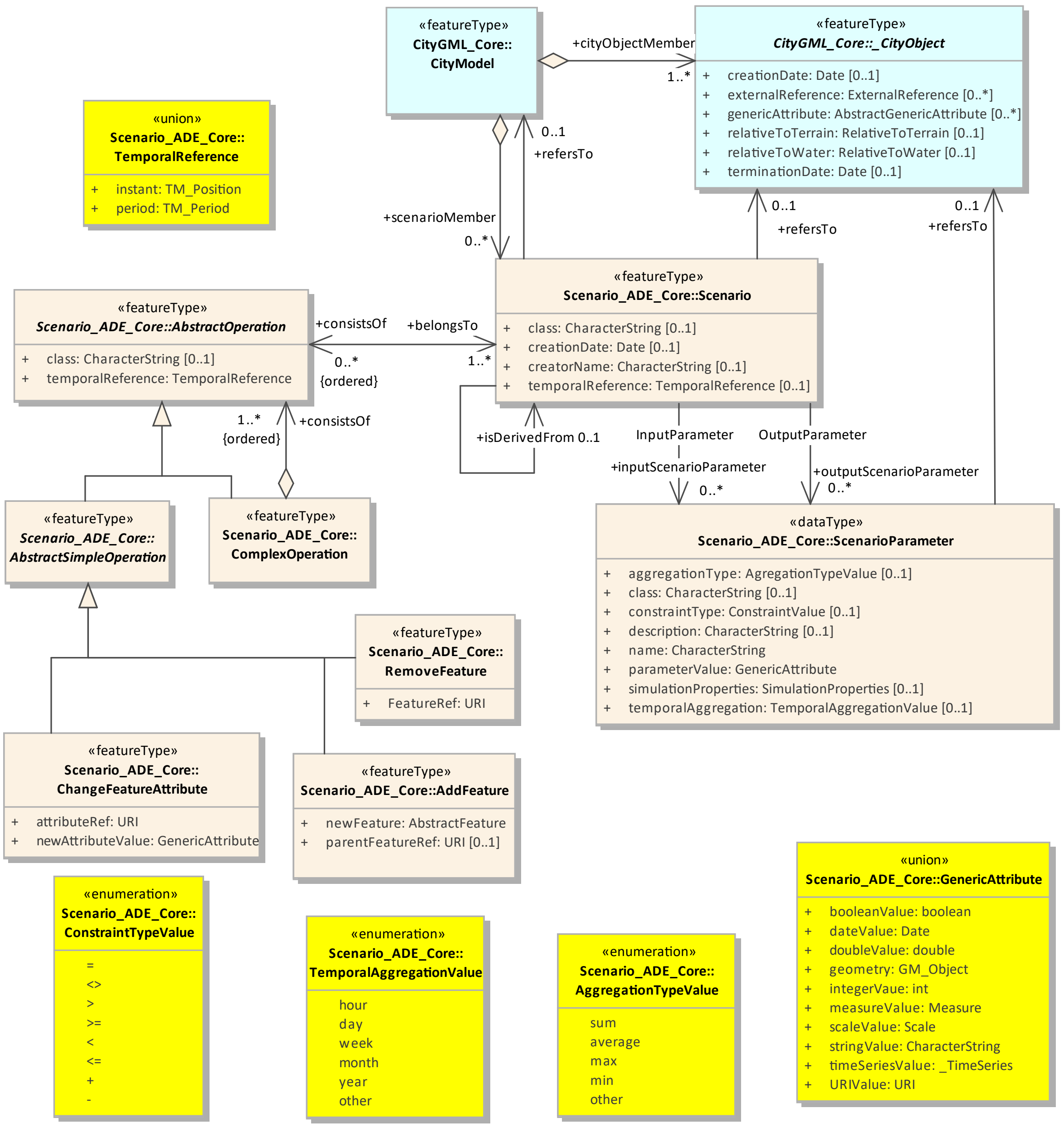
«featureType» CityGML_Core:: CityModel
+cityObjectMember
1..*
«featureType» CityGML_Core::_CityObject
+ creationDate: Date [0..1]
+ externalReference: ExternalReference [0..*]
+ genericAttribute: AbstractGenericAttribute [0..*]
+ relativeToTerrain: RelativeToTerrain [0..1]
+ relativeToWater: RelativeToWater [0..1]
+ terminationDate: Date [0..1]
«union» Scenario_ADE_Core:: TemporalReference
+ instant: TM_Position
+ period: TM_Period
0..1
+refersTo
+scenarioMember
0..*
0..1
+refersTo
0..1
+refersTo
«featureType» Scenario_ADE_Core::Scenario
+ class: CharacterString [0..1]
+ creationDate: Date [0..1]
+ creatorName: CharacterString [0..1]
+ temporalReference: TemporalReference [0..1]
«featureType» Scenario_ADE_Core::AbstractOperation
+ class: CharacterString [0..1]
+ temporalReference: TemporalReference
+consistsOf
+belongsTo
0..*
{ordered}
1..*
1..*
{ordered}
+consistsOf
+isDerivedFrom 0..1
InputParameter
OutputParameter
+inputScenarioParameter
0..*
+outputScenarioParameter
0..*
«featureType» Scenario_ADE_Core:: AbstractSimpleOperation
«featureType» Scenario_ADE_Core:: ComplexOperation
