Modeling Semantics for Building Deconstruction

Alberto Paoluzzi[1], Christian Vadalà[1], Danilo Salvati[1], Federico Spini[2], Enrico Marino[2], Michele Vicentino[3], Antonio Bottaro[3]

[1]*Department of Mathematics and Physics, Roma Tre University, Rome, Italy*

[2]*Department of Engineering, Roma Tre University, Rome, Italy*

[3]*Geoweb S.p.A., Rome, Italy*

{paoluzzi, vadala, salvati, spini, marino}@ing.uniroma3.it, {mvicentino, abottaro}@geoweb.it

Keywords: Building modeling, BIM, Deconstruction Semantics

Abstract: In this paper we discuss the motivation, the technology, the design and the use-model of a novel web service for quantity surveyors, aiming to exploit virtual and augmented reality methods to implement a "zero waste" model, i.e. a new design paradigm where the waste materials from demolition become resources for reconstruction. The goal of this project is to provide virtual/augmented reality tools through quick modeling of buildings and their fast augmentation with semantic content.

1 INTRODUCTION

This project aims to contribute to a collective response to combatting climate change that afflict our times. In particular, a sensible contribution can come from a change of the *building life-cycle*, regarded as a "building organism" that can be brought back in line with the cyclical nature of natural phenomena. We need solutions that serve to *close the circle* of the building life-cycle, changing the traditional linear response (cradle to grave), greatly energy-consuming, toward cycles of materials reuse in deconstruction/reconstruction (cradle to cradle).

The tendency to not humanise new territory, but to better reuse as built and in disuse, is a compelling necessity in advanced societies. You must integrate the 'zero energy' model (each building has to produce the energy it consumes) with the 'zero waste' model, i.e. a new design paradigm where the waste materials from demolition become resources for reconstruction (Altamura, 2012).

Building process and design must be renewed to accommodate environmental concerns. To reduce the environmental impact of construction projects, the design has to deal with the issue of materials, very important concern for the governance of incentive policies for re-use. Public administrations need suitable tools for calculation and control of reused or disposed materials. The new instruments should handle the digital processing of materials in time, supporting new project requirements such as: Design for Deconstruction, Design for Recycling and Design for Waste.

All restructuring cycles should provide steps of deconstruction and re-construction targeted to the replacement of materials for the achievement of states of greater efficiency. Consideration of materials would require appropriate encoding both for disposal (CER – European Coding of Waste) and for the planning and design of the new (BIM – Building Information Modeling). For this purpose we need georeferenced scenes of augmented reality based on fast, easily navigable and measurable 3D modeling.

We already have an excellent knowledge of construction costs (from scratch) but little is known of the replacement rates (complete selective demolition). A modern selective demolition requires human intervention, with higher insurance costs for the danger of such interventions. This aspect opens the theme of "driving" automated robots to replace humans. In this scene it must be possible to drive robots without human presence (drones) to operate in a semantically known context, and giving real-time updates while reality contextually changes.

Therefore we believe that modern and easy-to-use modeling frameworks for building deconstruction in AEC industry are strongly needed, possibly augmented through semantic recognition by computer vision and by photogrammetric precision until to centimetric definition. Such virtual/augmented reality tools require both fast 3D building modeling and augmentation with semantic content, in order to be controlled in almost realtime: a real challenge also re-

quired by the future developments of IoT.

In this section we have discussed the motivation of the project described in this paper. The remaining sections are organized as follows. Section 2 introduces a more technical viewpoint about the state of deconstruction topics in Europe and in Italy. Section 3 shorthy recalls the methodology and the programming style and environment of our geometric computing approach to virtual and augmented reality. Section 4 describes the client application and its typical use-case. Section 5 illustrates the framework architecture. In the conclusion section we outline the work to be done and provide our forecast about possible developments.

2 BIM AND DESIGN FOR DECONSTRUCTION

La *gestione dei rifiuti* è un tema che negli ultimi decenni si è fatto sempre più attuale, considerando gli impatti economici, ambientali e energetici che da questa derivano.

