2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)

Model-Driven Design of City Spaces via Bidirectional Transformations

Ennio Visconti
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy

Christos Tsigkanos
Distributed Systems Group
TU Wien
Austria

Zhenjiang Hu
National Institute of Informatics
University of Tokyo
Japan

Carlo Ghezzi
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy

***Abstract*—Technological advances enable new kinds of smart environments exhibiting complex behaviors; smart cities are a notable example. Smart functionalities heavily depend on space and need to be aware of entities typically found in the spatial domain, e.g. roads, intersections or buildings in a smart city. We advocate a model-based development, where the model of physical space, coming from the architecture and civil engineering disciplines, is transformed into an analyzable model upon which smart functionalities can be embedded. Such models can then be formally analyzed to assess a composite system design. We focus on how a model of physical space specified in the CityGML standard language can be transformed into a model amenable to analysis and how the two models can be automatically kept in sync after possible changes. This approach is essential to guarantee safe model-driven development of composite systems inhabiting physical spaces. We showcase transformations of real CityGML models in the context of scenarios concerning both design time and runtime analysis of space-dependent systems.**

***Index Terms*—Bidirectional Model Transformations, Model-driven Engineering, CityGML, Cyber-physical spaces**

I. Introduction

Contemporary buildings and urban areas often are complex spatial environments, hosting computational elements as well as humans and provide different kinds of functionalities, typically to enable various forms of interaction. As societies evolve and complexity grows, engineering complex cyber-physical systems inhabiting spatial environments presents new challenges, in typical scenarios dominated by information from multiple domains and the need for assurances regarding the overall system's behavior.

The development of such space-dependent, cyber-physical systems demands for software engineering support facilities that span their lifecycle, from design to operation. Engineering can be enabled with model representations of their spatial environment [1]; such representations can be sourced from domain models originating in other disciplines and dominated by their practices, tools and domain knowledge. Design tools and approaches, as used in civil engineering and architectural informatics, frequently produce artifacts which are geometrical or geographical representations describing physical spaces, such as buildings or cities. Although relying on international standards and accessible in machine-readable formats, such physical space descriptions [2], [3] are still often intended for static documentation or domain-specific purposes. The resulting models are therefore of non-easily analyzable types, which hinders their consideration for engineering software-intensive, composite cyber-physical systems [4]. The domain models we consider conform to the CityGML [5] standard which also encompasses buildings (Building Information Models – BIM [6]), widely used in practice for descriptions of buildings and cities, for which numerous real-world models are becoming available [7].

Our work grounds on model-driven principles and aims at the development of integrated and open tool environments for systematic model-based engineering of space-intensive systems, on top of traditional spatial descriptive models currently used in practice. Our approach entails automatic synchronizations between spatial domain models and generic, graph-based analyzable models. The idea is to use exactly the same spatial domain models used by practitioners to represent urban areas, buildings and city spaces and project from them some abstract and more computationally convenient representation, which can be transformed back to the original one when needed. The analyzable models we target are formally modeled topological structures –*cyber-physical spaces* [4]– enjoying well-defined semantics, where formal reasoning can be performed. Cyber-physical spaces are composite models integrating human agents, computational and physical aspects of systems useful for analysis. Our proposed formal programming technique and technical framework assure that relevant information added, or changes applied to the domain (resp. analyzable) model are reflected back in the analyzable (resp. source domain) model automatically and coherently. The technique developed is rooted in the theory of bidirectional transformations, which guarantees that synchronization between models is consistent and well-behaved.

Our key contribution is a technical framework based on bidirectional model transformations to support engineering of space-dependent systems. The novel bidirectional reflection facilities we provide for domain and analyzable models can be readily used to (i) derive models from spatial models occurring in practice, since CityGML models of cities are widely available, and (ii) instrument modeling and analysis facilities for spatially-dependent cyber-physical systems. Thus, they have a high potential for adoption by the community. We further point out that application of bidirectional model transformations to physical space models describing cities has not been investigated before by the community. To provide

978-1-7281-2536-7/19/$31.00 ©2019 IEEE
DOI 10.1109/MODELS.2019.00-16

concrete evidence of the proposed model-based approach, we demonstrate that bidirectional transformations can be achieved in practice on real city models. The concrete realization of the proposed framework as a prototype is freely available as open source software.

The rest of the paper is structured as follows. Section II gives an overview of the proposed approach, while Section III provides necessary background, design goals and challenges. Section IV describes the design of a bidirectional transformation between city models and analyzable models. Section V presents tool support, while Section VI provides an assessment of the proposed approach over two case studies, targeting design and operation. Lastly, Section VII gives an insight of related work in the field, and Section VIII concludes the paper.

II. Model-Based Engineering of City Spaces

Engineering systems inhabiting physical spaces requires providing facilities that span their lifecycle, from design to validation of their requirements. Model-based engineering plays a crucial role, as representations of systems not only enable design but also analysis and runtime reasoning. This is evident in contemporary smart building or smart city applications, where the use of domain-specific models has long been recognized as beneficial. Representations of the physical space (such as CityGML, BIM, or GIS), originate in the respective *source domain*, typically from other engineering disciplines. Such representations may enable validation of the overall system's requirements. However, from an engineering perspective, such validation analysis cannot be performed upon domain representations; analyzable models, typically some graph-based abstractions, must be obtained from the source model and brought to the *semantic domain*. Keeping such domain models in sync with derived analyzable models is then crucial. Our approach is illustrated in Figure 1.

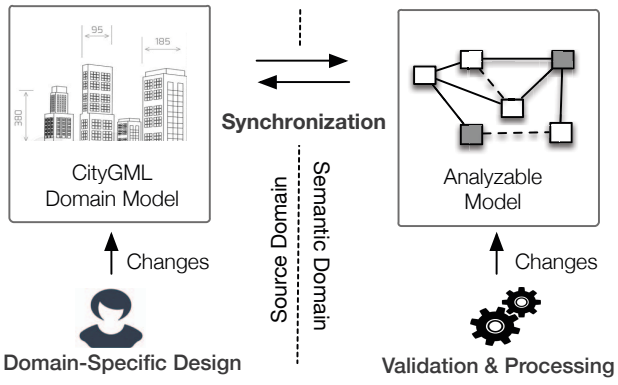


Figure 1: Bidirectional Model Transformations of City Spaces.

For a system at the design phase, the development cycle is naturally based on source domain models, usually encompassing GIS or BIM, depending on the design goals. CityGML provides a unified standard for all of them – think of an architect or urban planner altering the design of a city domain model such as changing road transportation routes. This occurs within the source domain and is dominated by the practices, the tools and the domain knowledge of the particular discipline (e.g., the transportation expert), such as civil engineering or urban planning. The overall system inhabiting the physical space specified by the CityGML description however, may need to satisfy certain quality attributes demanding particular kinds of reasoning. For example, the transportation expert may change the design in order to facilitate an emergency evacuation scenario. Such an activity is part of the composite system's development cycle.