«dataType» Scenario_ADE_Core::ScenarioParameter
+ aggregationType: AgregationTypeValue [0..1]
+ class: CharacterString [0..1]
+ constraintType: ConstraintValue [0..1]
+ description: CharacterString [0..1]
+ name: CharacterString
+ parameterValue: GenericAttribute
+ simulationProperties: SimulationProperties [0..1]
+ temporalAggregation: TemporalAggregationValue [0..1]
«featureType» Scenario_ADE_Core:: RemoveFeature
+ FeatureRef: URI
«featureType» Scenario_ADE_Core:: ChangeFeatureAttribute
+ attributeRef: URI
+ newAttributeValue: GenericAttribute
«featureType» Scenario_ADE_Core::AddFeature
+ newFeature: AbstractFeature
+ parentFeatureRef: URI [0..1]
«union» Scenario_ADE_Core::GenericAttribute
+ booleanValue: boolean
+ dateValue: Date
+ doubleValue: double
+ geometry: GM_Object
+ integerVaue: int
+ measureValue: Measure
+ scaleValue: Scale
+ stringValue: CharacterString
+ timeSeriesValue: _TimeSeries
+ URIValue: URI
«enumeration» Scenario_ADE_Core:: ConstraintTypeValue
=
<>
>
>=
<
<=
+
-
«enumeration» Scenario_ADE_Core:: TemporalAggregationValue
hour
day
week
month
year
other
«enumeration» Scenario_ADE_Core:: AggregationTypeValue
sum
average
max
min
other

