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

Pirouz Nourian Carlos Martinez-Ortiz
Ken Arroyo Ohori

This is an author's version of the paper. The authoritative version is:

Essential Means for Urban Computing: Specification of Web-Based Computing Platforms for Urban Planning, a Hitchhiker's Guide. Pirouz Nourian, Carlos Martinez-Ortiz and Ken Arroyo Ohori. *Urban Planning* 3(1), March 2018, pp. 47–57. ISSN: 2183–7635.
DOI: 10.17645/up.v3i1.1299

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.

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 et al. [2007]; Zheng et al. [2014]) and the closely related field of Urban Informatics (e.g., as referred to in Foth et al. [2011b]). 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 et al. [2014]; Zheng et al. [2013]);
- modelling energy use/generation (e.g., Simão et al. [2009]);
- transport modelling (e.g., Zheng et al. [2011]);
- monitoring health (e.g., Varshney [2007]);
- epidemiology (e.g., Lopez et al. [2015]);
- social informatics (e.g., Foth et al. [2011a]; Pires and Crooks [2017]);
- criminology (e.g., Bogomolov et al. [2014]); and
- participatory planning (e.g., Robinson and Johnson [2016]; Tenney and 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 and 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 and 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 and 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 et al. [2012]) and the effect of land-use distribution on biodiversity 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 and 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 and Stillwell [2009]) must adopt open-source and agile development principles for effective 'market' uptake and ensuring the viability of the R&D products [Crowston and Howison, 2005; Hey and Payne, 2015; Pressman, 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 'geospatial 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?

  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 SPDSS is the very social/human process of developing them. These systems can be developed by Research Software Engineers[1]. 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[2].

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) APIs[3]

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:

[1] http://rse.ac.uk/who/

[2] http://www.commonwl.org/

[3] Application Programming Interfaces

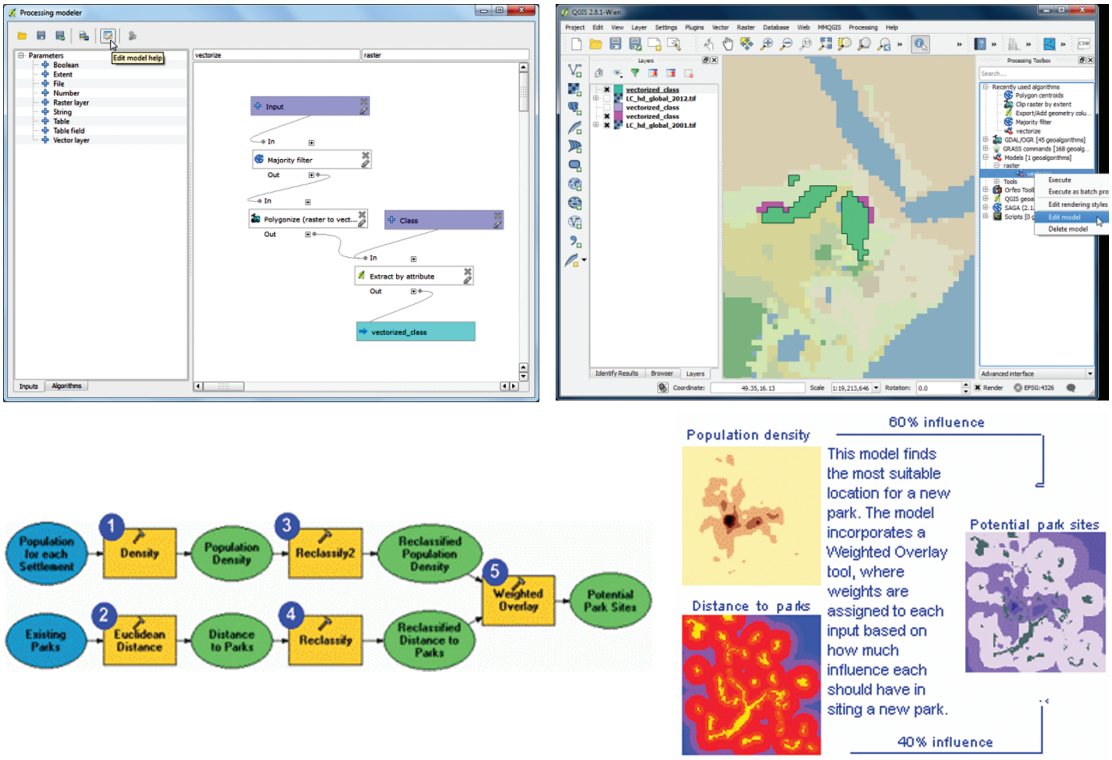

Figure 1: Two examples of geo-spatial data processing workflows from QGIS Processing Modeller[4] (top) and ArcGIS Model Builder[5] (bottom), respectively made for calculating area of water within 25 metres of urban roads (tutorial), and finding suitable locations for urban parks (tutorial).

