



Delft University of Technology

Essential Means for Urban Computing

Specification of Web-Based Computing Platforms for Urban Planning, a Hitchhiker's Guide

Nourian, Pirouz; Martinez-Ortiz, Carlos; Arroyo Ohori, Ken

DOI

[10.17645/up.v3i1.1299](https://doi.org/10.17645/up.v3i1.1299)

Publication date

2018

Document Version

Final published version

Published in

Urban Planning

Citation (APA)

Nourian, P., Martinez-Ortiz, C., & Arroyo Ohori, K. (2018). Essential Means for Urban Computing: Specification of Web-Based Computing Platforms for Urban Planning, a Hitchhiker's Guide. *Urban Planning*, 3(1), 47-57. <https://doi.org/10.17645/up.v3i1.1299>

Important note

To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.

Article

Essential Means for Urban Computing: Specification of Web-Based Computing Platforms for Urban Planning, a Hitchhiker’s Guide

Pirouz Nourian ^{1,*}, Carlos Martinez-Ortiz ² and Ken Arroyo Ohori ¹

¹ Faculty of Architecture and Built Environment, TU Delft, 2628 BL Delft, The Netherlands; E-Mails: p.nourian@tudelft.nl (P.N.), g.a.k.arroyoohori@tudelft.nl (K.A.O.)

² Netherlands eScience Center, 1098 XG Amsterdam, The Netherlands; E-Mail: c.martinez@esciencecenter.nl

* Corresponding author

Submitted: 30 November 2017 | Accepted: 23 February 2018 | Published: 29 March 2018

Abstract

This article provides an overview of the specifications of web-based computing platforms for urban data analytics and computational urban planning practice. There are currently a variety of tools and platforms that can be used in urban computing practices, including scientific computing languages, interactive web languages, data sharing platforms and still many desktop computing environments, e.g., GIS software applications. We have reviewed a list of technologies considering their potential and applicability in urban planning and urban data analytics. This review is not only based on the technical factors such as capabilities of the programming languages but also the ease of developing and sharing complex data processing workflows. The arena of web-based computing platforms is currently under rapid development and is too volatile to be predictable; therefore, in this article we focus on the specification of the requirements and potentials from an urban planning point of view rather than speculating about the fate of computing platforms or programming languages. The article presents a list of promising computing technologies, a technical specification of the essential data models and operators for geo-spatial data processing, and mathematical models for an ideal urban computing platform.

Keywords

dataflow programming; decision support; planning support; spatial computing; urban computing

Issue

This article is part of the issue “Crowdsourced Data and Social Media for Participatory Urban Planning”, edited by Bernd Resch (University of Salzburg, Austria), Peter Zeile (Karlsruhe Institute of Technology, Germany) and Ourania Kounadi (University of Salzburg, Austria).

© 2018 by the authors; licensee Cogitatio (Lisbon, Portugal). This article is licensed under a Creative Commons Attribution 4.0 International License (CC BY).

1. Introduction

In this article we focus on the applications of urban computing in *Smart Cities Planning* practice (as proposed by (Batty et al., 2012)). They suggest that there is a need for a paradigm-shift in urban planning, from focus on the built environment problems to social problems such as deprivation, and their relations to space, spatial distributions and spatial planning. Considering the complexity of cities, they imply that there is a need to develop “a new science of human [spatial] behaviour”.

This paradigm shift towards developing new [spatial] sciences of cities can be facilitated by the so-called urban computing practices, e.g., by facilitating access to large datasets on human spatial behaviour. This article seeks to illustrate what are the essential means of urban computing practice from a methodological point of view, i.e., computational requirements for 1) developing scientific knowledge in the form of validated analytic/simulation models using spatial data and spatial relations; and 2) informing planning actions using the insight gained from analytic/simulation models on effectiveness of actions.

1.1. What is Urban Computing?

It is difficult, and perhaps even futile, to provide a comprehensive definition of the emerging fields of *Urban Computing* (e.g., as referred to in Kindberg, Chalmers, & Paulos, 2007; Zheng, Capra, Wolfson, & Yang, 2014) and the closely related field of *Urban Informatics* (e.g., as referred to in Foth, Choi, & Satchell, 2011). These two are umbrella terms for describing diverse practices involving geo-spatial data analysis related to cities and citizens. While the former has a technical connotation related to sensing, analysis and actuation technologies (Kindberg et al., 2007), the latter is more focused on the computational social sciences applied to analysis of cities. Without attempting to provide a comprehensive definition, we choose to use the term urban computing with a broader scope to refer to all data-intensive ‘computational workflows’ that can be used for improving urban planning and urban decision-making by providing the means of data acquisition, analysis and simulation, e.g., to reduce traffic congestion or energy consumption. From a technical point of view, urban computing can involve acquisition, integration, and analysis of (big) data generated by diverse sources such as sensing technologies and large-scale computing infrastructures in the context of urban spaces. The volume, velocity and variety of such data often requires the use of cloud computing infrastructure and software services (Hashem et al., 2015). Urban Computing is applicable in a variety of fields, namely:

- environmental studies (e.g., Shang, Zheng, Tong, Chang, & Yu, 2014; Zheng, Liu, & Hsieh, 2013);
- modelling energy use/generation (e.g., Simão, Densham, & Haklay, 2009);
- transport modelling (e.g., Zheng, Liu, Yuan, & Xie, 2011);
- monitoring health (e.g., Varshney, 2007);
- epidemiology (e.g., Lopez, Gunasekaran, Murugan, Kaur, & Abbas, 2015);
- social informatics (e.g., Foth, Forlano, Satchell, Gibbs, & Donath, 2011; Pires & Crooks, 2017);
- criminology (e.g., Bogomolov et al., 2014); and
- participatory planning (e.g., Robinson & Johnson, 2016; Tenney & Sieber, 2016).