Figure 1. Excerpt of the UML diagram of the Scenario ADE, core module

define some initial conditions of the scenario (i.e. as input parameters) and can contain information about possible constraints, e.g. the desired target $CO_2$-emissions being lower than a certain level. Alternatively, the ScenarioParameter class is intended to model also any type of results associated to a scenario and obtained by an application, i.e. as output parameters. For this purpose, some attributes are defined to describe the origin of the data and the type and temporal level of aggregation. It is possible to store data of any type: Boolean, text, integer, real, etc., and time series data, too. Finally, Scenario parameters can be associated to specific city objects, thus allowing to be represented geographically by means of the geometries of the associated city object;

- The classes AddFeature, RemoveFeature, ChangeFeature-Attribute – derived from the abstract class AbstractOperation – can be used directly as simple, atomic operations (through AbstractSimpleOperation) or combined by means of the ComplexOperation class. These classes are used to add information about the operations carried out on a specific city model or city object in order to achieve the desired configuration the scenario parameters refer to. Starting from a certain city model, it is thus possible to describe how GML features were added, removed or changed. In this way, changes from one initial “urban configuration” can be recorded and documented, and there is no need to store multiple times those objects that do not change.

The Scenario ADE is implemented as a database schema extending the already available 3DCityDB. The approach follows the same rules used for the implementation of the Energy ADE (Agugiaro et al., 2018) and described in details by (Agugiaro and Holcik, 2017).

## 2.3 Data model for interactive optimization

The IO workflow within URB$^{io}$ using Mixed Integer Linear Programing and parallel coordinates is described by (Cajot et al., 2017). The focus here is to describe the workflow from a data perspective along with the corresponding data model (figure 2).

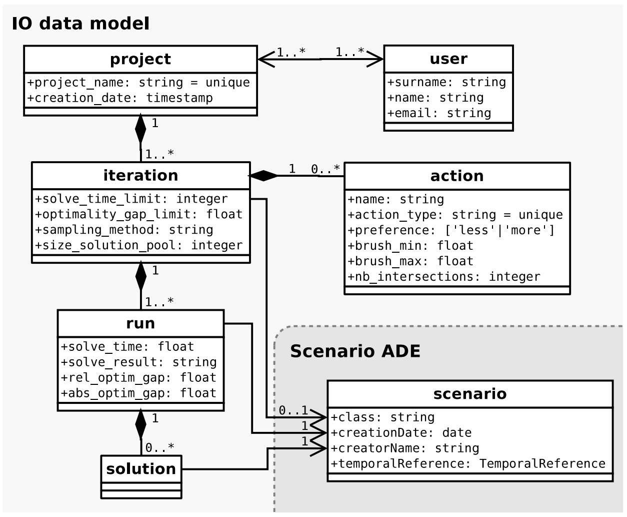

Figure 2. UML diagram of the data model for interactive optimization and its relation to the CityGML Scenario ADE

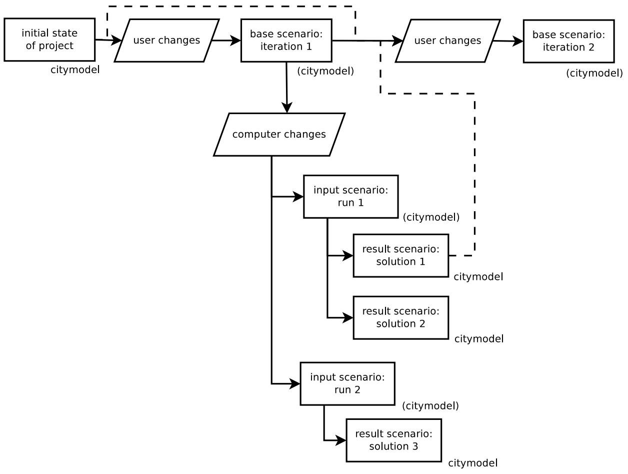

Figure 3. Flowchart of the interactive optimization workflow (dashed lines indicate alternative paths of the workflow)

**2.3.1 Projects & users** A project in URB$^{io}$ is owned by one or more users to allow them to collaborate on it. It is created by choosing a name (`project_name`) for the project and a city where the planning project is situated. Afterwards an initial spatial project perimeter can be chosen from the map, comprising the districts, parcels, or buildings of interest. This perimeter, however, can be still updated later in the workflow depending on the project's needs. Being interactive, this workflow is an alternation between changes made by the user and changes resulting from computer calculations (figure 3). It starts from a base scenario, which comprises initial information about city objects of the planning project at hand, like parcels on which to build or already existing buildings, and information considered immutable, as e.g. meteorologic conditions. From there the workflow proceeds iteratively.