Validation of a design against requirements entails an *analysis* activity. This analysis may be tailored to specific types of quality attributes and can be performed on some representation of the domain model, which is *analyzable* and thus situated in a semantic domain. For a transportation expert for example, analysis may entail e.g., simulating rescue teams within an evacuation scenario, by placing them in the analyzable model and reasoning on their behavior. Note that multiple semantic domains may be derived depending on different analyses that are sought. If the design changes, the analysis activity should be triggered again, to ensure requirements satisfaction on the changed design – this is often the case in exploratory processes. Moreover, if requirements are not satisfied, analysis processes may suggest or incur changes to the analyzable model in order to achieve a satisfactory design. The transportation expert for instance, may seek to visualize effects of rescue teams upon the city domain model, or investigate effects of the evacuation scenario bahavior upon other aspects captured in the domain model. Thus, the main contribution realized in this paper consists of synchronization facilities that ensure both that changes on the physical domain model (in our case a CityGML description) – are reflected in the analyzable model (a graph-based representation), as well as changes performed to the latter are reflected back to the former.

However, given the informational asymmetry between the two different types of models, properly synchronizing them is normally not a trivial task. When performing some operations to synchronize two models, the transformation is deemed correct if they are consistent (i.e. some equivalence relation is defined between the information contained in them) [8]. When the consistency relation has been defined, *bidirectional transformations* become a powerful tool to make sure the synchronization between the models is consistent and well-behaved. We note that bidirectional transformations have been of limited use in practical applications until now; we believe the present work can aid in better understanding some of the benefits and challenges engineers face in order to achieve a correct and meaningful reflection of changes between a domain-specific model and a more abstract one, within our particular domain.

III. Physical Spaces and their Representations

Engineering space-intensive cyber-physical systems can be enabled with model representations of their spatial environment. Spatial environment descriptions are typically found in

other engineering disciplines such as civil engineering, architectural informatics or architecture. We consider such spatial environment descriptions as source, *domain models*. Specifically, we adopt the ones used by practitioners to represent city-wide spaces (i.e. CityGML), since they also encompass buildings (i.e. Building Information Models, BIM [6]). However, CityGML descriptions are of non-easily analyzable types for the purpose of engineering software-intensive composite cyber-physical systems. To this end, we first briefly describe our source models, before succinctly defining the composite cyber-physical models we target. Those models are analyzable, enjoy well defined semantics, and can be used for software engineering purposes.

A. CityGML Descriptions as Source Models

CityGML as virtual 3D city models, have been widely adopted to analyze and take actions in a growing number of scenarios including urban planning, emergency management, traffic noise simulation, navigation systems, urban solar potential estimation, or visual communication [9], [10]. CityGML is playing a major role, given its ability to combine both thematical and spatial representations, in progressive levels of details [2], [5].

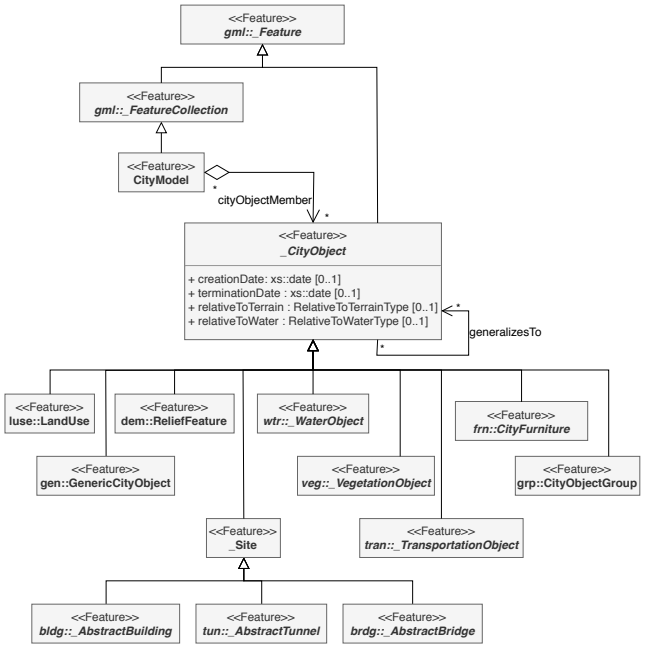


Figure 2: CityGML 2.0 top level class hierarchy addressed by our framework.

An interesting aspect of CityGML is the flexibility it introduces, by providing a way of defining Application Domain Extensions (ADEs), in which application requirements related to the city models can be described, while the enriched model still complies to the specification [11]. ADEs are formally defined extensions, specified in XML Schema Definition or Unified Modeling Language, capable both of adding new properties to existing CityGML classes and of adding entirely new classes and data types. For example, an ADE can be a set of extra attributes and elements nested into a standard CityGML model, to extend the capability of CityGML buildings in order to fully support Building Information Modeling descriptors. This also includes adding extra elements within the ADE, which reference standard CityGML objects and describe new relationships among them. These extensions can be arbitrary, ranging from geometrical aspects like shadow orientation to process-specific, like historical priority. More than 40 ADEs have been developed so far, with purposes including noise propagation, energy distribution, spatial topology and time variation among the others [11].

Despite being valuable sources of information, CityGML models' volume and domain-oriented design, make it challenging to actually consider them as a data source for complex analysis and operation, requiring huge application-specific preprocessing and postprocessing.

Our technical framework has been designed under the idea of automatically migrating both changes to the standard CityGML thematic features, shown in Figure 2, and a given ADE. To the best of our knowledge, despite the many CityGML ADEs available neither tools nor data are readily accessible for any of them as of today and, therefore, a preprocessing step is still needed in order to prepare the source information describing application-specific relationships, by referencing objects of the original city model. In the next section, to simplify the discussion, we will assume the `key()` and `children()` functions are properly defined with the purposes of providing a unique identifier of the CityGML feature and retrieving a list of sub-features respectively.

B. Cyber-Physical Spaces as Target Models

The analyzable models that we target are formally modeled topological structures specifically aimed at cyber-physical systems, termed *cyber-physical spaces* (CPSp) [1], [4] whereupon formal reasoning can be performed. We opted for this generic graph-based target model because of (i) its flexibility and applicability to various types of analyses and (ii) its formal semantics, allowing for a precise definition of the correctness of a transformation. Cyber-physical spaces are essentially graph-based representations of topological relations inherent in a space, which may span physical or computational barriers. This allows increased expressive power to represent complex systems and their interaction with active agents which may include devices, humans, software components or infrastructure.