1.2. Why Is Urban Computing needed in Urban Planning?

In Urban Planning, we are often interested in analysing the so-called what-if scenarios using simulations and projections (Batty & Torrens, 2001). Traditionally, the geo-spatial analysis of intervention scenarios, urban plans, and urban data is done by means of Geographic Information Systems (GIS), Planning Support Systems (PSS; see Batty, 2007; Harris & Batty, 1993) and Spatial Decision Support Systems (SDSS). The PSS and SDSS systems are typically stand-alone desktop applications that have

a database, a library of computational methods for geo-spatial data processing, and an interface. Despite the technical similarities in using a spatial database, the two categories are different in that the SDSS are geared towards *operational* decision-making whereas the PSS are geared towards *strategic* planning that often involves land-use planning and thus requiring the consideration of land-use transport interactions (the distinction between PSS and SDSS from Geertman & Stillwell, 2009). In these systems, there exist some workflows for spatial analysis of urban data, which do not require new ground-breaking technology. However, the prospect of urban computing is the potentials of the web-based computing platforms for developing a new generation of shareable and editable geo-spatial data processing workflows for informing decisions in urban planning. From urban computing applications listed in Section 1.1, it can be seen that so far urban computing technologies have been mostly applied in the operational and managerial contexts (based on the definition of urban planning actions; Couclelis, 2005). For a wider adoption of urban computing practices in strategic urban planning, urban computing platforms must provide the essential means of analysis and simulation procedures needed in PSS.

Although most of the scholarly works in the area of PSS are focused on land-use change, there are other aspects of urban dynamics that could be modelled computationally; that is to say, the broader discussion is on what changes can be explained, anticipated, and taken into account when making strategic decisions on spatial plans, this broader field of research and development is called Urban Modelling (Batty, 2009). Considering the nature of outcomes of planning processes, (e.g., land-use plans) we can observe that the spatial relations between land-use distributions and a variety of phenomena need to be considered while making strategic planning decisions: for instance, land-use and transport interactions and their effects on energy use in transport (see Keirstead, Jennings, & Sivakumar, 2012) and the effect of land-use distribution on bio-diversity and the use of natural resources (especially water) should ideally be considered when proposing plans. From a pragmatic point of view, however, the adoption of PSS in practice is not high (Geertman & Stillwell, 2009):

It is disturbing, in fact, to observe the extent to which new computer-based support systems are developed by researchers to the point of adoption but are never implemented in planning practice or policy making. Similarly, there is evidence to indicate that systems which are made operational are not extensively used, after the initial novelty has passed, by those planning organizations for which they have been developed in the first instance. In terms of application, it is possible to point to more failures than successes, i.e., to more cases where systems have not been implemented than examples where they are used routinely. Moreover, many state-of-the-art systems appear to

take a long time to reach the ‘market’ and this is often a process requiring considerable financial resources.

We suggest that the research and development culture of Spatial Planning and Decision Support Systems (SPDSS, terminology of Geertman & Stillwell, 2009) must adopt *open-source and agile* development principles for effective ‘market’ uptake and ensuring the viability of the R&D products (Crowston & Howison, 2005; Hey & Payne, 2015; Pressman & Roger, 2009; von Krogh, 2003). By adopting urban computing practices, utilization of scientific knowledge in planning practice will be eased; because web-based computing platforms facilitate rapid prototyping, development, release, sharing, and test of SPDSS (incorporating a variety of Urban [Analysis/Simulation] Models).

1.3. Problem Statement

Although much can be said about the graphical user interfaces of GIS applications, we do not focus on them; because these interfaces are generally geared towards manual operations. Instead our focus is on the essential

means for developing ‘geo-spatial computing workflows’. Workflows can be as simple as routines of sequential actions or more sophisticated procedures with flow-control mechanisms, which are better known as algorithms (see Figures 1 and 2 for workflow examples). There are two types of challenges in using the currently available GIS desktop applications for innovative inter-disciplinary research in Urban Computing applied in Urban Planning (i.e., Design and Development of Web-Based SPDSS):

- Data-Related Challenges:
 - Data-Availability: how easy is it to acquire a relevant dataset?
 - Data-Interoperability: how easy is it to read/write datasets from/to file formats?
 - Data-Mergeability: how easy is it to overlay multiple datasets?
- Workflow-Related Challenges:
 - Workflow Comprehensibility: to what extent is the whole workflow understandable?