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 Programming[6] is an appropriate paradigm for setting up an R&D/prototyping environment [Blackstock and Lea, 2014; Szydlo et al., 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 inter-

[4] http://gracilis.carleton.ca/CUOSGwiki/index.php/Automating_Vector_and_Raster_Workflows_using_the_Graphical_Modeler_in_QGIS#Introduction

[5] http://resources.esri.com/help/9.3/ArcGISengine/java/doc/bab90fcc-320b-4b33-902d-a00afd18cfcb.htm

[6] https://stackoverflow.com/questions/461796/dataflow-programming-languages/2035582

act 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 patterns[7];
- 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 SPDSS); and
- The possibility of public participation in planning processes by means of rapid development and integration of apps (e.g. using Node-RED[8], a visual data-flow programming tool for wiring together hardware devices, APIs and online services, see Figure 2).

[7] http://www.workflowpatterns.com/

[8] https://nodered.org/

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)

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 et al., 2016; Volchenkov and Blanchard, 2007; Wei and Yao, 2014]. Performing spectral analyses requires using a computational linear algebra library such as NumPy[9]. Generally, considering the interdisciplinary nature of urban computing, evident in the breadth and variety of practices mentioned in Section 1.1, we propose that scientific and numerical computing libraries must be available in an ideal platform for urban computing.