Attraverso diverse Direttive che si sono susseguite nel corso degli ultimi anni (98/2008 EU, 1357/2014, âĂę), la Comunità Europea ha definito norme e obiettivi (qualitativi e quantitativi) che gli Stati Membri devono rispettare e ottenere - attraverso lâĂŹemanazione di regolamenti nazionali e la definizione di strumenti economici.

Lo smaltimento dei rifiuti, e dei costi diretti e indiretti che ne derivano, non è ovviamente estraneo nellâĂŹambito delle attivitÃă delle Costruzioni e delle Demolizioni.

Nella Classificazione Europea dei Rifiuti (CER) [955/2014 EU] [in inglese European Waste Code/Classification], i rifiuti che si producono in particolare dalle attività di demolizione di edifici e fabbricati, sono identificati da una specifica classe - la *17 - Construction And Demolition Wastes (Including Excavated Soil From Contaminated Sites)*.

Questa classificazione rende possibile, allo stesso tempo, sia una corretta identificazione del rifiuto prodotto dalle specifiche modalitÃă di demolizione adottabili sia delle azioni di *gestione del rifiuto* in termini di possibile *riuso, riciclo o di smaltimento in discarica*.

Ciò detto, il quadro normativo che si determina nel contesto della gestione dei rifiuti, in particolare nei singoli contesti nazionali, risulta tuttâĂŹaltro che chiaro.

Malgrado i costi derivanti dalla scelta di âĂIJsmaltimento in discaricaâĂİ, gli operatori del settore preferiscono questa strada piuttosto che rischiare di incorrere in sanzioni amministrative, o anche penali, per una mancata conformità a regole e norme poco chiare.

Per superare questo approccio non risulta certo utile, nella sostanza, imporre dâĂŹufficio delle soglie di minime di âĂIJgestione dei rifiutiâĂİ dalle attivitÃă di cantiere, che pure sono parte essenziale delle direttive Europee citate.

E‘ allâĂŹinterno di questo quadro normativo che rischia di essere *farraginoso* che riteniamo utile promuovere lâĂŹutilizzo di strumenti (di supporto) che, grazie una modellazione 3D dellâĂŹedificio capace di integrare anche una descrizione semantica delle parti che lo compongono e delle loro relazioni, possano favorire il superamento delle difficoltà amministrative come pure lâĂŹefficacia nella corretta identificazione dei rifiuti prodotti, e quindi delle opportune scelte fra le possibili azioni *virtuose*, ovvero quelle di recupero e di riuso.

A cui si aggiunge, in contesto di *modellazione*, la possibilità di identificare in modo puntuale i costi, cosÃň come i possibili ricavi derivanti dal percorso alternativo di *riciclo/riuso* rispetto allo smaltimento, oltre che la composizione e integrazione di informazioni utili alla pianificazione delle attivitÃă di cantiere.

Inoltre, in un *ambiente di modellazione e calcolo*, si rende possibile sia la verifica del raggiungimento delle soglie di *riuso/recupero* previste dalle normative (una sorta di *validazione dellâĂŹintervento sul cantiere*) che anche la possibilità di confrontare fra loro diverse opzioni

*** Spunti da sviluppare (?)

Uno strumento di editing —veloce— rende percorribile la strada di adempiere agli obblighi di legge

DallâĂŹabstract di questo articolo Building Information Modeling (BIM) for existing buildings âĂŤ Literature review and future needs (`www.sciencedirect.com/science/article/pii/S092658051300191X`)

mi sembra interessante la seguente lista di cose

Results show scarce BIM implementation in existing buildings yet, due to challenges of

(1) high modeling/conversion effort from captured building data into semantic BIM objects,

(2) updating of information in BIM and

(3) handling of uncertain data, objects and relations in BIM occurring in existing buildings.

dati circa lo elevato costo sociale, ambientale e economico delle materie prime utilizzate in edilizia

.

il 10-15% consumi energetici del settore C&D si deve alla estrazione delle materie prime (Rapporto United Nations Environment Programme: *Buildings*

and Climate Change 2002)

il 25% dei rifiuti prodotti (in Italia) deriva dal settore C&D

3 SOLID MODELING

The server-based core of the modeling architecture is based on a set of python libraries, including `pyopengl`, `scipy`, `pyplasm` and `larlib`, that provide our current implementation of the LAR (Linear Algebraic Representation) scheme for solid modeling, 3D imaging and mesh representation, and which is briefly described in the following.