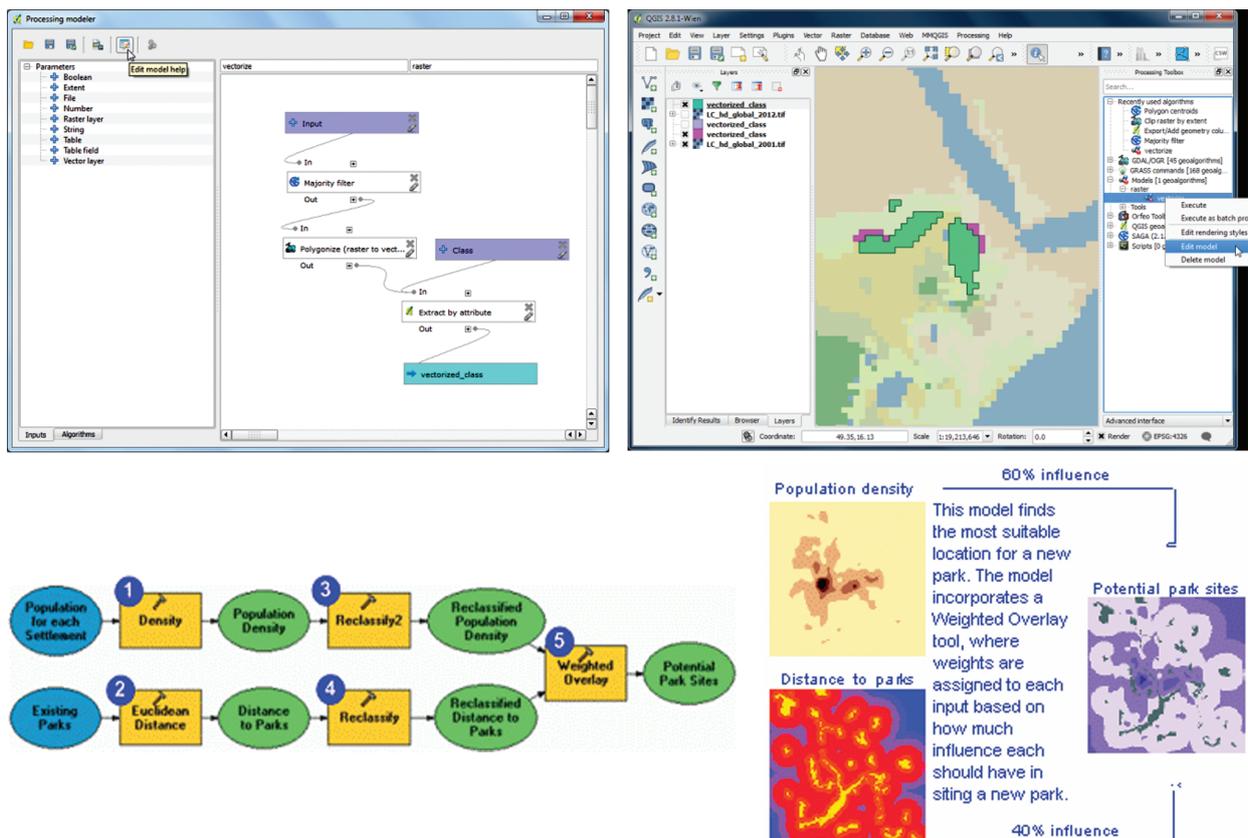


Figure 1. Two examples of geo-spatial data processing workflows from QGIS Processing Modeller¹ (top) and ArcGIS Model Builder² (bottom), respectively made for calculating area of water within 25 metres of urban roads (tutorial), and finding suitable locations for urban parks (tutorial).

¹ http://gracilis.carleton.ca/CUOSGwiki/index.php/Automating_Vector_and_Raster_Workflows_using_the_Graphical_Modeler_in_QGIS#Introductions
² <http://resources.esri.com/help/9.3/ArcGISengine/java/doc/bab90fcc-320b-4b33-902d-a00afd18cfc.htm>

- Workflow Editability: how easy is it to modify the workflow explicitly?
- Workflow Repeatability: how easy is to repeat a certain data processing workflow?
- Workflow Shareability: how easy is it to share a workflow from one system to another?
- Workflow Scalability: how easy is to process large datasets with a workflow?
- Workflow Sustainability: to what extent is the workflow modular and recyclable?

A rather neglected matter about SPSS is the very social/human process of developing them. These systems can be developed by Research Software Engineers.³ A typical research software developer is not necessarily a software engineer, but usually a domain-specific researcher who can develop software or computational workflows. A typical research software engineer, often does not have the means of a software vendor to develop a large application with a custom-made GUI. The core of the work of research software development is on developing analytic workflows.⁴

2. What Do We Need for Urban Computing?

We argue that there are three determining factors to consider with regards to ‘the suitability of a computing technology for urban computing’, i.e., the availability and quality of:

1. Visual Data Flow Programming
2. Spatial Computing Libraries
3. Internet of Things (IoT) APIs⁵

2.1. Visual Dataflow Programming

It is well known that the time spent on research and development is often much more valuable than the computation time. Therefore, we need to consider human interface requirements with regards to the ease of ideation-development-test cycles (prototyping). We propose that using a dataflow programming platform, the user can interact with the platform knowing only a common programming language to edit the nodes (blocks of code) and only a handful of UI manoeuvres to get started; without the problem of learning a sophisticated UI. In processing big data, there are two generic approaches, namely: batch processing and real-time processing (Hashem et al., 2015). Considering the real-time data processing requirement, especially in dealing with managerial and operational planning actions, we can conclude that the Dataflow Programming⁶ is an appropriate paradigm for setting up an R&D/prototyping environment (Blackstock & Lea, 2014; Szydlo, Brzoza-Woch,

Sendorek, Windak, & Gniady, 2017). Considering that the sustainability and the repeatability of the workflow, it is practical to adopt a modularization and standardization approach to workflow development. Standardization is important for reusability. Specifically, the code-blocks (alias nodes, blocks, or subsystems) of a workflow must input and output data in formats readable for one another. Of course, having a visual overview of the workflow is of high added value, as it makes the workflow as intuitive as a flowchart. The idea of a visual dataflow programming language is to represent the high-level logic of a program/workflow as a graph of nodes, which are blocks of (reusable/shareable) code. The representation of the high-level logic as a graph makes it easy to focus on the complex big-picture for a group of developers working on a workflow. Instead of developing a complete software application with a graphical user interface, a research software engineer can focus on the core of the workflow, model the workflow, test it, share it, and release it as a functional prototype.

If the workflow description language is a (de facto) standard, the intended user does not need to learn a new interface to interact with the workflow. In other words, instead of focusing on optimizing a new software application in terms of its interface and the computational efficiency, more attention can be paid to the effectiveness of the workflow itself. In addition, if the workflow is also cloud-based, then it will be easier to share them and collaborate on-line in real-time.

In short, adopting a visual cloud-based dataflow processing language (and ecosystem) brings about a few advantages:

- Automation of repetitive tasks for data cleansing, validation, etc.;
- Informal and yet sustainable standardization based on common-practices and bottom-up emergence of workflow patterns⁷;
- Sharing workflow pattern solutions instead of re-inventing the wheel;
- The possibility of interdisciplinary collaboration;
- Ultimate modularization of workflows based on sharing nodes/blocks of code;
- Agile development-test-release cycles;
- Promotion of Open-Source development practices and therefore rapid progress;
- Ensuring re-usability and repeatability of workflow-based practices such as spatial analyses;
- Saving time by significantly reducing the time and effort in re-inventing interfaces;
- Raising the level of comprehensibility of analytic workflows by providing a glass-box view of the process (as opposed to black-box SPSS); and
- The possibility of public participation in planning

³ <http://rse.ac.uk/who/>

⁴ <http://www.commonwl.org/>

⁵ Application Programming Interfaces.

⁶ <https://stackoverflow.com/questions/461796/dataflow-programming-languages/2035582>