**2.3.2 Iterations & actions** Each iteration starts with changes made by the user. These changes can concern information of three types:

- It can be methodology-specific information. This information is stored as attributes of the **iteration** class. It concerns the chosen sampling method, like systematic sampling (Thompson, 2012) or quasi-random sampling with e.g. Sobol sequences (Burhenne et al., 2011), to explore the decision space for criteria which shall be varied (`sampling_method`). Furthermore, optimization-specific settings can be stored such as an optimality gap (`optimality_gap`), which specifies e.g. the maximum expectable difference between a solution and the global optimum (Lawler and Wood, 1966), and a time limit (`solve_time_limit`), which cause the optimization to stop once either stopping criterion is met, and the number of solutions to return per optimization run (`size_solution_pool`). While, in principle, these changes could also be made using the parallel coordinate interface, they are more commonly made using forms, buttons, and dropdown menus.

- The changed information can be context-specific information which is not geo-referenced. These changes are mainly made using the parallel coordinate interface by selecting and brushing the displayed axes (`name`) in order to define actions of different types (`action_type`) to be carried out on the associated criterion: (a) If the action is to optimize a criterion, it can be either maximized or minimized depending on the selected `preference`. (b) If the action is to constrain a criterion, it is assured to stay above or below certain limits (`brush_min` / `brush_max`, `preference`). (c) If the action is to explore the range of a criterion, it will be varied systematically within this range (`brush_min`, `brush_max`, `preference`) according to the number of desired axis intersections (`nb_intersection`) if the sampling method requires so. The actions on the different axes are instances of the **action** class.

- Finally, the changed information can be context-specific information which is geo-referenced e.g. the number of floors of an already existing building or the permission to build ground source heat pumps on a specific parcel. These changes are stored using the operation mechanism of the Scenario ADE (see section 2.2).

**2.3.3 Runs** Depending on the chosen sampling algorithm and optionally how many intersections per axis (`nb_intersections`) were requested by the user, the algorithm generates input scenarios for optimization runs by translating the axes actions into values for the parametrized constraints or objective weights, respectively. These values are stored as scenario parameters (see section 2.2). An optimization run is further characterized by optimization-specific information, namely the actual time needed to finish (`solve_time`), its result status (`solve_result` e.g. solved), and the optimality gap in relative and absolute terms of its final solution (`rel_optim_gap`, `abs_optim_gap`).

**2.3.4 Solutions** Depending on the user settings, each optimization run can generate zero, one or many solutions: (a) If no solution is generated this means that no feasible configuration respecting all constraints could be found, at least not before reaching the solve time limit. (b) A single stored solution is always the best performing configuration found within the time limit or the optimality gap. (c) If a pool of solutions is requested from an optimization run (`size_solution_pool`), several solutions are stored. In contrary to solutions from different runs, these solutions are required to all respect the same, run-specific constraints. However, they differ in the choices for decision variables, thus still resulting in different plans. Consequently all but one of the solutions are sub-optimal. For every solution a result scenario is stored along with its associated operations and scenario parameters.

**2.3.5 Continuation of workflow** An iteration stops once either all requested runs are finished or the user interrupts the generation of new runs. Since every solution is loaded into the interface as soon as it is available, the user can already start to explore the solutions found so far while still new solutions are created. Based on the gained insights, and if the results are not yet satisfying, the user can initiate a new iteration by deciding to (a) proceed from any of the found result scenarios, adopting the intermediate changes on input parameters or city object attributes or (b) proceed from the base scenario of the current iteration or even (c) revert previously made changes to proceed from an earlier state of the planning project, be it a base or a result scenario. The workflow continues with the steps described in section 2.3.2.

### 2.4 Link between the IO data model and the Scenario ADE

The link between the IO data model and the CityGML Scenario ADE consists in the creation of scenarios for instances of one of the three classes **iteration**, **run** or **solution** (section 2.3). The relations between those scenarios and these IO class instances are realized using an associative table. The actions, as explained in section 2.3.2, are translated internally into input information for the optimization. These are assigned to a run scenario and stored as parameters of `type` “input” using the scenario parameter mechanism of the Scenario ADE (see section 2.2). The `class` attribute is used to further differentiate between objectives and constraints (figure 2). For parameters of type *objective*, the parameter value can e.g. indicate a weight for the weighted sum method for multi-objective optimization (Marler and Arora, 2010). The attribute `constraintType` is used to indicate if this parameter shall be maximized (+) or minimized (-). Those attributes are used likewise for constraints, in order to fix parameters at certain values (=) or to specify upper ($\leq$,$<$) or lower bounds ($\geq$,$>$).