3.1 Linear Algebraic Representation

LAR is a general-purpose representation scheme (Di-Carlo et al., 2014) for geometric and solid modeling introduced recently. The domain of the scheme is provided by dimension-independent *cellular complexes*, while its codomain is that of *sparse matrices*, stored using either the CSR (Compressed Sparse Row) or the CSC (Compressed Sparse Column) `scipy`'s memory format. The LAR polyhedral domain coincides with complexes of connected d-cells, even non-convex and/or including any number of holes.

The very general shape allowed for cells makes the LAR scheme notably appropriate for solid modeling of buildings and their components. E.g., the whole frontage of a construction can be modeled as a single 3-cell of its solid model. Also, the algebraic foundation of LAR allows not only for fast queries about incidence and adjacency of cells, but also to resolve—via fast SpMV computational kernels—the boundary extraction of any 3D subset of the building model.

It is worth noting that LAR provides a direct management of all subsets of cells and their physical properties trough the linear spaces of *chains* induced by the model partitioning, and their dual spaces of *cochains*. The linear operators of *boundary* and *coboundary* between such linear spaces, suitably implemented by sparse matrices, directly provide, depending on the dimension of the mapped spaces, the discrete differential operators of *gradient*, *curl* and *divergency*, while their product gives the *Laplacian*.

3.2 Geometric computing of shape

Our computational environment is strongly oriented towards the most general parametric modeling of component shapes of buildings. This attitude is produced by two Python libraries, that provide a dimension-independent algebraic calculus with shapes (`pyplasm`) and their representation in the LAR scheme (`larlib`).

PLaSM (Paoluzzi et al., 1995), which stands for Programming Language for Solid Modeling (Paoluzzi, 2003), is a geometry-oriented extension of the Backus' FL language (Backus et al., 1989). PLaSM is a project developed in the nineties in the framework of the *Building Technologies Project* (*"PF Edilizia"*) of the Italian National Research Council. The `pyplasm` module (2006-) is the C++ porting of PLaSM to Python via SWIG wrapping, providing several operators for fast and concise functional computing of shapes.

On top of the Scipy/Pyplasm stack we started (2012), using literate programming methods, to build a set of software modules, named `larlib`, and using the LAR scheme. This library provides the computation of topological queries and physical properties of meshes and complexes, including integration of polynomials over the boundary of any chain of cells, using fast algebraic methods with sparse matrices. For interactive visualization it relies on the `pyplasm` viewer, based on OpenGL. A porting of the most engaging parts of `larlib` to *Julia*, the last-generation programming language for scientific computing (Bezanson et al., 2014) started very recently, with the purpose of taking advantage of the great computational efficiency and parallelism of Julia in more demanding applications.

3.3 Plugin server framework

Our building deconstruction framework has a web-based client-server architecture, discussed in Section **??**. *Metior*, the web client application, is illustrated in Section **??**. The server-side of the framework, discussed in this section, is a plugin server written in Python, which capitalizes on the stack of geometric programming tools described above.

The Metior user quickly develops a 3D hierarchical assembly of different parts of the building envelope, as well as the horizontal and vertical partitions, using very simple 2D drawing tools. The more geometrically complex parts of the construction are conversely set up by user picking from context-based boards of predefined plugin templates, that are Python scripts generating solids models which are interactively dimensioned, partly by using 2D drawing tools, partly by user's numeric input from keyboard.

Of course, our list of *plugin templates* embraces most of building parts that are not manageable for quick shape input via 2D interaction. In particular, the picking boards include templates for planar concrete frames, spatial building frames, building foundations,

roofs and stairs of different types, attics and dormers, fireplaces and fitted wardrobes, shover cabins and sanitary equipments, doors and windows, etc.