⁷ <http://www.workflowpatterns.com/>

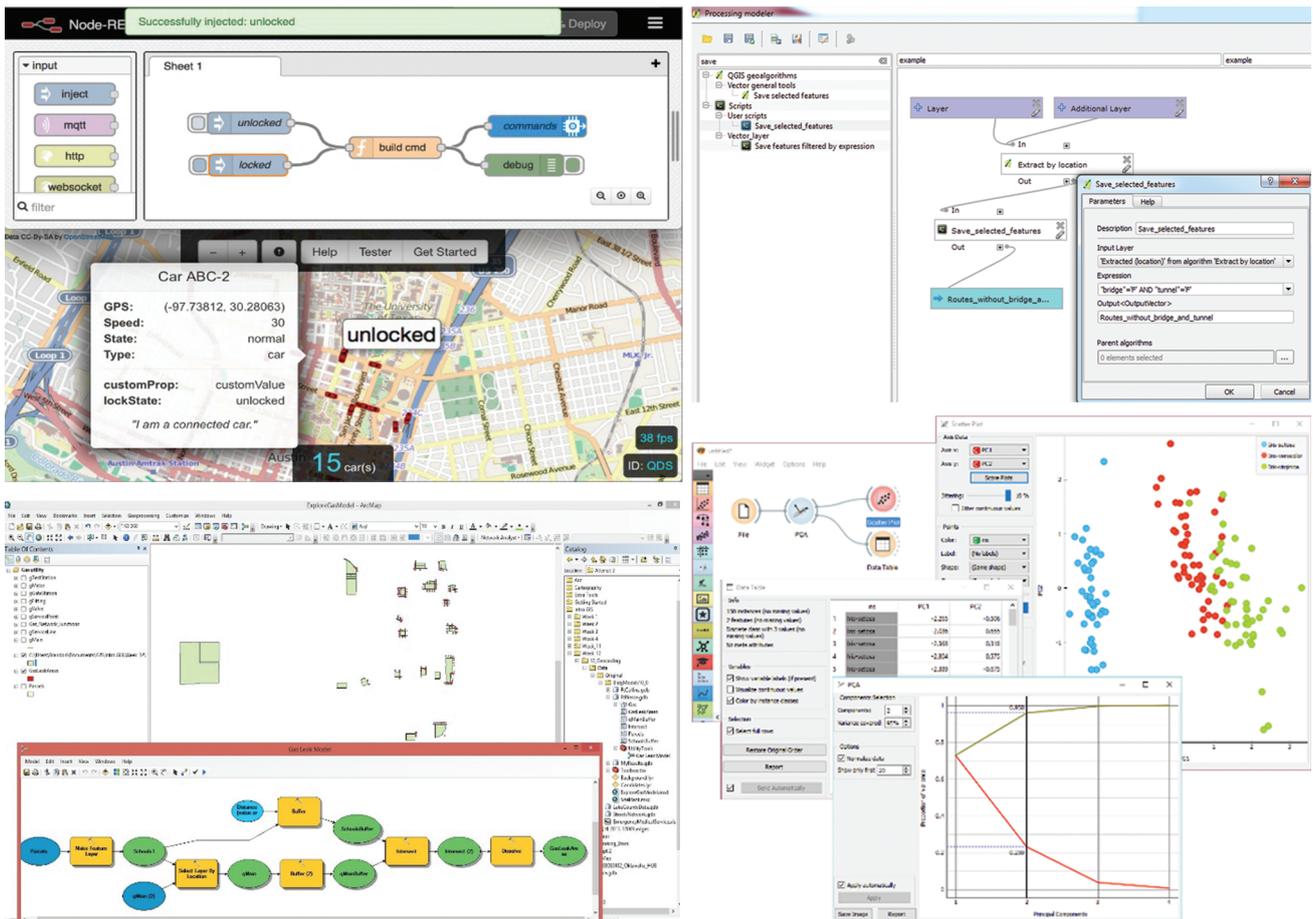


Figure 2. Data processing workflow examples, respectively from top left, clockwise, node-RED, editable by JavaScript (picture from Boyd, 2015), QGIS Graphical Modeler⁸, Anaconda Orange3⁹, and ArcGIS Model Builder¹⁰, all of which offer Python APIs. The GIS dataflow programming environments make it easy to automate routines, share them, and use standard modules; however, the installation procedures, their domain specific nature and their UI make them much less accessible than the two all-purpose data-flow programming environments shown.

processes by means of rapid development and integration of apps (e.g., using Node-RED¹¹, a visual data-flow programming tool for wiring together hardware devices, APIs and online services, see Figure 2).

2.2. Spatial Computing Libraries

Here we provide an overview of the requirements of a software application for urban computing; and focus on the specific functionalities that deal with geo-spatial data. Geo-spatial data can be analysed in at least five spatial forms from the most concrete to the most abstract:

- Geographical Data Models: geographically positioned points, lines, polygons, and polyhedrons;
- Geometrical Data Models: points, lines, polygons, and polyhedrons (in local coordinate systems);

- Topological Data Models: vertices, edges, faces, and bodies (algebraic\combinatorial topology);
- Graphical Data Models: objects and links (Graph Theory); and
- Spectral Data Models: eigenvectors and eigenvalues.

The use of the last category of data models is relatively newer than the other types of the models and is used for modelling the dynamics of diffusion flows and Markov Processes in networks (Nourian, 2016; Nourian, Rezvani, Sariyildiz, & van der Hoeven, 2016; Volchenkov & Blanchard, 2007; Wei & Yao, 2014). Performing spectral analyses requires using a computational linear algebra library such as NumPy¹². Generally, considering the inter-disciplinary nature of urban computing, evident in the breadth and variety of practices mentioned in Section 1.1, we propose that scientific and numerical com-

⁸ https://docs.qgis.org/2.8/en/docs/user_manual/processing/modeler.html?highlight=workflow

⁹ <https://orange.biolab.si/screenshots/>

¹⁰ <http://pro.arcgis.com/en/pro-app/help/analysis/geoprocessing/modelbuilder/what-is-modelbuilder-.htm>

¹¹ <https://nodered.org>

¹² <http://www.numpy.org/>

puting libraries must be available in an ideal platform for urban computing.

In Table 1, we have shown the computational modules required to make spatial analysis and spatial simulation models, which are, in other words, the essential data-models and operations in geo-spatial data processing for urban computing. Central to this schema are the three distinct ways of modelling *space* as:

- **Manifolds**¹³ (often approximated as simplicial complexes);
- **Grids** (a.k.a. 2D/3D raster data models, see Zlatanova, Nourian, Gonçalves, & Vo, 2016);
- **Networks** (a.k.a. [directed/weighted] graphs).

In Figure 3, we have categorized the specifically required functionalities for spatial computing as to the previously introduced fields of application of urban computing. There we have shown an overview of exemplary types of analysis or simulation models for planning support workflows, their typical goals and required data models related to the previously listed areas of applications of urban computing.

2.3. IoT APIs

IoT for smart environment is defined by (Gubbi, Buyya, Marusic, & Palaniswami, 2013) as follows:

Table 1. A list of typical goals, required spatial data types, and analytic (mathematical) or simulation (computational) modelling approaches of urban computing.