Their formal semantics have been given in terms of bigraphs [12], a process meta-calculus consisting of two superimposed graphs. Such dynamic semantics are quite similar to graph transformation systems. For complete definitions and proofs of formal semantics, which are not covered in this paper, the interested reader can refer to the vast body of literature on the topic [12]. Scoped to our framework, bigraphs can be described in terms of the following components:

- A set of *labelled nodes* $\mathtt{v} \in \mathtt{V}$ which represent the elementary objects of the environment. In the following we will consider them as labelled with a pair $(\mathtt{identifier}, \mathtt{type})$, and we assume that a $\mathtt{key}(\mathtt{v})$ function returning the label is properly defined. In additon, we suppose that $\mathtt{findNode}(\mathtt{k}, \mathtt{S})$ is a function that returns a node v from the set $\mathtt{S}$ labelled with $\mathtt{k}$.
- A *place graph* is a forest, i.e. a set of rooted trees defined over nodes; this graph captures the notion of containment -nesting- of nodes. Given the structure of CityGML models, we can slightly simplify the discussion, considering that the containment relation develops from a single root representing the `CityModel` and, thus, the forest degenerates to a tree. In this perspective, we refer to $\mathtt{child}(\mathtt{n})$ for a node that has n as a parent in the containment relationship.
- A *link graph* is a hypergraph defined over the same set of nodes. Hyper-edges link any number of nodes; this graph represents generic links (i.e. many-to-many relationships) among nodes. Subsequently, we suppose that a proper function, similar in principle to $\mathtt{findNode}(\mathtt{k}, \mathtt{S})$, is available to find the links connecting a given node. Place and link graphs are orthogonal, and edges between nodes can cross locality boundaries.

Bigraphs allow us to achieve both the level of expressiveness needed by key topological characteristics and a high level of flexibility: the place graph defines a hierarchical structure, allowing us to model the locality in space of the city objects in terms of topological nesting, while the link graph can represent arbitrary connections among nodes (i.e. some other topological relation), enabling the representation of application-specific relations.

C. Synchronization: Design Goals and Challenges

In our view, model-based engineering of cyber-physical space-dependent systems should convey the following design principles, which underly our design of a bidirectional transformation between the two models:

1) Interoperability with well-established domain-specific standards and data models, namely CityGML and BIM as used in practice;
2) Provision of an actionable representation of the model in a non-domain-specific language that can enable complex analysis, in our case cyber-physical space reasoning;
3) Automatic composition of changed and unchanged parts of the model in a suitable way (i.e. well-behaved transformations), highly pertinent to both support of design activities as well as runtime model operations;
4) Decoupling of independent levels of reasoning (e.g. topological from geometrical) whenever possible, since those can be considered as being on different levels of abstraction.

Whichever the goals, it must be noted that the biggest challenge in synchronizing a highly detailed CityGML model (originating from domain-specific tools and practices) and an analyzable model (crafted for representing high-level application-specific features in terms of topological relations), relies in keeping the consistency between the two asymmetric sources of information in both the "*forward*" direction (i.e. the abstraction process) and the "*backward*" – or "*putback*" – one (i.e. the reification process). It is particularly the *putback* direction that needs special attention, since it requires *new* information to be generated, in order to fill missing details and produce a meaningful and consistent result in terms of practitioners' knowledge.

In the following, we illustrate how the above challenges may be tackled by designing and implementing a consistent and well-behaved bidirectional transformation (BX) between source city models and analyzable models which, by design, properly propagates changes when either one of the models is modified.

IV. Bidirectional Transformations Definition

At the core of any bidirectional model transformation, regardless of the direction, is the need of carefully defining when the information related to an object of the source and the one related to an object of the target are equivalent. This equivalence relation is usually referred in the literature as consistency between the two (or more) sources of information [13]. In the following, we first succinctly describe the laws underlying our transformation and the formalization of the consistency relation between source city models and cyber-physical spaces, then we sketch the algorithms implemented for consistency enforcement in our framework and lastly discuss some issues and limitations of the putback strategy in our approach.

A. Consistency Specification

Bidirectional transformations (BX) is a development methodology for maintaining the consistency relation between models, which can be expressed in terms of lenses [13]. A lens consists of a pair of transformations get and put [14]. The *forward* transformation $get(s)$ is used to produce a target view v from a source s, while the *putback* transformation $put(s, v)$ is used to reflect updates on the view v to the source s. We talk about asymmetric lenses when the transformations take place between two models where one side, which is called the source, has more information than the other, which is called the view. A pair of get and put should be *well-behaved*, in the sense that it satisfies the following round-tripping laws:

$$\begin{aligned} put(s, get(s)) &= s \quad \text{GetPut} \\ get(put(s, v)) &= v \quad \text{PutGet} \end{aligned}$$

The GetPut property requires that no change of the view shall be reflected to no change of the source, while the PutGet property requires all changes in the view should be completely reflected to the source so that the changed view can be computed again by applying the forward transformation to the updated source.

Concerning the models we investigate, the consistency relation can be formally specified in the following way; $\forall s, s'$ elements of the CityGML model, $\forall r$ relationship defined in

the CityGML ADE and $\forall v, v'$ nodes of the bigraph, we can say that s and v having the same keys ($key(s) = key(v)$) are *synchronized* ($s \rightleftharpoons v$) if and only if, the following conditions hold:

A.1 $instanceOf(s, \_CityObject) \wedge instanceOf(v, Node)$
A.2 $isContained(v, pv) \rightarrow childOf(s, ps) \wedge ps \rightleftharpoons pv$
A.3 $isLinked(v, v') \rightarrow holds(r, s, s') \wedge s' \rightleftharpoons v'$

The predicate $istanceOf$ guarantees an object is of the specified type, $childOf$ expresses the parent-child relationship of CityGML elements, while $isContained$ and $isLinked$ represent respectively containment and links of the bigraph. $holds$ describes both the presence of a relationship in the ADE and that its application-related meaning, somehow, holds.

We may say that a source model is *place-consistent* with respect to a view model if both **A.1** and **A.2** are satisfied. Likewise, we may say that it is *link-consistent* (w.r.t. a view model) if **A.1** and **A.3** are satisfied. When a source model is place-consistent and link-consistent at the same time, then it is *consistent* (i.e., the models are synchronized). Place-consistency has been fully formalized and therefore it can always be checked without ambiguity. This means that in no case we can have e.g., a road inside a building or similar irregular cases which are not allowed by the CityGML specification. On the other hand, link-consistency cannot be in principle solved unambiguously, since it is application-specific. This not-completely formalized approach is not new in BX, since, in some cases, local correctness checks (sometimes also called black-box operations) are needed in order to achieve consistency [15].

B. Consistency Enforcement

The three conditions described are enforced by our framework in the same order as presented. Algorithms 1 and 2 show the functions used for the putback transformation, which take a CityGML object and a node of the CPSp as input and return an updated version of the original entity within the CityGML description – the interested reader can refer to the accompanying artifact for complete implementations and further technical details. The corresponding functions for the forward transformation are automatically generated.

Our implementation has been developed in the BiGUL language, a putback-based bidirectional transformations language [16]. The strength of BiGUL, in comparison to other BX approaches, relies on the fact that only the putback direction has to be explicitly developed, as the forward direction is automatically derived by the language. This results in more flexibility for the modular design adopted, as, to support new application domains, it suffices to just change the *Application Policy*, a specific component to which Section IV-C is devoted.