Optimization results are mainly stored in form of scenario operations, which are assigned to an according solution scenario. Only information that can not be retrieved by aggregating the stored, finer grained information of e.g. all buildings of a scenario, is stored explicitly as scenario parameters of type “output”.

Finally the inheritance mechanism between scenarios is employed to allow for the different options listed in section 2.3.5.

## 3. APPLICATION

The developed and implemented concepts are demonstrated via an URB$^{io}$ use case. The focus here is on describing the correlation between the workflow and the resulting dataflow. The use case itself is described in detail in (Cajot et al., 2018).

The chosen planning project concerns the further development of an existing neighborhood of about 300 buildings in the canton of Geneva, Switzerland. One overall political goal is to react to urbanization by increasing the density of the zone in terms of its floor area ratio (FAR). In this example this shall be achieved by building new floors on top of already existing buildings while respecting legal building height constraints. Another goal is to increase the share of energy coming from renewable sources (RES). A new project is therefore created in URB$^{io}$ , resulting in entries in the project table (figure 2). The project starts at the present state of the neighborhood (figure 3), which is characterized by the existing buildings and their current energy supply systems stored in the 3DCityDB. In order to determine the maximum achievable density, the user first decides to maximize the FAR without specifying further constraints and accepting the default optimization-specific settings. This is done by brushing the according axes of the interactive parallel coordinate plot. These actions result in entries in the iteration table and action table (section 2.3.2). Based on those entries, an optimization run is created and executed. Its methodology-specific outcomes are then stored in the run table, while all context-specific results, i.e. the actually resulting scenario of the neighborhood, are stored in the CityGML tables as well as the Scenario ADE tables as explained in section 2.2. Aggregated information for each solution is then displayed in the parallel coordinate interface for the user to explore. Here the first iteration revealed that the density of the neighborhood could be increased from currently 1.17 to maximum 1.26.

In the following step, the user decides to explore the decision space by defining ranges for the FAR and the share of RES. In a first iteration they opt for larger ranges, and in a second iteration they choose narrower ranges. The objective is kept to minimize the total costs implied by the installed energy systems. The definition of ranges results in the generation and execution of several optimization runs which differ in their parametrized constraints. The outcome of these iterations can be seen in figure 4.

Since a further political goal is to reduce the number of existing oil boilers in the neighborhood, the user plots the according axis and finds that there are still many of them, even for high shares of renewables. A map of an according scenario calculated during the last iteration can help to understand this result (figure 5). The map reveals that the annual energy demands are mainly covered by the heating network, which implies that the already existing oil boilers are only used to cover peak demands. In a further iteration the user might now decide to decrease the number of oil boilers by defining an upper constraint on it.

## 4. CONCLUSIONS

This article presented work carried out in order to employ the CityGML data model for the planning support system URB$^{io}$ . This employment was enabled by the definition and implementation of the Scenario Application Domain Extension (ADE) for the CityGML standard, which allows to cope with different scenarios according to which a city could evolve. It further required the definition of a data model for interactive optimization. The usability of the developed concepts was finally demonstrated by means of a use case of URB$^{io}$ .

In principle, these concepts are ready to be tested and employed by other software applications. For the IO data model this implies other applications than urban planning. Thus ongoing work concerns the integration of the IO concept in the planning tool “EnergyScope”, which treats the generation of scenarios for the energy transition in Switzerland on a national scale (Moret et al., 2014).