In Figure 3, 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

[9] http://www.numpy.org/

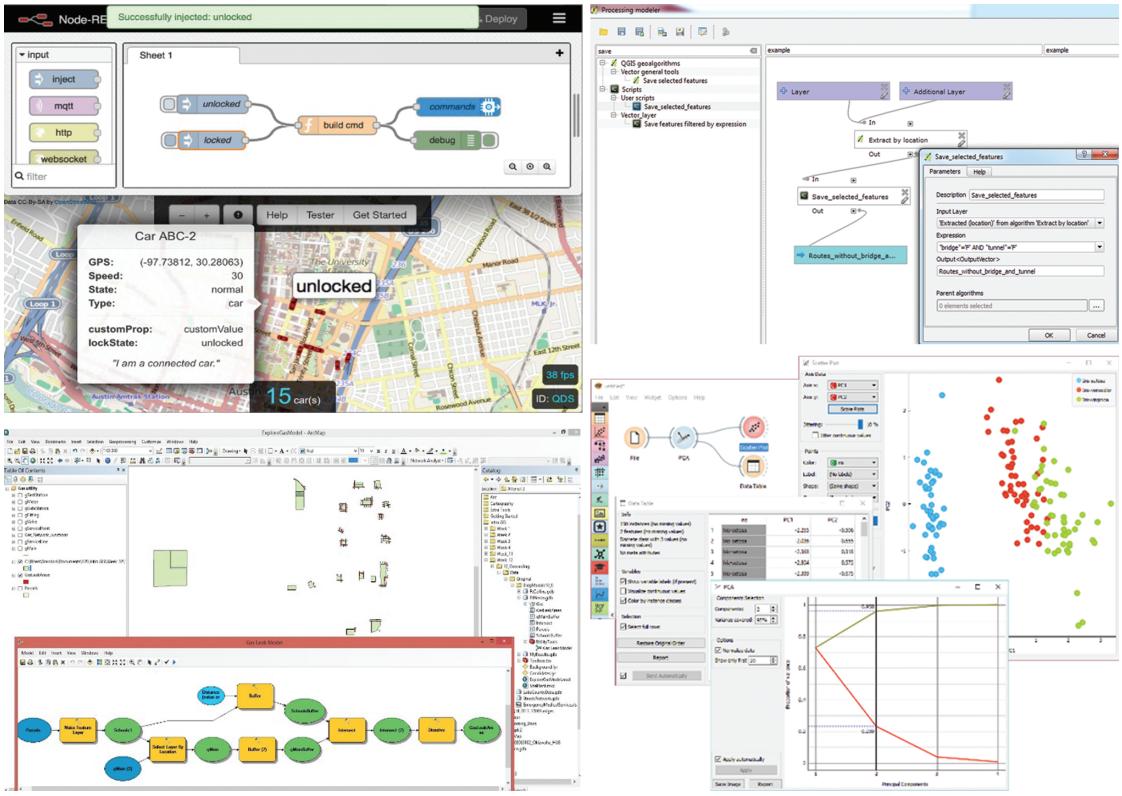

Figure 2: Data processing workflow examples, respectively from top left, clockwise, node-RED, editable by JavaScript (picture from [Boyd, 2015]), QGIS Graphical Modeller[10], Anaconda Orange3[11], and ArcGIS Model Builder[12], all of which offer Python APIs. The GIS dataflow programming environments 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.

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 et al. [2016])

Networks (a.k.a. [directed/weighted] graphs)

In Table 1, 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 Internet of Things APIs

Internet of Things (IoT) for smart environment is defined by Gubbi et al. [2013] as follows:

"Interconnection of sensing and ac-

[10] https://docs.qgis.org/2.8/en/docs/user_manual/processing/modeler.html?highlight=workflow
[11] https://orange.biolab.si/screenshots/
[12] http://pro.arcgis.com/en/pro-app/help/analysis/geoprocessing/modelbuilder/what-is-modelbuilder-.htm
[13] http://mathworld.wolfram.com/Manifold.html

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, crowdsourced 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

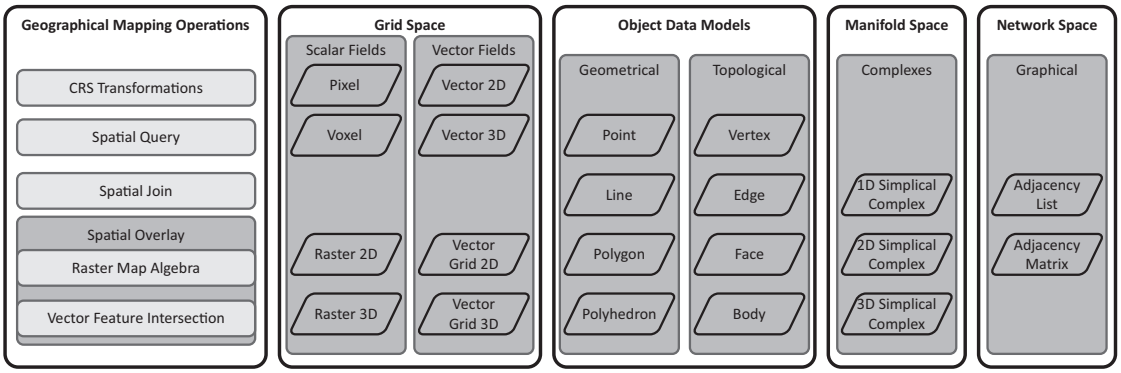

Figure 3: Essential mapping operations and data models required for geo-spatial computing. In Table 1, we have categorized the specifically required functionalities for spatial computing as to the previously introduced fields of application of urban computing. There we have shown an overview of exemplary 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.

> *tuating 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.*"

Internet of Things (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 Arduino[14] and Raspberry Pi[15]. 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.

[14] https://www.arduino.cc/
[15] https://www.raspberrypi.org/

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 easier to share (standardized) workflows and collaborate on them. In the next section we focus on the potentials of three 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 and JavaScript 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.

Python This programming language is used for example in the Geoda-Web[16], that is the web-based version of CAST[17] with its spatial analysis library PySal[18] 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, Rasterio[19] and Fiona[20]. 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 and Granger, 2007] and web-based Jupyter notebooks [Shen, 2014] seems to be a promising technology 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 JupyterHub[21], NBViewer[22], or SURF-sara [Templon and Bot, 2016] provide hosting and viewing services for sharing Jupyter notebooks. A few options which stand out for simple 3D visualization in Python are: MatPlotLib[23], Mayavi[24] or VisPy[25], while more high-performance applications can be built in OpenGL using PyOpenGL[26]. Web mapping in Python is possible by means of GeoDjango[27].

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 WireCloud[28] for visual programming and prototyping web applications. Another flow-based programming environment for Java development supported by Apache Hadoop[29] is NiFi[30]. Java can also provide for interactivity and 3D visualization. The OpenGeoSpatial foundation (aka OSGeo[31]) also provides an open source GIS toolkit for Java called GeoTools[32]. 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 JogAmp[33], while a more advanced option is JOGL[34].