It is worth noting that, by virtue of the great expressiveness of the PLaSM operators and its functional style of programming, using first-class curried functions and dimension-independent geometry, the development of a new plugin template is very easy even for non-experienced programmers, and usually requires a tiny amount of time and code, that may range between 4-8 hours, and between 10-100 lines of Python/pyplasm code.

Two important points to remark are the following: (a) the expressive power of a geometric language is greatly empowered by currying, i.e. by the technique of translating the evaluation of a function that takes multiple arguments (or a tuple of arguments) into evaluating a sequence of functions, each with a single argument; (b) Python/pyplasm are currently used even to teach (geometric) programming to K12 students (). Most of plugin templates currently used by Metior were developed in class by the students of the graphics course taught by one of the authors.

4 DECONSTRUCTION APP FOR SURVEYORS

Una decostruzione efficiente ed improntata al massimo riuso dei materiali, deve essere necessariamente supportata da una corretta metodologia che guidi l'utente verso una pianificazione e stima dei costi del processo. Verrà presentata di seguito la metodologia individuata e un'applicazione software che ne rappresenti l'implementazione. In particolare questo software focalizza la propria attenzione sulla determinazione dei costi di demolizione, smaltimento e trasporto dei materiali.

Nella stesura della metodologia sono state prese in considerazione software analoghi come SMART-Waste (Hurley, 2002). Questo, permette di ricavare le stime delle quantità dei materiali fornendo una descrizione della tipologia di edificio e della zona in cui è stato costruito. Grazie a queste informazioni, vengono automaticamente riempiti dei moduli che forniscono una rappresentazione aggregata dei dati di interesse. Il nostro approccio alla decostruzione, prevede invece una modellazione geometrica e semantica delle sue componenti. Infatti, il panorama edilizio italiano, è eterogeneo ed è quindi necessaria una modellazione di dettaglio per poter ricavare informazioni sufficiente precise.

Il vantaggio della nostra metodologia consiste anche nella possibilità di perseguire un approccio *iterativo-incrementale*, in cui ad ogni fase di modellazione può seguire una fase di validazione dei costi parziali ottenuti. Il risultato può essere eventualmente raffinato ciclando su tale processo. Di conseguenza, il software è stato pensato in modo da agevolare tale metodologia incoraggiando una modellazione che preveda diverse fasi.

4.1 Fasi della progettazione

Creazione progetto: La prima fase della modellazione consiste nella descrizione dell'edificio, in modo da fornire indizi fondamentali per una corretta attribuzione della semantica alle componenti. In particolare sono state identificate come caratteristiche fondamentali l'*età presunta*, lo *stile di costruzione*, lo *storico delle destinazioni d'uso* e la *geolocalizzazione*. L'età presunta e lo stile di costruzione permettono di determinare informazioni sui materiali utilizzati; lo storico delle destinazioni d'uso permette di ricostruire le note di pericolosità degli elementi da smaltire – ad esempio ci possiamo aspettare elementi pericolosi in un'azienda chimica – la geolocalizzazione consente infine di ricavare gli impianti di riciclaggio più vicini al sito.

Modellazione edificio: Durante questa fase l'utente descrive il fabbricato utilizzando alcune tipologie di elementi. Per prima cosa viene definito lo *scheletro dell'edificio* (struttura portante) , ovvero l'insieme delle travi e dei pilastri. Su di esso vengono poi costruiti i *muri* (partizioni interne), su cui vengono posizionati gli *infissi* (comunicazioni orizzontali). I *solai* e i *pavimenti* (chiusure orizzontali), invece, sono automaticamente generati a partire dalla topologia dello scheletro e dei muri. Infine, vengono collocati vari elementi come *scale*, *ascensori* (comunicazioni verticali) e *tetti* (chiusure orizzontali) terminando quindi la fase di modellazione dell'edificio.