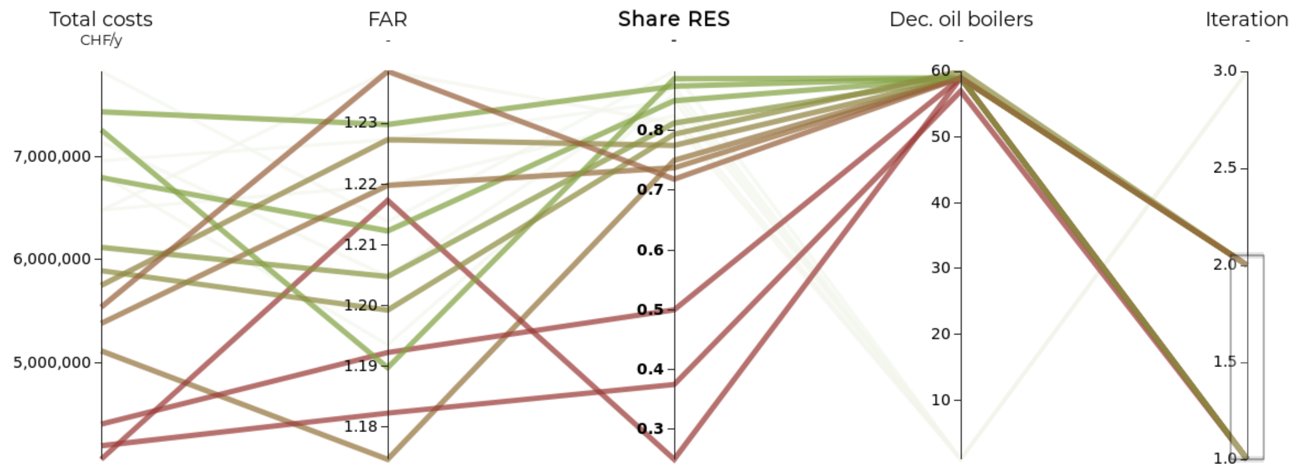

Figure 4. Parallel coordinate plot of solutions created via interactive optimization in URB$^{io}$ and stored via the proposed data model and the CityGML Scenario ADE (solutions are colored according to share of RES)

Analogously to CityGML, the Scenario ADE was intentionally conceived and implemented as a generic, application-independent data model. In other words, it can be employed in other software applications and case studies which might rely on methods other than optimization. Here, for example, another possible usage would be to test the Scenario ADE to link scenario management with co-simulation tools. Furthermore, it is still required and planned to develop visualization tools allowing to quickly display and compare side-by-side urban scenarios.

One of the main disadvantages due to the novelty of the Scenario ADE is that its adoption might be hindered by the lack of currently available auxiliary software tools for CityGML supporting ADEs, such as e.g. the 3DCityDB Importer-Exporter. However, research and implementation work is being carried out at the time of writing by the 3DCityDB development team in order to overcome this shortcoming.

## ACKNOWLEDGEMENTS

The work concerning the interactive optimization data model and its linking to the CityGML data model was carried out within the project “FURIES”[5], which is funded by the Swiss Competence Center for Energy Research. The work regarding the development of the Scenario ADE was carried out within the IntegrCiTy project[6], which, in Austria, is funded by the Austrian Federal Ministry for Transport, Innovation and Technology within the framework of the ENERGIE DER ZUKUNFT/JPI Urban, with support from the European Union’s Horizon 2020 research and innovation programme. All supporting agencies are gratefully acknowledged.

[5]`https://sccer-furies.epfl.ch/`
[6]`http://iese.heig-vd.ch/projets/integrcity`