Algorithm 1 exhibits the first stage of the synchronization logic, which, starting from the root of the city model and from the outermost node of the bigraph, traverses the two structures and repairs the differences by adding or removing the needed nodes at the correct position of the city model. Thus, at the end of its execution, the source model will be place-consistent (i.e., conditions **A.1** and **A.2** must hold).

Algorithm 1 Containment Graph Syncing

```
function syncCont(s :: _CityObject, v :: Node)
  for all o ∈ children(s) do
    if key(o) ∉ children(v) then
      REMOVE(s, o)
    else
      syncCont(o, findNode(key(o), children(v))
    end if
  end for
  for all n ∈ children(v) do
    if key(n) ∉ children(s) then
      ADD(s, n)
    end if
  end for
  return s
end function
```

Algorithm 2 Link Graph Syncing

```
function syncLinks(s :: _CityObject, v :: Node)
  rs ≔ getRelsWith(s)
  ls ≔ getLinksWith(v)
  for all l ∈ ls do
    if key(l) ∈ rs then
      rel ≔ findRel(key(l), ls)
      for all n ∈ nodes(l) do
        if key(n) ∉ rel then
          UPDATE(n, ls)
          break
        end if
      end for
    else
      UPDATE(n, ls)
    end if
  end for
  loop_on_ade_relationships()
  repeat syncLink() on children
  return s
end function
```

Conversely, Algorithm 2 describes the second stage of the synchronization. It also starts from the root of the two models, but it makes the assumption that the model is place-consistent, and therefore has the only goal of repairing relationships between the objects. It loops on the bigraph links and checks if the corresponding relation exists. If this is the case, a further check has to be done to verify that the relationship and the link reference the same elements. When these checks fail, a repairing procedure updates the source element. The same logic is mapped to the children nodes (the same in both models), until the source model is link-consistent (i.e., **A.3** holds).

Lastly, in both Algorithms 1 and 2, procedures in uppercase represent *Application Policy* actions, which play an important role in the transformation and are hence analyzed in the next

section. Illustrated functions highlight the main aspects of the effective transformations. For more details, the interested reader can refer to Section V and the accompanying artifact.

C. Dealing with Domain-Specifics

The algorithms previously introduced have been designed with the goal of satisfying the consistency conditions. However, condition **A.3** is not completely formalized, as application-specific requirements are, in general, unknown. This is because in principle, the reification strategy for new or removed objects may greatly vary depending both on the purpose of the specific object and on application scope and requirements.

Application Policy is the component appointed for ultimately verifying that task. Since different applications are likely to require different policies, application policy is an external component, interacting with our framework through clearly scoped interfaces called *actions*. Actions can access a limited set of information in order to achieve their goal, and they are required to produce an output that does not break previous assumptions.

The following actions have been defined:

- `ADD(s :: _CityObject, v :: Node) :: _CityObject`, which is bound to generate missing objects of the source. To that extent, it has access to all the information available from the parent of the target object. It is also allowed to change its actual representation (this is needed in some applications such as keeping spatio-semantic coherence) and it must return a new child having the key and type provided by the respective bigraph node.
- `REMOVE(s :: _CityObject, v :: Node) :: _CityObject`, which symmetrically to `ADD` has the purpose of removing extra objects from the source. It has access to the same information with the same constraints, albeit in this case it only returns the updated representation of the parent.
- `UPDATE(s :: _CityObject, v :: Node) :: _CityObject` is the most general action, responsible for both updating ADE relationships and potentially changing the representation of the current object. The problem of correctly reflecting a set of links may be very hard to solve in general. For this reason, our framework makes two simplifying hypotheses. Firstly, we assume that a change in a relationship (or the definition of a new one) can be fully expressed in terms of separated updates to the objects corresponding to the different nodes of a link. Moreover, we assume that the information required to address this task is limited to the subgraph of nodes and links related to the current one.

To understand the generality and thus the complexity inherent in `UPDATE`, consider a scenario in which we have two touching buildings, A and B in our cyber-physical space. A reasonable change could be, for example, to remove the *touching* relation between them and add a new one between B and C. Such an edit could be reflected in the original model in many different ways: a feasible result could be to just change the position of those objects. Nevertheless, another option could be to change the position of all the objects in the city, in order to satisfy the new requirement. Our framework can currently only deal with cases of the former, since the latter changes the model so significantly that it results in a completely different one, potentially triggering an endless loop of breaking-repairing operations in other areas of the model. The extent to which both these interfaces and their underlying assumptions are limiting is still a matter of active investigation.

V. TOPOCITY BX FRAMEWORK

In the previous section we presented the laws and the defined consistency relation, together with algorithms for guaranteeing them, as long as some domain-specific aspects and assumptions are met. To provide concrete support for our model transformation framework, we realized TOPOCITY, a prototypical tool implemented in Haskell, which is freely available for use[1]. TOPOCITY's main components are shown in Figure 3; its modular design allows for external component development and integration.

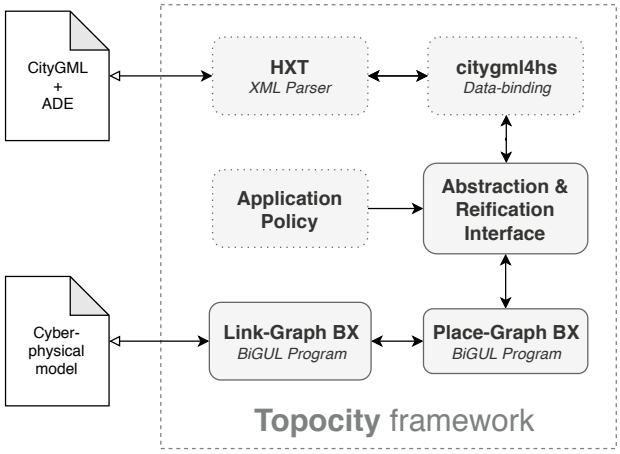


Figure 3: Combined view of architecture and dataflow of TOPOCITY. Dotted boxes represent external components.

The functionality of TOPOCITY revolves around two models, a source CityGML description, and a view, which is the CPSp model. The *HXT* component has been adopted for standard I/O operations like loading, parsing and storing the XML-based CityGML files. Auxiliary data-binding operations are supplied by the *citygml4hs* component implemented on top of HXT, providing a typed interface among data manipulations, which gives the capability to rely on type checking. The *Abstraction & Reification interface* is the software component delegated to deliver a common representation of citygml4hs types, so that more generalized BX could be defined on top of them. The *Place-Graph BX* and *Link-Graph BX* components make use of BiGUL primitives to implement Algorithms 1

[1] TOPOCITY– https://topo.city

and 2 respectively, while the *Application Policy* refers to the actions presented in Section IV-C.