	Goal	Typically Required Spatial Data Models	3D?	Exemplary/Potentially Applicable Modelling Methodologies
[Land-Use &] Transport Modelling	understanding potentials (accessibility) and predicting the dynamics of mobility [& land-use change]	road network lines, land-polygons, cellular phone network data, GPS trajectories, etc.	possibly beneficial	Discrete-Choice Modelling, Gravity Models, Agent-Based Modelling (ABM), Cellular Automata (CA), Markov Chains, Operations Research
Sociometrics & Econometrics	understanding potentials, and dynamics of social and economic interactions	demographic data attributed to building, block, district, city, or region polygons, crowd-sourced geo-tagged data points, etc.	probably unnecessary	Markov Chains, Markov Chain Monte Carlo (MCMC), Network Centrality, Artificial Intelligence, Statistical Modelling, Predictive Analytics
Criminology & Crime Prevention	understanding potentials, and dynamics of crime in cities	road-networks, demographics attributed to building, & city polygons, geo-tagged (positioned) spatial crime data, etc.	possibly beneficial	Statistical Modelling, Predictive Analytics, Agent-Based Modelling (ABM), Cellular Automata (CA), Markov Chains, Monte Carlo Simulation
Energy Modelling	understanding potentials, and dynamics of energy use and [renewable] energy generation	3D polyhedral models of buildings, point clouds	necessary	Solar Irradiance Simulation (requiring geometric intersections), Computational Fluid Dynamics (CFD, requiring raster and vector fields and differential operators), Monte Carlo Methods
Environmental Modelling	understanding potentials, and dynamics of environmental threats & opportunities (air pollution, noise, vegetation, etc.)	aerial photos, point clouds, vector maps, raster maps	necessary	Analytic Models and Simulation Models (e.g., CA and ABM), Complex System Dynamics, Hydrology, Complex Adaptive Systems

¹³ <http://mathworld.wolfram.com/Manifold.html>

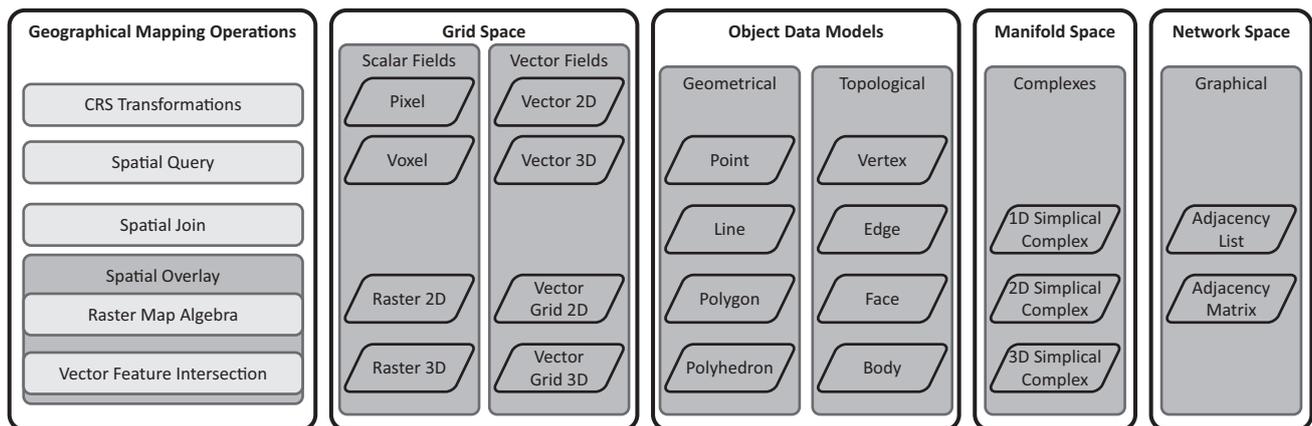


Figure 3. Essential mapping operations and data models required for geo-spatial computing.

Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large scale sensing, data analytics and information representation using cutting edge ubiquitous sensing and cloud computing.

IoT applications can be used for acquisition of data from sensors. They can also be used to directly control some dynamics of cities such as traffic lights. The devices needed for enabling control of physical things are called actuators or actuating devices. The electronic devices that can connect sensors and actuators to internet could be micro-controllers or micro-computers, some of which are open devices popular among amateur enthusiasts such as Arduino¹⁴ and Raspberry Pi¹⁵. The capabilities of a computing technology for interacting with such devices can be a key factor in making it more pervasive among enthusiast makers and academic software developers, due to the accessibility of such devices in terms of low prices and ease of learning.

Operational planning actions can especially benefit from actuators and sensors in urban environments. For instance, traffic lights can be actuated (controlled) by a controller system connected to many of both sensors and actuators in real-times (thus having a real-time overview of a city) continuously analysing the data coming from sensors sensing the volume of traffic. In other words, IoT devices can facilitate (real-time) operational planning actions. With regards to the IoT potentials for Urban Computing, it is logical to assume that Web-based GIS services (alias web mapping) are necessary for urban computing. In addition, moving all workflows from desktop applications to web-based platforms makes it eas-

ier to share (standardized) workflows and collaborate on them. In the next section we focus on the potentials of four programming languages for setting up web-based computational workflows for geo-spatial data analytics and simulations.

3. Promising Technologies for Urban Computing

We have identified a few promising technologies for urban computing, based on Python, Java, JavaScript and R-Spatial languages. From a practical perspective, we consider their potential in terms of ease of prototyping, geo-spatial mapping, 3D visualization, handling big data, and numerical computing (computational linear algebra). From a mathematical/computational point of view, all required models mentioned in Figure 3 can be rather easily developed on top of a robust computational linear algebra library. Apart from numerical capabilities, we argue that for a research software engineer, the visualization and mapping capabilities are essential to consider while making technical choices.