Attribuzione della semantica: In questa fase agli elementi precedentemente inseriti, si attribuisce della semantica mediante *annotazioni*. Vengono definiti i materiali costituenti, che possono essere uno o più ed in particolare si definisce la densità. Ai fini dello smaltimento, è necessario attribuire uno o più codici CER (***QUI CI VA UN LINK AL PARAGRAFO DOVE SONO SPIEGATI I CER?***) ed il grado di pericolosità. Viene inoltre assegnato un attributo che fa riferimento al cronoprogramma di smaltimento delle componenti del fabbricato.

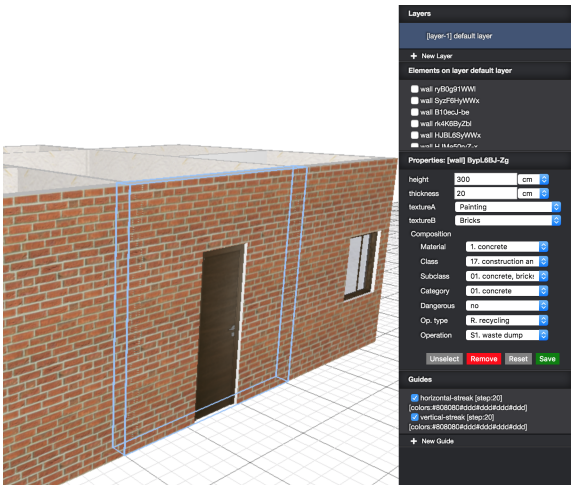

Figure 1: Interfaccia grafica per l'attribuzione di semantica ad un oggetto della modellazione

Visualizzazione in realtà aumentata: Completate le fasi di modellazione e attribuzione della semantica, si può validare l'intero modello immergendolo all'interno di una *point cloud* precedentemente ottenuta. (***QUI CI VA UN LINK AL PARAGRAFO DOVE SONO SPIEGATI I CER?***). In questo modo si può verificare l'aderenza del modello alla realtà, ripercorrendo eventualmente i passi precedenti se il risultato non è ancora soddisfacente.


Figure 2: A model inside a point cloud

4.2 Risultati finali

Terminate le fasi del workflow e validata la geometria del modello, l'applicazione fornisce una stima del costo di demolizione. Questo rappresenta l'output desiderato dall'utente in quanto permette di capire se le decisione prese sono corrette o conveniente dal punto di vista economico ed ambientale. Questo report finale si compone da quattro documenti:

- Una stima dei volumi dei materiali di ogni singolo componente, ricavata a partire dalle geometrie definite in fase di modellazione con opportuni calcoli di integrazione.
- Una stima dei costi di demolizione, smaltimento e recupero. Partendo dai volumi, si determinano le masse grazie alle proprietà dei materiali, mentre conoscendo i CER e quindi la modalità di smaltimento, si stimano i costi del conferimento in discarica.
- Una stima dei costi di trasporto necessari per trasferire i materiali dal sito di demolizione alla discarica. Per fare questo si tiene conto della posizione geografica del modello calcolando i percorsi stradali più convenienti.
- Un stima dei tempi previsti per la demolizione completa del fabbricato, collocati su un diagramma di Gantt. Per fare questo si utilizza l'informazione del cronoprogramma attribuito durante la fase di attribuzione della semantica.

5 DESIGN AND ARCHITECTURE

Workflow and requirements described in the previous section have been received in *Metior*, a prototypal application serving as proof of concept. With the aim of maximize accessibility for surveyors, it is strongly web based and runs in all modern browsers. It is built using React by Facebook and an MVC design pattern with *unidirectional data flow* (Abramov, 2016): it ensures the best code maintainability and debuggability by centralizing access to the application state to a single controller.