To use TOPOCITY in practice, one follows four progressive steps:

1) Loading the source model (which is the pair of a CityGML and CityGML ADE description) by calling e.g., `load(city.gml, ade.gml)`.
2) Generating a CPSp target model (i.e. perform the *get* transformation) by simply calling `get(source)`.
3) Generating an updated source model (i.e. perform the putback transformation) by calling `put(source, view)`.
4) Storing the new source model in a file by calling `store(filename.gml)`.

In addition, we make available concrete analyzable models derived by TOPOCITY from real city models, obtained from publicly accessible repositories[2].

VI. EVALUATION: USE CASES

Our contribution consists of a technical framework implemented with formal programming techniques to support model-based engineering of space-dependent systems. To provide concrete tooling for our framework, TOPOCITY is based on BiGUL [16] and supports CityGML descriptions. The target models are graph-based and enjoy formal semantics [17]. Synchronizations between CityGML source models and target models are well-behaved and consistent as described in Section IV. Thereupon, we evaluate our approach over two exemplar cases where such bidirectional transformations can be used for engineering city space-intensive systems:

- During system development, analysis may be sought as part of an exploratory design cycle. In such a case, supporting validation is crucial. To this end, we demonstrate how an analyzable model can be automatically derived. TOPOCITY's synchronization facilities ensure that any changes on the CityGML spatial domain model are reflected to the analyzable model. To illustrate our approach, we showcase transformations upon a exemplar problem used in the civil engineering domain and scientific literature which concerns construction site layout planning: a tower crane positioning problem.
- During system operation, keeping an analyzable model alive can be instrumental in capturing contextual information received through monitoring. Analysis performed on this model can provide insights or serve as input to planning processes. Changes must be be reflected back to the source spatial model, to be visualized or combined with other domain-specific models that are interoperable with CityGML. TOPOCITY ensures that the updated analyzable model is consistent with the source CityGML model, by accordingly reflecting changes back. To illustrate such activities, we consider emergency response in a city.

The cases we consider for our evaluation purposes are model problems: they are representative cases where bidirectional transformation can play a big role in engineering an overall space-intensive system. We stress that the transformations inherent in the model problems presented are performed on real CityGML models obtained from public repositories, namely a district of Remscheid, North Rhine-Westphalia, Germany and Flat Iron Street in New York, USA. We conclude with a discussion.

[2]Topological city models repository – https://topo.city.

A. Facilitating System Design: Tower Crane Positioning

Proper optimization of construction sites layout is key to efficient construction activities. Before construction starts, site layout planning provides the necessary equipment and temporary facilities for the construction process, including allocation and dimensioning of elements like tower cranes, containers or storage areas. Decisions taken during this planning phase have direct impact on cost development and occupational safety on site during construction. Positioning of tower cranes is an important exemplar [18], [19]. Recent literature has provided techniques to automate the solution of this task, where two critical issues have been identified: (i) the lack of a simple but formal language capable of expressing rules, standards and best practices to check a building model [20], and (ii) the absence of tools able to perform this kind of operations by exploiting BIM/GIS descriptions like CityGML models, so that meaningful solutions can be found before implementation takes place [19]. In the following, we demonstrate how a flexible solution can be designed in which our framework plays a central role.

We consider an hypothetical construction site to be placed in a district of the city of Remscheid, North Rhine-Westphalia, Germany. For the real CityGML models we rely on North Rhine-Westphalia open data [21] – the linking structure related to tower crane positioning, is designed ad-hoc, since this step could be easily generalized and reproduced by modern user-guided CAD software [22]. Figure 4 shows the most relevant part of the model generated by our framework; an extra object and extra links are shown, corresponding to the changes made to the cyber-physical space in order to elicit the topological requirements for the new tower crane. Advanced analysis and model processing to generate such changes can take into account topological information in the analyzable model, such as proximity of construction site elements or complex relationships in the space layout, positioning the crane in a manner that satisfies some occupational safety or optimal placement requirements. As we are concerned with model transformations only, we consider such reasoning facilities as out of scope for this paper.

Once the target model is updated reflecting some reasoning (e.g., identifying the optimal position of the crane), changes have to be reflected back to the original model. To this end, TOPOCITY takes care of identifying changed objects and prompts the Application Policy to provide the 3D shape of the tower crane and spatial coordinates. For our case study, this was a fixed position, but a policy can specify arbitrary alternatives, from random to user-defined positioning, depending on the kind of links defined. Once those are given, TOPOCITY identifies the place in the original source hierarchy

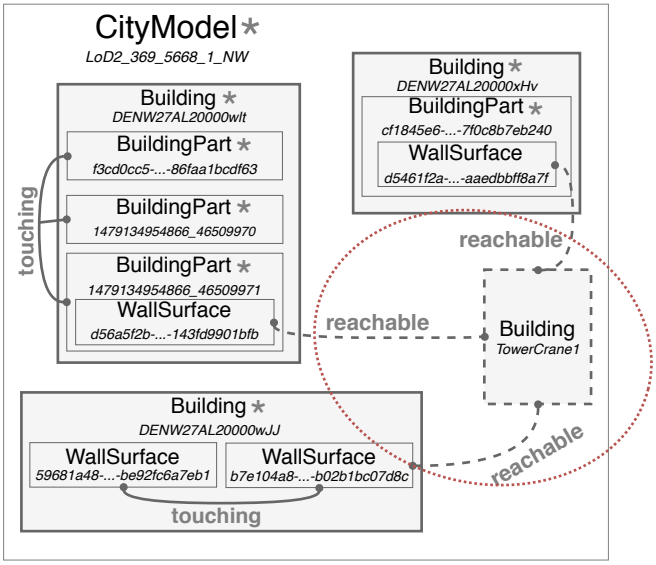
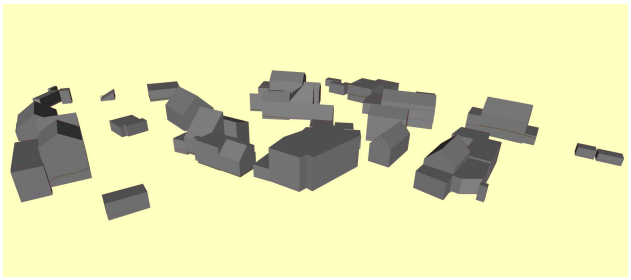
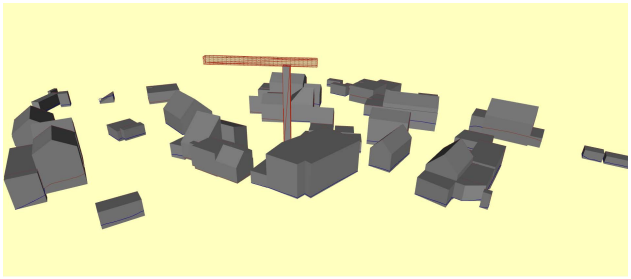


Figure 4: Fragment of the view model derived from the CityGML description of a district in Remscheid. Nodes are ID-Type pairs as they appear in the real CityGML model. Presence of other, not shown, elements of the model is indicated by *.



(a) Area without a tower crane (before).



