SYMPOSIUM ON SIMULATION IN ARCHITECTURE + URBAN DESIGN
SimAUD 2020

BuildFit: A Cloud Computing Open Platform for Building-Lifecycle Data Analysis and Energy Modelling

ABSTRACT

In this paper, we present a platform that integrate three main aspects in the building industry: 1) Building data from both IoT devices and Building Management System (BMS), 2) Building Energy modelling and simulation engine and 3)Data analysis and optimization libraries, all of which are combined in one cloud computing platform whose interface is a visual-programming one. The platform is dedicated for non-professional programmers yet provide a procedural (parametric) interface that is developed to overcome some of the programming notations cognitive dimensions problems for the end-use developers such as visibility of components, hidden dependencies, role expressiveness, abstraction, secondary notations and consistency. At the same time, this research represents an evaluation of difficulty and amount of customization of the platform amongst other available tools. This platform was basically developed for the purpose of performing continuous life-time building energy calibration. However its uses could exceed this purpose to do temporal data analysis and visualization, building performance modelling and simulation - other uses -

Author Keywords

Guides; instructions; author's kit; conference publications; keywords should be separated by a semi-colon.

ACM Classification Keywords

I.6.1 SIMULATION AND MODELING: Computing methodologies Model development and analysis - Software and its engineering Visual languages

1 INTRODUCTION

It is obvious that building sector has witnessed massive development recently in the way by which building systems are managed in order to alleviate the massive energy consumption of this sector by better controlling the resources; providing sustainable and more efficient solutions; developing a better understanding of microclimate, buildings, and users' behaviours; and making better decisions based on either insightful ground-truth data (black-box approach), physics-based simulation models (white box approach) or a mixture of them (gray box approach). In this regard, it is worth mentioning that building sector is responsible for 30% of the world energy consumption and a third of the associated CO2 emissions [**?**].

On the other hand, the emergence of the micro-electro mechanical devices (MEMS) which lead to the ubiquitness of Internet-of-Things (IoT) devices (mobile devices, sensors, wearables, actuators ..etc) in the last few years to the extent that the number of interconnected IoT devices reached 11.2 billion by the time of writing this paper and continuing growing exponentially. These devices drastically changed computing paradigms, sensing and monitoring [**?**], which enabled massive flow of data from each aspect of life such that the data collected during the last two decades exceeded that which is collected from the beginning of history. The biggest challenge ,though, is not the availability of data itself, but rather managing this big data: transferring, storing, preprocessing, wrangling and mining to obtain knowledge, optimization and control in a robust cyberinfrastructure. This lead to cloud computing or ubiquitous computing, that is , computing data in place, without paying much effort in transferring data to local storage/processing machines using scalable storage and computational power on demand maintained by professionals and provided in a form of Infrastructure as a Service (IaaS) and Platform as a service (PaaS) [**?**, **?**, **?**].

The convergence of those technologies however, requires users with proper domain knowledge alongside with programming or procedural thinking skills for automating, prototyping, analyzing, building work-flows using the massive amount of data and IoT sensors. Thus the term End-use programmer first introduced by [**?**] or Novice developer is used to describe people who are not professional developers yet able to create/modify software without significant knowledge of programming language and code structures [**?**]. In fact, most programs today are written by novice programmers [**?**] as it had been estimated that by 2012, there would be less than 3 million professional programmers compared to more than 55 million end-use programmers, using spreadsheets, writing add-ons/plugins to support their work or add functionality to a software, running Matlab simulations, writing python codes, or IPython notebooks and so on [**?**]. The main difference between the end-use developer and the professional developer is the goal of the development. The former writes program to do specific task with the focus on getting this task done without paying so much attention to debugging, unit-testing or re-usability. however the latter develops, maintain, debug, and test a robust software for others to use. However they both face common challenges such as choosing which APIs, libraries, and functions to use; in addition, testing, verifying and debugging their codes regardless of the programming language used and its degree of complexity, for exam-

ple: an end-use programmer could use general purpose language such as C++ to achieve specific task without the intent to publish the code for other to use. On the other side, a professional programmer could use a simple markup language such as HTML/CSS to develop a commercial website.

Each programming has what is so called a programming notation, this notation could be textual, diagrammatic, gesture-based .. etc. which is used to represent the state of the world. The selection of the proper programming notation to describe the problem is a trade-off between a number criteria that are based on cognitive psychology introduced by [**?**], These are: **1)Viscosity (resistance to change)**, for example, if we need to change the interval time granularity of all sensors simultanously, how resistant would be the code? **2)Visibility (ability to view components easily)**, i.e. information are not hidden in encapsulations. **3)Premature commitment (constraints on the order of doing things)**, for example, the need to declare variable types or allocate memory initially. **4) Hidden dependencies (important links between entities are not visible)**, for instance, if there are interconnected entities, where the change one's value would affect the others, how visible is this interconnection?

1.1 Title and Authors

Your papers title, authors and affiliations should run across the full width of the page in a single column 17.8 cm (7 in.) wide. The title should be in Ariel 18-point bold; use Helvetica if Ariel is not available. Authors names should be in Times New Roman 12-point bold, and affiliations in Times New Roman 12-point (not bold, nor italic).

1.2 Normal or Body Text

Please use a 10-point Times New Roman font or, if this is unavailable, another proportional font with serifs, as close as possible in appearance to Times New Roman 10-point. The Press 10-point font available to users of Script is a good substitute for Times New Roman. On a Macintosh, use the font named Times not Times New Roman. Please use sans-serif or non-proportional fonts only for special purposes, such as headings or source code text.

2 PAGE NUMBERING, HEADERS AND FOOTERS

3 PRODUCING AND TESTING PDF FILES

4 CONCLUSION

ACKNOWLEDGMENTS