3.1. Python

This programming language is used for example in the Geoda-Web¹⁶, that is the web-based version of CAST¹⁷ with its spatial analysis library PySal¹⁸ seems to be a promising open-source project. Python is the de facto language of open-source development in the field of Geo information science, e.g., in QGIS, Rasterio¹⁹ and Fiona²⁰. Python provides a wide range of libraries for numerical and scientific computing such as NumPy, SciPy and Pandas, which facilitates development. Interactive development environments such as IPython (Interactive Python) (Perez & Granger, 2007) and web-based Jupyter notebooks (Shen, 2014) seems to be a promising technology

¹⁴ <https://www.arduino.cc/>

¹⁵ <https://www.raspberrypi.org>

¹⁶ <http://spatial.uchicago.edu/geoda-web>

¹⁷ <https://geodacenter.github.io/CAST/>

¹⁸ <http://pysal.readthedocs.io/en/latest/users/tutorials/dynamics.html>

¹⁹ <https://github.com/mapbox/rasterio>

²⁰ <https://github.com/Toblerity/Fiona>

for prototyping and interactive computing. Some universities have started facilitating the use of Jupyter interactive documents as a common means of exchanging reproducible research products, e.g., on JupyterHub²¹, NBViewer²², or SURF-sara (Templon & Bot, 2016) provide hosting and viewing services for sharing Jupyter notebooks. A few options which stand out for simple 3D visualization in Python are: Matplotlib²³, Mayavi²⁴ or VisPy²⁵, while more high-performance applications can be built in OpenGL using PyOpenGL²⁶. Web mapping in Python is possible by means of GeoDjango²⁷.

3.2. Java

This programming language is used for example in a web-GIS for environmental analyses by (Zavala-Romero et al., 2014). The FIWARE platform (Zahariadis et al., 2014) offers an “Application MashUp Generic Enabler”, i.e., the WireCloud²⁸ for visual programming and prototyping web applications. Another flow-based programming environment for Java development supported by Apache Hadoop²⁹ is NiFi³⁰. Java can also provide for interactivity and 3D visualization. The OpenGeoSpatial foundation (aka OSGeo³¹) also provides an open source GIS toolkit for Java called GeoTools³². Considering the might of Hadoop for big data analytics and the support of OSGeo Java seems to be a fertile language for urban computing. One option for 3D visualization in Java is JogAmp³³, while a more advanced option is JOGL³⁴.

3.3. JavaScript

This programming language is used for example in OpenLayers³⁵ and Carto³⁶ SaaS (Software as a Service, for-

merly known as CartoDB³⁷) to provide user-friendly Web-GIS tools, which can moreover be deployed as desktop applications with tools like Electron³⁸. However, neither of them supports explicit workflow development. The other promising JavaScript platform for spatial analysis is MapBox³⁹, which offers access to the Turf library⁴⁰. Node-RED (Blackstock & Lea, 2014), based on IBM BlueMix (a.k.a. IBM Cloud)⁴¹, seems to be a promising technology in terms of visual programming and the ease of prototyping IoT applications. Node-RED is distributed as part of an open-source software ecosystem called node package manager or NPM⁴², that is managed by the Node.js⁴³ foundation. Interactive visualization in web-browsers is well supported in JavaScript, and arguably more advanced than comparable libraries in Python, thanks to the D3.js library, by Mike Bostock⁴⁴ (Bostock, Ogievetsky, & Heer, 2011). In addition to D3 for interactive graphics, there is three.js⁴⁵ for WebGL rendering in the browser. Other JavaScript libraries which should not go unnoticed for urban computing are Leaflet⁴⁶ (mobile-friendly interactive maps providing access to OSM⁴⁷) and Cesium⁴⁸, the latter providing for quality 3D visualization.

3.4. R Spatial

R is a programming language that is part of the R Project for Statistical Computing⁴⁹, which includes a complete set of vector algebra operations and functions to create graphics such as plots. The statistical functions in R are much more complete than those available in other languages (e.g., Python). The R Spatial⁵⁰ functionality includes the more relevant parts for urban computing, such as representations for raster and vector data, deal-

²¹ <https://github.com/jupyterhub>

²² <https://nbviewer.jupyter.org>

²³ <https://matplotlib.org/index.html>

²⁴ <http://docs.enthought.com/mayavi/mayavi>

²⁵ <http://vispy.org/index.html>

²⁶ <http://pyopengl.sourceforge.net>

²⁷ <https://docs.djangoproject.com/en/dev/ref/contrib/gis/>

²⁸ <https://catalogue.fiware.org/enablers/application-mashup-wirecloud>

²⁹ <http://hadoop.apache.org>

³⁰ <https://hortonworks.com/apache/nifi>

³¹ <http://www.osgeo.org>

³² <http://www.geotools.org>

³³ <http://jogamp.org>

³⁴ <http://jogamp.org>

³⁵ <http://openlayers.org>

³⁶ <https://carto.com/blog/how-to-use-spatial-analysis-in-your-site-planning-process>

³⁷ <https://cartodb.github.io/training/intermediate/columbia-sipa.html>

³⁸ <https://electronjs.org>

³⁹ <https://www.mapbox.com/help/how-analysis-works>

⁴⁰ <http://turfjs.org>

⁴¹ <https://www.ibm.com/cloud>

⁴² <https://www.npmjs.com>

⁴³ <https://nodejs.org/en>

⁴⁴ <https://bl.ocks.org/mbostock>

⁴⁵ <https://threejs.org>

⁴⁶ <http://leafletjs.com>

⁴⁷ <http://www.openstreetmap.org>

⁴⁸ <https://cesiumjs.org>

⁴⁹ <https://www.r-project.org>

⁵⁰ <http://www.rspatial.org>

ing with coordinate systems and creating 2D maps. Spatial.ly⁵¹ shows several examples of the more advanced visualisation functions in R, including 3D visualisation and animated globes. Shiny⁵² is a tool to build web apps with R. There are also other ways in which web sessions of R can be deployed, such as with Rweb⁵³ and rApache⁵⁴. Similar to Python, Jupyter notebooks can also be used thanks to the IRkernel⁵⁵.

4. Conclusion

In response to this question: “What are the essential means for urban computing?”, we have provided an overview of specific data models and functionalities required in dealing with geo-spatial data processing (spatial analysis and spatial simulation), referred to as spatial computing in Figure 3 and Table 1, which we deem as the essential means for urban computing. We have considered four programming languages and their promising aspects for urban computing. They all come with their own advantages and shortcomings. It is difficult (and perhaps futile) to point to one of these languages as the most promising language for urban computing. We stress that these technologies are not mutually exclusive, but they can (in some cases) be used in combination with each other. For example, a web-based GIS system could use a Python backend with Flask⁵⁶ and a JavaScript frontend with a 3D visualiser based on Cesium, or a processing pipeline could use Python to fetch data from the web using a tool like BeautifulSoup⁵⁷, use Java to parse and process the data, use R to do statistical analysis on it, and then visualize the results in a browser using JavaScript. However, it can be said that each of them is stronger in a certain direction, respectively: Java in server-side tools, R Spatial in statistical and mathematical operations, Python in the availability of GIS tools, and JavaScript in IoT and web visualisation. Their respective strengths can be combined by using the best language for each task.