(b) The crane is placed automatically via a *put* to the source model (after), reflecting its addition on the view model.

Figure 5: Placement of a crane entity on the derived, analyzable model (Fig. 4) entails its automatic reflection on the source city model (Fig. 5a), resulting in Fig. 5c.

to arrange the new objects and reifies the model back again to the CityGML description.

Figure 5 shows a fragment of the original model and the final result as visualized CityGML descriptions. Note how certain reachability links between edges of three buildings are additionally defined, supposing these are buildings of interest for the construction site (Figure 4).

B. Facilitating System Operation: Emergency Response

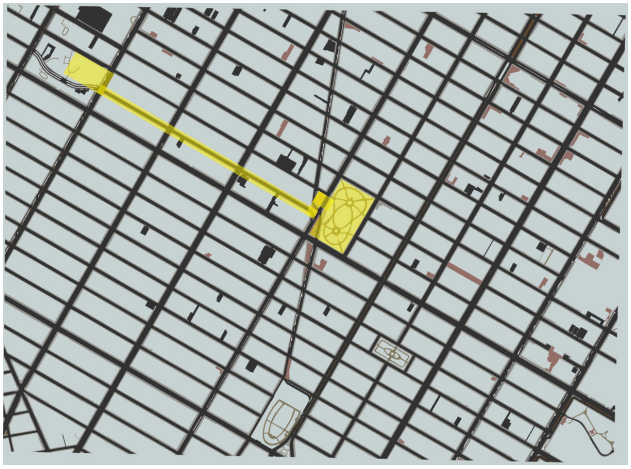
Technology adoption for fast emergency response in urban environments is gaining increasing attention: technological advances may in fact provide new human-computer interaction capabilities, allowing for effective real-time response. Consider the classical setting [23] where a disaster scenario is replicated in the Flatiron Building area of New York [24], with several relief entities (e.g. rescue teams, ambulances or Unmanned aerial vehicles – UAVs) dispatched throughout the area to locate and rescue victims [4], [25].

The agents have initial knowledge of the environment, given by the original model of the city. However, in such a scenario, we expect the model to be updated regularly, as soon as new information is acquired by monitoring processes. Agents must dynamically adjust search operations and rescue priorities through some criteria such as the likelihood of finding victims in an area or current disaster propagation. In order to perform such tasks, which largely amount to *planning* and *surveillance* [26], an actionable representation of the city can be a hypergraph in which nodes represent city objects, while links represent safe connections between multiple nodes. This typically occurs within a Monitor-Analyze-Plan-Execute loop, as this is an instance of a self-adaptive system. Agents monitor the area and update the model with the information they collect about safety of streets and buildings, while others escort civilians from the disaster area to hospitals. Path planning takes place based on analyzed monitored information upon the model, with the purpose of e.g., maximizing the number of victims rescued. Both the planning and monitoring facilities are not relevant for the synchronization of the models and they are therefore out of the scope of our research.

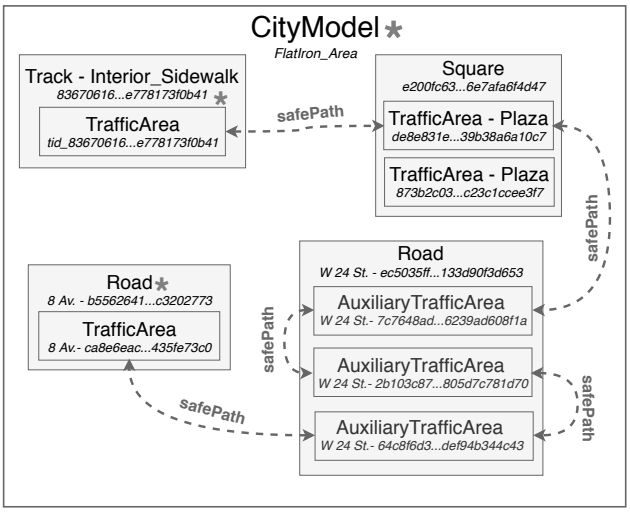
In our approach, we define and extract a CityGML ADE from the city model and populate it with real-time information, with the goal of making the *safe distance* relation between city objects explicit. TOPOCITY provides the hypergraph exploited by the agents, which is updated at runtime as the monitoring process generates new information. Figure 6 shows the aerial view of the Flat Iron Street area of New York as described by the CityGML model (6a), and the corresponding analyzable view (6b). A viable safe path for the city area is shown, both in the original model and in the analyzable one.

C. Discussion

We have demonstrated that, by using our framework, bidirectional model transformations upon real spatial descriptions can be performed, keeping analyzable models and CityGML descriptions synchronized. The two exemplar cases presented are different, as (i) they target different models and different levels of details within CityGML and (ii) they showcase uses



(a) The area nearby Flat Iron Street considered for our analysis.



(b) Fragment of the corresponding view generated.

Figure 6: Runtime safe path analysis models. The source (a) is transformed into the analyzable model (b). The highlighted area in (a) represents the safe path illustrated in (b). Nodes are ID-Type pairs as they appear in the available CityGML model of New York; the presence of other elements in parts of the model (not shown) is indicated by *.

of the framework for both systems design and operation. Hence, we believe they show the potential of our approach.

From our experience within model transformations of CityGML descriptions and considering the perspective of practitioners aiming to use our model-based engineering approach, interfaces and tooling integration might significantly support the design cycle. Moreover, we defer a performance evaluation of the transformations developed for future work, as our proof-of-concept tooling is in prototypical state.

A significant flexibility constraint has been briefly presented in Section IV-C. As anticipated there, links can be a very powerful medium for expressing arbitrarily complex configurations: in some convoluted scenarios, a putback to the original model may not be feasible or even worse, it may result in changes affecting a vast number of features, essentially resulting in a different model. We believe our solution addresses a relatively general set of meaningful applications, but further research on application scenarios may result in more precise understanding of practical limitations. Moreover, a considerable problem in making our framework an effective tool for practical use is the absence of any public ADE data or generation tool. Nonetheless, we believe this limitation may soon be overcome, thanks to the growing interest in the CityGML standard by domain experts [11].

An important aspect in BX design is the level of automation desired – ideally, one would expect to be able to choose an Application Policy that meets certain needs, plug it in our framework and use the combinations of these programs with no extra effort, regardless of the application context. However, our experience shows that some very complex CityGML features containing highly varying objects, still need some minimal custom bridging code to build the transformation. Tackling this problem in a generic manner requires extending the approach, something we identify as future work.

It is worth mentioning that [20] already solves the tower crane problem of Section VI-A by developing a plugin for Autodesk Revit –an established tool in building and urban design. However, as pointed out by the authors, only a small set of pre-defined simple rules are allowed, implemented ad-hoc for this purpose. In addition, [19] shows that GIS-BIM models (like CityGML) have enough information for treating the problem in terms of geometrical and topological analysis. Our approach, on the contrary, is general enough to allow for complex rules and user-defined customization if a proper Application Policy is set in place.