5.1 UI & UX

The web application presents itself as a simplified CAD, the UI comprises three main areas of interaction: *toolbar*, *canvas* and *sidebar*, as shown by Figure 4.
From the *toolbar* the user can access functionalities related to: 1) project life cycle (`new`, `save`, `load`); 2) project editing (`show-catalog`); 3) view/interaction mode switching (`2D`, `3D`); interaction mode changing (`selecting`, `pan`, `zoom`).
The *canvas* is the area in which the user can interact with actual model data. It supports two different view and interaction modes: 1) in *2D-mode* model is displayed as a 2D projection from the top, interaction consist of elements insertion, selection and editing (according to specific plugin interaction proto-

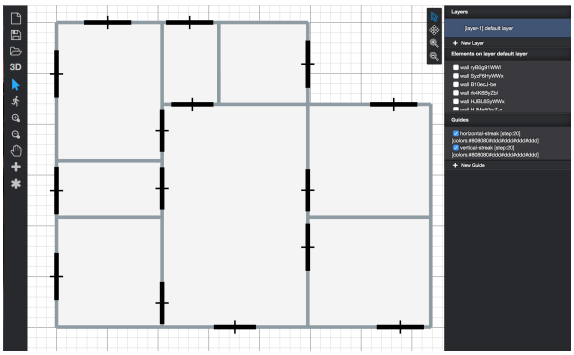

Figure 3: *Metior* user interface

Figure 4: *Metior* user interface

type, see 5.2.2); 2) in *3D-mode* 3D model can be inspected and navigated respectively via trackball or first-person interaction style, while object picking allows for element selection.
The *sidebar* shows properties of the currently selected element. In the properties panel it's possible to view the description of the element, to add/remove metadata and to modify any property. The latter is the interaction that allows the user to associate semantic to each part of the model.

5.2 Plugin-architecture

The application has been designed to provide a small set of core interaction functionalities and to encapsulate the generation logic for architectural components (from the very basic to the most articulated) into specific plugins.
A plugin is a software component that can be seamlessly integrated into the system in order to extends its capabilities. In *Metior*, a plugin represents an architectural element that extends the Building Information Model design. Technically, a plugin represents a *prototype* (namely a "class" in OOP) of a construction element that can be inserted ("instantiated") into the *canvas*, thus defining a new element, i.e. a new component of the model.

5.2.1 Plugin definition

A plugin is described by the following eight properties: 1) a unique name; 2) a description; 3) a set of metadata; 4) the *occupation type* (one among *linear*, *area* or *volume*); 5) the *placement type* (*inside* or *over*); 6) a set of specific properties mapping the semantic to associate to the plugin; 7) a construction function that returns the 2D representation of the element (in SVG format, to be used in the *2D-mode*); 8) a construction function that returns the 3D representation of the element (in OBJ format, to be used in the *3D-mode*)

5.2.2 Plugins taxonomy

The plugins can be organized according to the *occupation type* and the *placement type*.
In the *occupation type* three different kind of plugins can be identified: *linear*, *area* or *volume* plugins.
The *linear* ones extend in one dimension (unless a radial thickness) (e.g. hydraulic lines, electrical cables).
The *area* plugins extend in two dimensions (unless a linear thickness), (e.g. separation elements). They can be divided into *horizontal area* (e.g. floor and ceil), and *vertical area*, (e.g. walls). The *volume* plugins extend in three dimensions. They can be *fixed volume*, (e.g. a piece of furniture) and *scalable volume*, that can be scaled (proportionally or not), (eg. pillars, staircases).
The *occupation type* determines a different way to instantiate and to insert the plugins into the canvas. In particular, in *2D-mode*, *linear* plugins are inserted drawing lines by mean of a drag&drop interaction; the *area* plugins are inserted drawing the bounding-box of the element by mean of a drag&drop interaction; the *volume* plugins are inserted picking the position of the element by mean of a point&click interaction, and adjusting their dimensions modifying the bounding-box by drag&drop.
The *placement type* determines if the element can be inserted into the canvas in a specific point occupied or not by other elements.
The *placement type* determines the relationship between a new instance of a plugin and instances of other plugins previously added to the model. The relationship can be of two kind: *inside* or *over*.
Plugins belonging to the *inside* category can be added only inside other element (that can be *linear*, *area* or *volume*); e.g., a "window" is a "volume inside vertical area" element, while an "hydraulic line" is a "linear inside horizontal area" element.
Plugins of the *over* category can be added only over other elements (of any type); e.g., a "pillar" is a "vol-

ume over horizontal area" element, while an "electric panel" is a "volume over vertical area" element.