In addition, it is perhaps noteworthy to mention that in the related field of computer-aided design (CAD), there is an active movement towards development of visual programming languages and connecting them together by means of a cloud platform, e.g., Flux⁵⁸, initially sponsored by Google⁵⁹. Considering the attractiveness of aligning urban design and urban planning actions, it would be ideal to work in an environment where planners, designers, and research software engineers could

all work and share their workflows, for example, a 3D city modelling SaaS such as Möbius⁶⁰ (Janssen, Li, & Mohanty, 2016), Tygron⁶¹ or CityZenith⁶² could potentially become such a shared development environment.

Acknowledgements

We thank Dr. George Jennings for kindly performing the final proofreading of this article. We are grateful for the meticulous comments of the anonymous reviewers. Ken Arroyo Ohori has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 677312 UMnD).

Conflict of Interests

The authors declare no conflict of interests.

References

- Batty, M. (2007). *Planning support systems: progress, predictions, and speculations on the shape of things to come* (UCL Working Papers Series).
- Batty, M. (2009). Urban Modelling: Chapter XXX. *International Encyclopedia of Human Geography*. Retrieved from www.elsevierdirect.com/brochures/hugy/overview.html
- Batty, M., Axhausen, K. W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., . . . & Portugali, Y. (2012). Smart cities of the future. *The European Physical Journal Special Topics*, 214(1), 481–518. <http://doi.org/10.1140/epjst/e2012-01703-3>
- Batty, M., & Torrens, P. M. (2001). Modeling complexity: The limits to prediction. *Cybergeo: European Journal of Geography*. <http://doi.org/10.4000/cybergeo.1035>
- Blackstock, M., & Lea, R. (2014). Toward a distributed data flow platform for the Web of Things (Distributed Node-RED). *Proceedings of the 5th International Workshop on Web of Things* (pp. 34–39). <http://doi.org/10.1145/2684432.2684439>
- Bogomolov, A., Lepri, B., Staiano, J., Oliver, N., Pianesi, F., & Pentland, A. (2014). Once upon a crime: Towards crime prediction from demographics and mobile data. In *ICMI’14* (pp. 427–434). Istanbul: ACM. <http://doi.org/10.1145/2663204.2663254>
- Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3 data-

⁵¹ <http://spatial.ly/r>

⁵² <https://shiny.rstudio.com>

⁵³ <http://pbil.univ-lyon1.fr/Rweb>

⁵⁴ <http://rapache.net>

⁵⁵ <https://irkernel.github.io>

⁵⁶ <http://flask.pocoo.org>

⁵⁷ <https://www.crummy.com/software/BeautifulSoup>

⁵⁸ <https://flux.io/>

⁵⁹ <https://bimandintegrateddesign.com/2014/10/24/googles-bim-busting-app-for-design-and-construction>

⁶⁰ <https://phtj.github.io/mobius>

⁶¹ <http://www.tygron.com/>

⁶² <http://www.cityzenith.com/smartworld>

- driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12), 2301–2309. <http://doi.org/10.1109/TVCG.2011.185>
- Boyd, B. (2015). Build a connected-car IoT app with Geospatial Analytics. Retrieved from <https://www.ibm.com/developerworks/library/mo-connectedcar-app/index.html>
- Couclelis, H. (2005). “Where has the future gone?” Rethinking the role of integrated land-use models in spatial planning. *Environment and Planning A*, 37(8), 1353–1371. <http://doi.org/10.1068/a3785>
- Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, 2(SPEC). <http://doi.org/10.5210/fm.v10i2.1207>
- Foth, M., Choi, J. H., & Satchell, C. (2011). Urban informatics. *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work—CSCW ‘11*, 1. <http://doi.org/10.1145/1958824.1958826>
- Foth, M., Forlano, L., Satchell, C., Gibbs, M., & Donath, J. (2011). *From social butterfly to engaged citizen: urban informatics, social media, ubiquitous computing, and mobile technology to support citizen engagement*. MIT Press.
- Geertman, S., & Stillwell, J. C. H. (2009). *Planning support systems best practice and new methods*. Berlin: Springer.
- Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660.
- Harris, B., & Batty, M. (1993). Locational models, geographic information and planning support systems. *Journal of Planning Education and Research*, 12(3), 184–198. <http://doi.org/10.1177/0739456X9301200302>
- Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Ullah Khan, S. (2015). The rise of “big data” on cloud computing: Review and open research issues. *Information Systems*, 47, 98–115. <http://doi.org/10.1016/j.is.2014.07.006>
- Hey, T., & Payne, M. C. (2015). Open science decoded. *Nature Physics*, 11, 367–369. <http://doi.org/10.1038/nphys3313>
- Janssen, P., Li, R., & Mohanty, A. (2016). MÖBIUS, A parametric modeller for the web. In S. Chien, S. Choo, M. A. Schnabel, W. Nakapan, M. J. Kim, & S. Roudavski (Eds.), *Living systems and micro-utopias: Towards continuous designing. Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2016* (pp. 157–166).
- Keirstead, J., Jennings, M., & Sivakumar, A. (2012). A review of urban energy system models: Approaches, challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 16(6), 3847–3866. <http://doi.org/10.1016/j.rser.2012.02.047>
- Kindberg, T., Chalmers, M., & Paulos, E. (2007). Guest editors’ introduction: Urban computing. *IEEE Pervasive Computing*, 6(3), 18–20. <https://doi.org/10.1109/MPRV.2007.57>
- Lopez, D., Gunasekaran, M., Murugan, B. S., Kaur, H., & Abbas, K. M. (2015). Spatial big data analytics of influenza epidemic in Vellore, India. In *Proceedings of the 2014 IEEE International Conference on Big Data, IEEE Big Data 2014* (pp. 19–24). IEEE. <http://doi.org/10.1109/BigData.2014.7004422>
- Nourian, P. (2016). Configraphics: Graph theoretical methods for design and analysis of spatial configurations. *A+ BE| Architecture and the Built Environment*, 6(14), 1–348. <http://doi.org/10.7480/abe.2016.14.1348>
- Nourian, P., Rezvani, S., Sariyildiz, I. S., & van der Hoeven, F. D. (2016). Spectral modelling for spatial network analysis. In *Proceedings of the Symposium on Simulation for Architecture and Urban Design (simAUD 2016)*. Retrieved from <https://repository.tudelft.nl/islandora/object/uuid:81c02b9c-3ddc-4273-8c2b-7e84c6dc7604>
- Perez, F., & Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science and Engineering*, 9(3), 21–29. <http://doi.org/10.1109/MCSE.2007.53>
- Pires, B., & Crooks, A. T. (2017). Modeling the emergence of riots: A geosimulation approach. *Computers, Environment and Urban Systems*, 61, 66–80. <http://doi.org/10.1016/j.compenvurbsys.2016.09.003>
- Pressman, & Roger, S., (2009). *The manifesto for agile software development*, 7(2). Retrieved from <http://nlp.chonbuk.ac.kr/SE/ch05.pdf>
- Robinson, P. J., & Johnson, P. A. (2016). Civic hackathons: New terrain for local government-citizen interaction? *Urban Planning*, 1(2), 65–74. <http://doi.org/10.17645/up.v1i2.627>
- Shang, J., Zheng, Y., Tong, W., Chang, E., & Yu, Y. (2014). Inferring gas consumption and pollution emission of vehicles throughout a city. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: KDD ‘14* (pp. 1027–1036). <http://doi.org/10.1145/2623330.2623653>
- Shen, H. (2014). Interactive notebooks: Sharing the code. *Nature*, 515(7525), 151–152. <http://doi.org/10.1038/515151a>
- Simão, A., Densham, P. J., & Haklay, M. (2009). Web-based GIS for collaborative planning and public participation: An application to the strategic planning of wind farm sites. *Journal of Environmental Management*, 90(6), 2027–2040. <http://doi.org/10.1016/j.jenvman.2007.08.032>
- Szydło, T., Brzoza-Woch, R., Sendorek, J., Windak, M., & Gniady, C. (2017). Flow-based programming for IoT leveraging fog computing. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 74–79). IEEE. <http://doi.org/10.1109/WETICE.2017.17>