JavaScript : This programming language is used for example in OpenLayers[35] and Carto[36] SaaS (Software as a Service, formerly known as CartoDB[37]) to provide user-friendly Web-GIS tools, which can moreover be deployed as desktop applications with tools like Electron[38]. However, neither of them supports explicit workflow development. The

[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
[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/jogl/www/
[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

other promising JavaScript platform for spatial analysis is MapBox[39], which offers access to the Turf library[40]. Node-RED [Blackstock and 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 Internet of Things (IoT) applications. Node-RED is distributed as part of an open-source software ecosystem called node package manager or NPM[42], that is managed by the Node.js[43] 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 Bostock[44] [Bostock et al., 2011]. In addition to D3 for interactive graphics, there is three.js[45] for WebGL rendering in the browser. Other JavaScript libraries which should not go unnoticed for urban computing are Leaflet[46] (mobile-friendly interactive maps providing access to OSM[47]) and Cesium[48], the latter providing for quality 3D visualization.

R Spatial : R is a programming language that is part of the R Project for Statistical Computing[49], 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 Spatial[50] functionality includes the more relevant parts for urban computing, such as representations for raster and vector data, dealing with coordinate systems and creating 2D maps. Spatial.ly[51] shows several examples of the more advances visualisation functions in R, including 3D visualisation and animated globes. Shiny[52] 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 Rweb[53] and rApache[54]. Similar to Python, Jupyter notebooks can also be used thanks to the IRkernel[55].

[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
[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

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 three 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 Flask[56] 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 BeautifulSoup[57], 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

[56] http://flask.pocoo.org
[57] https://www.crummy.com/software/BeautifulSoup/

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. Flux[58], initially sponsored by Google[59]. 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 Möbius[60] [Janssen et al., 2016], Tygron[61] or CityZenith[62] could potentially become such a shared development environment.

Acknowledgements

We thank Dr. George Jennings for kindly performing the final proofreading of this paper. 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).

References

M. Batty. *Urban Modeling*, pages 51–58. Elsevier, 2009.

M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali. Smart cities of the future. *The European Physical Journal Special Topics*, 214(1):481–518, 2012.

Michael Batty. Planning support systems: progress, predictions, and speculations on the shape of things to come. In *Planning Support Systems for Urban and Regional Analysis*, 2007.

Michael Batty and Paul M. Torrens. Modelling complexity : The limits to prediction. *Cybergeo: European Journal of Geography*, 2001.

Michael Blackstock and Rodger Lea. Toward a distributed data flow platform for the web of things (distributed node-red). In *WoT '14 Proceedings of the 5th International Workshop on Web of Things*, pages 34–39. ACM, 2014.

Andrey Bogomolov, Bruno Lepri, Jacopo Staiano, Nuria Oliver, Fabio Pianesi, and Alex Pentland. Once upon a crime: Towards crime prediction from demographics and mobile data. In *ICMI '14 Proceedings of the 16th International Conference on Multimodal Interaction*, pages 427–434. ACM, 2014.

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

Bryan C. Boyd. Build a connected-car iot app with geospatial analytics. Available at `https://www.ibm.com/developerworks/library/mo-connectedcar-app/index.html`, 2015.

Helen Couclelis. "where has the future gone?" rethinking the role of integrated land-use models in spatial planning. *Environment and Planning A: Economy and Space*, 37(8):1353–1371, 2005.

Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, 10 (2), 2005.

Marcus Foth, Laura Forlano, Christine Satchell, and Martin Gibbs. *From Social Butterfly to Engaged Citizen: Urban Informatics, Social Media, Ubiquitous Computing, and Mobile Technology to Support Citizen Engagement*. MIT Press, 2011a.

[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`

Marcus Foth, Jaz Hee jeong Choi, and Christine Satchell. Urban informatics. In *CSCW '11 Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 1–8. ACM, 2011b.

Stan Geertman and John Stillwell. *Planning support systems best practice and new methods*, volume 95 of *GeoJournal Library*. Springer Netherlands, 2009.

J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

Britton Harris and Michael Batty. Locational models, geographic information and planning support systems. *Journal of Planning Education and Research*, 12(3): 184–198, 1993.

Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015.

Tony Hey and Mike C. Payne. Open science decoded. *Nature Physics*, 11:367–369, 2015.