In the design phase, an element that doesn't meet the placement constraints defined by the *placement type* is notified by the system as a visual warning, showing its bounding-box in semi-transparent blinked red color.

5.2.3 Plugin specific properties

Each plugin has a set of specific properties of the building elements that it represent. Each property is defined by: 1) a name; 2) a type, such as "number", "text", "boolean", o "custom", 3) a value.
According to its type, each property value can be inserted in different ways. For example, a boolean property value is set through a checkbox, while a textual property is set through a text box. The system is designed to accept custom kinds of property. To a custom property it is required to define the component of the UI that permit the user to insert its value. For example, a "color" property can be introduced defining a UI component composed by three text boxes (one for each RGB components), while a "length" property can be introduced defining a UI component composed by a text box for the value and a drop-down menu for the unit of measure.
The specific properties of an element can be edited in the relative panel in the sidebar, once the element is selected in the canvas.

5.3 Plugin Catalog

It is pivotal to provide surveyors with a rich catalog of plugins, to cover all the basic as well as the most advanced modeling requirements. Table 1 reports examples of plugins arranged according to the introduced taxonomy.

	inside	**over / free**
linear	`pipe`	`electrical-conduit`
ver. area	`window, door`	`wall`
hor. area	`light-panel`	`ground, ceil`
volume	`pillar`	`staircase`

Table 1: Plugins example according to taxonomy 5.2.2

5.4 Server-side models generation

Both 3D and 2D model generation has been designed to be asynchronous: the actual result of the invocation of the generation function is not the model itself but rather a *promise* of the expected result. Such a design is important since the computation for model generation may require a while. In the meantime the user must be able to interact with the interface, which in turn must remain responsive. Relying on this architecture, generation of the models can be easily delegated to a server (as shown in Figure 5), thus relieving the client from the burden of onerous computations. The server exposes a REST-like HTTP based JSON API to the client. The plugin span from the client to the server since the 2D and 3D generation functions ("3Dgf" and "2Dgf" respectively in Figure 5) defined by the plugin are actually executed on the server.

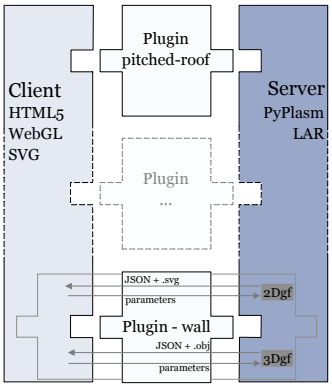

Figure 5: Client/Server architecture for server-side models generation

5.5 A plugin example

POTREBBE ESSERE UTILE INSERE LA DEFINIZIONE DEL PLUGIN SCALA AD ESEMPIO?

6 CONCLUSIONS

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan

nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

REFERENCES

Abramov, D. (2016). Redux: predictable state container for JavaScript apps. `http://redux.js.org/`. Accessed: 2016-11-09.

Altamura, P. (2012). *Gestione eco-efficace dei materiali da costruzione nel ciclo di vita del fabbricato*. PhD thesis, Sapienza UniversitÃă di Roma. in Italian.

Backus, J., Williams, J., Wimmers, E., Lucas, P., and Aiken, A. (1989). FL language manual, parts 1 and 2. Technical report, IBM Research Report.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2014). Julia: A fresh approach to numerical computing.

DiCarlo, A., Paoluzzi, A., and Shapiro, V. (2014). Linear algebraic representation for topological structures. *Comput. Aided Des.*, 46:269–274.

Hurley, J. W. (2002). How to SMARTWaste the Construction Industry. In *10th Symposium Construction Innovation and Global Competitiveness*, Conference Proceedings for the 10th Syposium Construction Innovation and Global Competitiveness.

Paoluzzi, A. (2003). *Geometric Programming for Computer Aided Design*. John Wiley & Sons, Chichester, UK.

Paoluzzi, A., Pascucci, V., and Vicentino, M. (1995). Geometric programming: a programming approach to geometric design. *ACM Trans. Graph.*, 14(3):266–306.