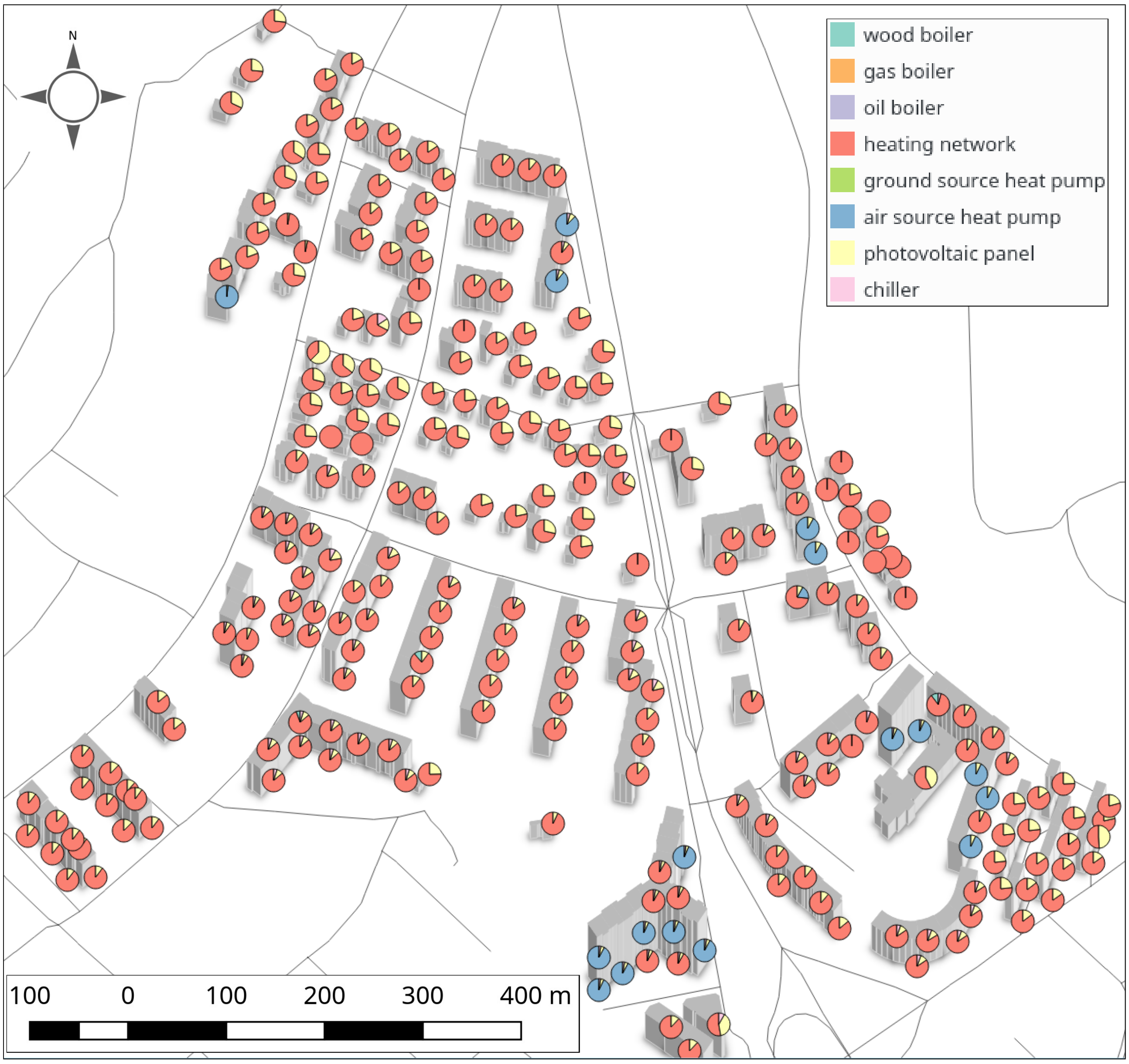

Figure 5. Map showing the building height and the annual energy supplied by the installed technologies: Heat is mainly supplied by the heating network while many buildings have PV panels installed

## NOMENCLATURE

**Acronyms**

| | |
|---|---|
| $ADE$ | Application Domain Extension |
| $FAR$ | floor area ratio |
| $GML$ | Geographic Markup Language |
| $IO$ | Interactive Optimization |
| $RES$ | renewable energy sources |

## REFERENCES

Agugiaro, G. and Holcik, P., 2017. 3D City Database extension for the CityGML Energy ADE 0.8 PostgreSQL Version - Documentation. Technical report.