The two cases considered for our evaluation purposes are model problems obtained from domain-specific literature, highlighting the use of bidirectional transformations within our framework for model-based engineering of space-dependent systems. We believe that the strength of our approach is twofold: firstly, adaptability is exhibited, since integrating disparate application-related sources of information still result in the same analyzable model; secondly, providing an automatic way to obtain an abstract model where verification can be performed, can lead to the development of more sophisticated analysis-based workflows.

VII. Related Work

We have presented a novel technical framework to engineering bidirectional model transformations of city models, offering assurances on correct and well-behaved transformations. Consequently, we classify related work into three categories. First, we discuss the state-of-the-art in model-based analysis of physical spaces, positioning our work. Then, we review transformation techniques and theoretical foundations on consistency. Lastly, we discuss related engineering approaches

from the domain of analyzable models (i.e. cyber-physical systems that build upon spatial representations).

Interest on model-based analysis of cities has been consistently growing in recent years. The adoption of CityGML for building modeling purposes has been studied extensively lately [27]–[29] and the integration of classical BIM features has been a leading design goal [30] in defining CityGML 3.0, to be soon released [5]. In addition, city-based analysis is being developed in all kinds of application scenarios; most notably, recent efforts have been on traffic noise analysis [31], photovoltaic potentiality analysis [32], urban emission measurements analysis [33] and ubiquitous robot networks management [34]. Official city datasets are increasing, with recent public effort from Turkey [35], Singapore [36] and Germany [21] among all.

Consistency between models has been of interest for many years, historically originating from the view-update problem in database research [37], [38]. It has become relevant within BX and, in most recent approaches, it has emerged as a means for automatic generation of lenses [15]. BiGUL is a formally verified putback-based bidirectional programming language [16]. Alternative relevant approaches based on different types of lenses are QVT [15], Triple Graph Grammars [8] and Edit Lenses [39].

Different forms of graphs as formal models of static representations of buildings or cities have been proposed in diverse fields such as architectural informatics [40] or computer graphics [41], with different objectives. Several approaches target case-based reasoning [42] in the architectural domain. However, actionable and analyzable models are necessary for advanced design and operation of overall space-dependent systems [1]. In [40], a topology of spatial configurations is extracted from building information models as well as handwritten architectural sketches [43] and represented as graphs. Within the Internet of Things, analyzable models are extracted from trajectories and reasoned upon with a spatial logic [44]. Focusing on security reasoning while aiming at early design phases, Porter et al. [45] propose a method and heuristics to discover security threats on building specifications via simulation. Analyses such as similarity checking are performed based on graph matching techniques [46]. Forms of graphs representing topology of space are highly useful. To this end, our target analyzable models are graph-based and readily analyzable with a variety of approaches [47]. The notion of a cyber-physical space refers to a composite model able to capture complex relations of human, cyber and physical entities, which may span physical or computational barriers [48]. Such a model may be obtained from a physical model and enriched with formally-specified dynamics capturing possible ways it can change [49]. Spatio-temporal model checking of evolving cyber-physical spaces can then be considered [4], since topology can provide a system with awareness of multiple characteristics [50].

VIII. Conclusions and Future Work

Motivated by model-based design and operation of space-dependent systems, we have presented a technical framework enabling synchronizations between city spatial domain models and graph-based analyzable models. Synchronizations produced are automatically derived, correct and well-behaved. The spatial domain models we have considered are based on CityGML, as widely used by practitioners to represent city or building spaces. The analyzable models we targeted are formally modeled topological structures –*cyber-physical spaces*– enjoying well-defined semantics, where formal reasoning can be performed. The novel bidirectional reflection facilities we have provided for spatial CityGML and analyzable models can be readily used to (i) derive real models from spatial models occurring in practice and (ii) instrument modeling and analysis facilities for cyber-physical systems. We have further provided concrete implementations of the algorithms presented in the form of an accompanying artifact.

The presented approach and accompanying artifact is merely the first step – considering the perspective of practitioners aiming to utilize such BX facilities, we have identified several research directions that could be pursued. Interfaces and toolchain integration would go a long way in supporting design. This goes hand in hand with tackling practical issues of CityGML, such as public ADE data or generation facilities, to ensure effective tooling for practical usage. The class of synchronization problems addressed by our study must be clarified: our framework's main hypothesis is the unchangeability of the consistency relation between the source and the view. An alteration to the latter may result in massive rewrites of the core of our framework; this is, in fact, a common problem of solutions grounded on bidirectional transformations. It is therefore apparent that applications in which the consistency between the models is a relevant source of change cannot benefit significantly by our work. Regarding theoretical aspects, we aim to investigate pluggable custom application policies and support arbitrary CityGML features. Lastly, an aspect that can be further studied is relaxing the hypothesis of the unchangeability of the consistency relation between the source and the view in our transformations.

References

[1] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. Architecting dynamic cyber-physical spaces. *Computing*, 98(10):1011–1040, 2016.

[2] Thomas Kolbe, Gerhard Gröger, and Lutz Plümer. Citygml: Interoperable access to 3d city models. In *Geo-information for disaster management*. Springer, 2005.

[3] Chuck Eastman, Charles M Eastman, Paul Teicholz, and Rafael Sacks. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. J.W & S, 2011.

[4] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. Modeling and verification of evolving cyber-physical spaces. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, 2017*, pages 38–48, 2017.

[5] Open Geospatial Consortium. City Geography Markup Language (CityGML) Encoding Standard, version: 2.0.0. http://www.opengis.net/spec/citygml/2.0, 2012.

[6] G.A. van Nederveen and F.P. Tolman. Modelling multiple views on buildings. *Automation in Construction*, 1(3):215 – 224, 1992.

[7] CityGML open data initiatives. http://www.citygmlwiki.org/index.php?title=Open_Data_Initiatives, 2017.

[8] Frank Hermann, Hartmut Ehrig, Fernando Orejas, Krzysztof Czarnecki, Zinovy Diskin, and Yingfei Xiong. Correctness of model synchronization based on triple graph grammars. In *MoDELS*, 2011.

[9] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu ltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015.

[10] Sameer Saran, Kapil Oberai, Parag Wate, Amol Konde, Arnab Dutta, Kavisha Kumar, and A. Senthil Kumar. Utilities of virtual 3d city models based on citygml: Various use cases. *Journal of the Indian Society of Remote Sensing*, 46(6):957–972, Jun 2018.

[11] Filip Biljecki, Kavisha Kumar, and Claus Nagel. Citygml application domain extension (ade): overview of developments. *Open Geospatial Data, Software and Standards*, 3(1):13, Aug 2018.

[12] Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[13] Jeremy Gibbons and Perdita Stevens, editors. *Bidirectional Transformations*. Springer International Publishing, 2018.

[14] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Trans. Program. Lang. Syst.*, 29(3):17, 2007.

[15] Perdita Stevens. Bidirectional model transformations in qvt: semantic issues and open questions. *Software & Systems Modeling*, 9(1):7, 2010.