Patrick Janssen, Ruize Li, and Akshata Mohanty. MÖbius, a parametric modeller for the web. In S. Chien, S. Choo, M. A. Schnabel, W. Nakapan, M. J. Kim, and S. Roudavski, editors, *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*, pages 157–166. The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), 2016.

James Keirstead, Mark Jennings, and Aruna Sivakumar. A review of urban energy system models: Approaches, challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 16(6):3847–3866, 2012.

Tim Kindberg, Matthew Chalmers, and Eric Paulos. Guest editors' introduction: Urban computing. *IEEE Pervasive Computing*, 6(3):18–20, aug 2007.

Daphne Lopez, M. Gunasekaran, B. Senthil Murugan, Harpreet Kaur, and Kaja M. Abbas. Spatial big data analytics of influenza epidemic in vellore, india. In *2014 IEEE International Conference on Big Data*. IEEE, 2015.

P. Nourian, S. Rezvani, I. S. Sariyildiz, and F. D. van der Hoeven. Spectral modelling for spatial network analysis. In Ramtin Attar, Angelos Chronis, Sean Hanna, and Michela Turrin, editors, *Proceedings of the Symposium on Simulation for Architecture and Urban Design (simAUD 2016)*. SimAUD, 2016.

Pirouz Nourian. *Configraphics: Graph Theoretical Methods for Design and Analysis of Spatial Configurations*. PhD thesis, Delft University of Technology, 2016.

Fernando Perez and Brian E. Granger. Ipython: A system for interactive scientific computing. *Computing in Science & Engineering*, 9(3), 2007.

Bianica Pires and Andrew T. Crooks. Modeling the emergence of riots: A geosimulation approach. *Computers, Environment and Urban Systems*, 61(A):66–80, 2017.

Roger S. Pressman. The manifesto for agile software development. Available at `http://nlp.chonbuk.ac.kr/SE/ch05.pdf`, 2009.

Pamela J. Robinson and Peter A. Johnson. Civic hackathons: New terrain for local government-citizen interaction? *Urban Planning*, 1(2), 2016.

Jingbo Shang, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu. Inferring gas consumption and pollution emission of vehicles throughout a city. In *KDD '14 Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1027–1036. ACM, 2014.

Helen Shen. Interactive notebooks: Sharing the code. *Nature*, 515:151–152, nov 2014.

Ana Simão, Paul J.Densham, and Mordechai (Muki) Haklay. 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, 2009.

Tomasz Szydlo, Robert Brzoza-Woch, Joanna Sendorek, Mateusz Windak, and Chris Gniady. Flow-based programming for iot leveraging fog computing. In *IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2017*. IEEE, 2017.

Jeff Templon and Jan Bot. The dutch national e-infrastructure for research. In *International Symposium on Grids and Clouds (ISGC) 2016*. Proceedings of Science, 2016.

Matthew Tenney and Renee Sieber. Data-driven participation: Algorithms, cities, citizens, and corporate control. *Urban Planning*, 1(2), 2016.

Upkar Varshney. Pervasive healthcare and wireless health monitoring. *Mobile Networks and Applications*, 12(2–3):113–127, 2007.

D. Volchenkov and Ph. Blanchard. Random walks along the streets and canals in compact cities: Spectral analysis, dynamical modularity, information, and statistical mechanics. *Physical Review E*, 75(2), 2007.

G. von Krogh. Open-source software. *MIT Sloan Management Review*, 44(3):14–18, 2003.

Xuebin Wei and Xiaobai A. Yao. The random walk value for ranking spatial characteristics in road networks. *Geographical Analysis*, 46(4):411–434, 2014.

Theodore Zahariadis, Andreas Papadakis, Federico Alvarez, Jose Gonzalez, Fernando Lopez, Federico Facca, and Yahya Al-Hazmi. Fiware lab: Managing resources and services in a cloud federation supporting future internet applications. In *IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), 2014*. IEEE, 2014.

Olmo Zavala-Romero, Arsalan Ahmed, Eric P. Chassignet, Jorge Zavala-Hidalgo, Agustin Fernández Eguiarte, and Anke Meyer-Baese. An open source java web application to build self-contained web gis sites. *Environmental Modelling & Software*, 62:210–220, 2014.

Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. In *UbiComp '11 Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98. ACM, 2011.

Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: when urban air quality inference meets big data. In *KDD '13 Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.

Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST) - Special Section on Urban Computing*, 5(3), 2014.

S. Zlatanova, P. Nourian, R. Gonçalves, and A. V. Vo. 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*, 2016.