Agugiaro, G., Benner, J., Cipriano, P. and Nouvel, R., 2018. The Energy Application Domain Extension for CityGML: Enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards* 3(1), pp. 2.

Balling, R. J., Taber, J. T., Brown, M. R. and Day, K., 1999. Multiobjective Urban Planning Using Genetic Algorithm. *Journal of Urban Planning and Development* 125(2), pp. 86–99.

Benner, J., 2017. KIT proposals for the next version of the Energy ADE.

Branke, J., 2008. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Lecture notes in computer science, State-of-the-art survey, Springer, Berlin ; New York.

Burhenne, S., Jacob, D. and Henze, G. P., 2011. Sampling based on Sobol’ sequences for Monte Carlo techniques applied to building simulations. In: *Proceedings of Building Simulation 2011*, Sydney.

Cajot, S., Schüler, N., Peter, M., Koch, A. and Maréchal, F., 2017. Interactive optimization for the planning of urban systems. In: J.-L. Scartezzini (ed.), *Energy Procedia*, Vol. 122, Elsevier, Lausanne, Switzerland, pp. 445–450.

Cajot, S., Schüler, N., Peter, M., Koch, A. and Maréchal, F., 2018. Interactive optimization with parallel coordinates: Exploring multidimensional spaces for decision support. *Frontiers in ICT*.

Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T. and Kolbe, T. H., 2017. Managing Versions and History Within Semantic 3D City Models for the Next Generation of CityGML. In: *Advances in 3D Geoinformation*, Lecture Notes in Geoinformation and Cartography, Springer, Cham, pp. 191–206.

CityGML, 2018. Cities around the world with open datasets.

Gal, T., 1995. *Postoptimal Analyses, Parametric Programming, and Related Topics*. 2nd ed. 1995. reprint 2010 edn, De Gruyter, Berlin, Boston.

Gröger, G., Kolbe, T. H., Nagel, C. and Häfele, K.-H., 2012. OGC City Geography Markup Language (CityGML) Encoding Standard.

Kolbe, T. H., Yao, Z., Nagel, C., Redweik, R., Willkomm, P., Hudra, G., Müftüoglu, A. and Kunde, F., 2016. 3D City Database for CityGML.

Lawler, E. L. and Wood, D. E., 1966. Branch-And-Bound Methods: A Survey. *Operations Research* 14(4), pp. 699–719.

Marler, R. T. and Arora, J. S., 2010. The weighted sum method for multi-objective optimization: New insights. *Structural and Multidisciplinary Optimization* 41(6), pp. 853–862.

Moret, S., Codina Gironès, V., Maréchal, F. and Favrat, D., 2014. Swiss-EnergyScope. ch: A Platform to Widely Spread Energy Literacy and Aid Decision-Making. In: *Chemical Engineering Transactions*, Vol. 39, pp. 877–882.

Rittel, H. W. J. and Webber, M. M., 1973. Dilemmas in a general theory of planning. *Policy Sciences* 4(2), pp. 155–169.

Schüler, N., Cajot, S., Peter, M., Page, J. and Maréchal, F., 2018. The Optimum Is Not the Goal: Capturing the Decision Space for the Planning of New Neighborhoods. *Frontiers in Built Environment*.

Sindram, M. and Kolbe, T. H., 2014. Modeling of urban planning actions by complex transactions on semantic 3D city models. In: *Proceedings of the International Environmental Modelling and Software Society (iEMSs)*.

Thompson, S. K., 2012. *Sampling*. Wiley series in probability and statistics, 3rd edn, Wiley.

van den Brink, L., Stoter, J. and Zlatanova, S., 2013. UML-Based Approach to Developing a CityGML Application Domain Extension. *Transactions in GIS* 17(6), pp. 920–942.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T. and Kolbe, T. H., 2018. 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards* 3, pp. 5.