[16] Hsiang-Shang Ko, Tao Zan, and Zhenjiang Hu. Bigul: a formally verified core language for putback-based bidirectional programming. In *PEPM*, 2016.

[17] Robin Milner. Bigraphical reactive systems. In Kim G. Larsen and Mogens Nielsen, editors, *CONCUR 2001 — Concurrency Theory*, pages 16–35, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[18] Mohammed Adel Abdelmegid, Khaled Mohamed Shawki, and Hesham Abdel-Khalek. Ga optimization model for solving tower crane location problem in construction sites. *Alexandria Engineering Journal*, 54(3):519 – 526, 2015.

[19] Javier Irizary and Ebrahim Karan. Optimizing location of tower cranes on construction sites through gis and bim integration. *Electronic Journal of Information Technology in Construction*, 17:351–366, 09 2012.

[20] Kevin Schwabe, Markus Knig, and Jochen Teizer. Bim applications of rule-based checking in construction site layout planning tasks. 07 2016.

[21] Nordrhein-westfalen open geographic data. https://www.opengeodata.nrw.de/produkte/geobasis/3d-gm/, 2017.

[22] Revit products 2018 documentation - constraints definition feature. https://knowledge.autodesk.com/support/revit-products/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Revit-Model/files/GUID-4AD7D371-F757-4BFF-9F3C-8321A77D3A02-htm.html, 2018.

[23] Mei-Po Kwan and Jiyeong Lee. Emergency response after 9/11: the potential of real-time 3d gis for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems*, 29(2):93 – 113, 2005.

[24] 3d city model of new york city - tum. https://www.gis.bgu.tum.de/en/projects/new-york-city-3d/, 2015.

[25] Wesley DeBusk. *Unmanned Aerial Vehicle Systems for Disaster Relief: Tornado Alley*, chapter Unmanned Aerial Vehicle Systems for Disaster Relief: Tornado Alley. Infotech@Aerospace Conferences. American Institute of Aeronautics and Astronautics, Apr 2010. 0.

[26] Christopher M. Eaton, Edwin K. P. Chong, and Anthony A. Maciejewski. Multiple-scenario unmanned aerial system control: A systems engineering approach and review of existing control methods. *Aerospace*, 3(1), 2016.

[27] Junxiang Zhu, Graeme Wright, Jun Wang, and Xiangyu Wang. A critical review of the integration of geographic information system and building information modelling at the data level. *ISPRS Int. J. Geo-Information*, 7:66, 2018.

[28] Ken Arroyo Ohori, Abdoulaye A. Diakité, Thomas Krijnen, Hugo Ledoux, and Jantien E. Stoter. Processing bim and gis models in practice: Experiences and recommendations from a geobim project in the netherlands. *ISPRS Int. J. Geo-Information*, 7:311, 2018.

[29] Rudi Stouffs, Helga Tauscher, and Filip Biljecki. Achieving complete and near-lossless conversion from ifc to citygml. *ISPRS Int. J. Geo-Information*, 7:355, 2018.

[30] CityGML 3.0 requirements - munich 2013. http://en.wiki.modeling.sig3d.org/index.php/Workshop_Munich_2013, 2013.

[31] Amol Konde and Sameer Saran. Web enabled spatio-temporal semantic analysis of traffic noise using citygml. 2017.

[32] Nazmul Alam and Volker Coors. Detecting shadow for direct radiation using citygml models for photovoltaic potentiality analysis. 2013.

[33] Dirk Ahlers, Frank Alexander Kraemer, Anders Eivind Braten, Xiufeng Liu, Fredrik Anthonisen, Patrick Driscoll, and John Krogstie. Analysis and visualization of urban emission measurements in smart cities. In *EDBT*, 2018.

[34] Yaemi Teramoto, Akiko Sato, Kishiko Maruyama, and Hitoshi Tomita. Map representation for ubiquitous network robot services. In *Proceedings of the Fourth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, ISA '12, pages 29–32, New York, NY, USA, 2012. ACM.

[35] S. Ates Aydar, Jantien E. Stoter, Hugo Ledoux, E. Demir Ozbek, and Tahsin Yomraliolu. Establishing a national 3 d geo-data model for building data compliant to citygml : Case of turkey. 2016.

[36] K. H. Soon and V. H. S. Khoo. Citygml modelling for singapore 3d national mapping. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4/W7:37–42, 2017.

[37] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Trans. Database Syst.*, 6(4):557–575, December 1981.

[38] Terrence W. Pratt. Pair grammars, graph languages and string-to-graph translations. *Journal of Computer and System Sciences*, 5(6):560 – 595, 1971.

[39] Daniel Wagner and Nate Foster. Symmetric edit lenses : A new foundation for bidirectional languages. 2014.

[40] Christoph Langenhan, Markus Weber, Marcus Liwicki, Frank Petzold, and Andreas Dengel. Graph-based retrieval of building information models for supporting the early design stages. *Advanced Engineering Informatics*, 27(4):413–426, 2013.

[41] Raoul Wessel, Ina Blümel, and Reinhard Klein. The room connectivity graph: Shape retrieval in the architectural domain. In *The 16-th Intl Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008.

[42] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

[43] Sheraz Ahmed, Markus Weber, Marcus Liwicki, Christoph Langenhan, Andreas Dengel, and Frank Petzold. Automatic analysis and sketch-based retrieval of architectural floor plans. *Pattern Recognition Letters*, 35:91–100, 2014.

[44] Christos Tsigkanos, Laura Nenzi, Michele Loreti, Martin Garriga, Schahram Dustdar, and Carlo Ghezzi. Inferring analyzable models from trajectories of spatially-distributed internet-of-things. In *1th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2019, Montreal, Canada, May 25-26, 2019*. IEEE Computer Society, 2019.

[45] Stuart Porter, Terence Tan, Tele Tan, and Geoff West. Breaking into bim: Performing static and dynamic security analysis with the aid of bim. *Automation in Construction*, 40:84–95, 2014.

[46] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[47] Christos Tsigkanos, Nianyu Li, Zhi Jin, Zhenjiang Hu, and Carlo Ghezzi. On early statistical requirements validation of cyber-physical space systems. In *Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems, ICSE 2018, Gothenburg, Sweden, May 27, 2018*, pages 13–18, 2018.

[48] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. On the interplay between cyber and physical spaces for adaptive security. *IEEE Trans. Dependable Sec. Comput.*, 15(3):466–480, 2018.

[49] Christos Tsigkanos, Timo Kehrer, Carlo Ghezzi, Liliana Pasquale, and Bashar Nuseibeh. Adding static and dynamic semantics to building information models. In *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, pages 1–7. ACM, 2016.

[50] Liliana Pasquale, Carlo Ghezzi, Claudio Menghi, Christos Tsigkanos, and Bashar Nuseibeh. Topology Aware Adaptive Security. In *Proc. of the 9th Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems*, pages 43–48, 2014.