- Templon, J., & Bot, J. (2016). *The Dutch national e-infrastructure*, 13–18. Retrieved from <http://pos.sissa.it>
- Tenney, M., & Sieber, R. (2016). Data-driven participation: Algorithms, cities, citizens, and corporate control. *Urban Planning*, 1(2), 101–113.
- Varshney, U. (2007). Pervasive healthcare and wireless health monitoring. *Mobile Networks and Applications*, 12(2/3), 113–127. <http://doi.org/10.1007/s11036-007-0017-1>
- Volchenkov, D., & Blanchard, P. (2007). Random walks along the streets and canals in compact cities: Spectral analysis, dynamical modularity, information, and statistical mechanics. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 75(2). <http://doi.org/10.1103/PhysRevE.75.026104>
- von Krogh, G. (2003). Open-source software. *MIT Sloan Management Review*, 44(3), 14–18.
- Wei, X., & Yao, X. A. (2014). The random walk value for ranking spatial characteristics in road networks. *Geographical Analysis*, 46(4), 411–434. <http://doi.org/10.1111/gean.12064>
- Zahariadis, T., Papadakis, A., Alvarez, F., Gonzalez, J., Lopez, F., Facca, F., & Al-Hazmi, Y. (2014). FIWARE Lab: Managing resources and services in a cloud federation supporting future internet applications. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing* (pp. 792–799). IEEE. <http://doi.org/10.1109/UCC.2014.129>
- Zavala-Romero, O., Ahmed, A., Chassignet, E. P., Zavala-Hidalgo, J., Fernández Eguiarte, A., & Meyer-Baese, A. (2014). An open source Java web application to build self-contained web GIS sites. *Environmental Modelling and Software*, 62, 210–220. <http://doi.org/10.1016/j.envsoft.2014.08.029>
- Zheng, Y., Capra, L., Wolfson, O., & Yang, H. (2014). Urban computing. In *ACM Transactions on Intelligent Systems and Technology*, 5(3), 1–55. <http://doi.org/10.1145/2629592>
- Zheng, Y., Liu, F., & Hsieh, H.-P. (2013). U-Air. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: KDD '13* (p. 1436). <http://doi.org/10.1145/2487575.2488188>
- Zheng, Y., Liu, Y., Yuan, J., & Xie, X. (2011). Urban computing with taxicabs. In *Proceedings of the 13th International Conference on Ubiquitous Computing: UbiComp '11* (p. 89). <http://doi.org/10.1145/2030112.2030126>
- Zlatanova, S., Nourian, P., Gonçalves, R., & Vo, A. V. (2016). Towards 3D raster GIS: On developing a raster engine for spatial DBMS. In *ISPRS WG IV/2 Workshop: Global Geospatial Information and High Resolution Global Land Cover/Land Use Mapping*.

About the Authors



Pirouz Nourian (PhD) is an Assistant Professor at the chair of Design Informatics at TU Delft. Pirouz has a PhD in Design Informatics, an MSc in Architecture, and a BSc in Electrical Engineering/Control Engineering. He develops design and research methods and software in the fields of architecture and urbanism. He teaches computational design and procedural 3D modelling in MSc architecture and MSc geomatics at TU Delft. His main areas of research are spatial computation and spatial configuration.



Carlos Martinez-Ortiz (PhD) is an eScience Research Engineer at the Netherlands eScience Center, Amsterdam. Carlos obtained his BSc in computer engineering at La Salle University in Mexico City. At the University of Exeter, he obtained his MSc in applied artificial intelligence and PhD on the topic of shape descriptors for image classification. Carlos has worked on various research projects at the University of Exeter and Plymouth University in collaboration with industrial partners such as C3 Resources Ltd and Met Office.



Ken Arroyo Othori (PhD) is a Postdoc Researcher at the 3D geoinformation group of the Delft University of Technology in the Netherlands. Ken has a BSc in Computer Science and Technology, an MSc in Geomatics, and a PhD in 3D geoinformation. His current research involves higher-dimensional (4D and higher) data models, data structures and algorithms for Geographic Information Systems. Other research interests include geometric modelling, the validation and repair of geographic data and visualising spatial data.