A Modernized Version of a 1D Soil Vegetation Atmosphere Transfer model for improving its future use in Land Surface Interactions Studies

Vasileios Anagnostopoulos[1,2], George Petropoulos[3,*],

Gareth Ireland[3], Toby N. Carlson[4]

[1]*Distributed and Knowledge Management Systems Lab, National Technical University of Athens, Greece*
[2]*InfoCosmos, Pindou 71, 13341, Athens, Greece*
[3]*University of Aberystwyth, Department of Geography and Earth Sciences, SY23 2EJ, Wales, United Kingdom*
[4]*Pennsylvania State University, Department of Meteorology, University Park, PA 16802, USA*

*Correspondence: *george.petropoulos@aber.ac.uk, Tel: +44-0-1970-621861*

ABSTRACT

SimSphere is a land biosphere model that provides a mathematical representation of vertical 'views' of the physical mechanisms controlling Earth's energy and mass transfers in the soil/vegetation/atmosphere continuum. Herein, we present recent advancements introduced to SimSphere code, aiming at making its use more integrated to the automation of processes within High Performance Computing (HPC) that allows using the model at large scale. In particular, a new interface to the model is presented, so-called "SimSphere-SOA" which forms a command line land biosphere tool, a Web Service interface and a parameters verification facade that offers a standardised environment for specification execution and result retrieval of a typical model simulation based on Service Oriented Architecture (SOA). SimSphere-SOA library can now execute various simulations in parallel. This allows exploitation of the tool in a simple and efficient way in comparison to the currently distributed approach. In SimSphere-SOA, an Application Programming Interface (API) is also provided to execute simulations that can be publicly consumed. Finally this API is exported as a Web Service for remotely executing simulations through web based tools. This way a simulation by the model can be executed efficiently and subsequently the model simulation outputs may be used in any kind of relevant analysis required.

The use of these new functionalities offered by SimSphere-SOA is also demonstrated using a "real world" simulation configuration file. The inclusion of those new functions in SimSphere are of considerable importance in the light of the model's expanding use worldwide as an educational and research tool.

Keywords: *SimSphere, SVAT, land surface interactions, Service Oriented Architecture, Web Service, software developments, Earth Observation*

1. Introduction

Land surface interactions govern the critical exchanges of energy and mass between the terrestrial biosphere and the atmosphere, and are major drivers of the Earth's system (Jung et al., 2011; North et al., 2015). There is an urgent need today for a better understanding and a more accurate monitoring of Earth's natural processes and interactions, evidently strengthened even more in the face of pressures from climate change and global food and water security (Coudert et al., 2008; Ireland et al., 2015; Mannschatz et al., 2016). The growing role of simulation in Earth systems science together with the increasing available computing power have resulted to an increase in the complexity of processes included in the design of simulators (Coon et al., 2016).

Land Surface Parameterisation schemes (LSPs, also known as Land Surface Models, LSMs) are one of the preferred scientific tools to quantify earth system interactions at different spatial and temporal resolutions. LSPs simulate a number of parameters characterising land surface interactions within the lower atmospheric boundary from a predefined set of surface characteristics (i.e. properties of soil, vegetation and water). One such group of LSPs includes Soil-Vegetation-Atmosphere Transfer (SVAT) models. Those mathematical models provide representation of vertical 'views' of the physical mechanisms controlling energy and mass transfers in the soil-vegetation-atmosphere continuum. SVATs provide estimates of the time course of soil and vegetation state variables at time-steps compatible with the dynamics of atmospheric processes. Also are able to describe the multifarious transfer processes through varying degrees of complexity, over different temporal and spatial scales (Ridler et al., 2012).

SimSphere is an example of a 1-D SVAT model, originally developed by Carlson and Boland (1978) and Lynn and Carlson (1990), and considerably modified to its current state with its latest version written in Java by Gillies et al., (1997) and Petropoulos et al., (2013a). Since its development it has been utilised in studies concerning the study of land surface interactions (North et al., 2015) and the examination of hypothetical scenarios studying feedback processes (Grantz et al., 1999). Furthermore, it has been used synergistically with Earth Observation (EO) data to derive spatiotemporal estimates of energy fluxes and/or soil moisture (Carlson, 2007). Variants of this technique are currently investigated by Space Agencies for developing related operational products (Chauhan et al., 2003; Piles et al., 2011; ESA STSE, 2012; Piles et al., 2016). An overview of the model use can be found in Petropoulos et al., (2009a).

Behind SimSphere's Graphical User Interface (GUI) lies an engine that executes the computation of a number of output parameters after a set of input parameters are provided by the user from the User Interface (UI). The computation engine is a direct port from an existing Fortran 77 codebase to Java 1.3. This port was originally undertaken 15 years ago, and during that time the code base has been amended with various computation routines by various developers for a number of applicational purposes. Moreover the GUI code is entangled with the logic of the computation engine and provides a custom interface by providing a set of parameters as an input to the computation routines. The introduction of Service Oriented Architecture (SOA) practices in the remote sensing realm is relatively new (El-Sharkawi et al., 2013). Modern execution environments, like cloud platforms or numerical computation Domain Specific Languages (DSLs), like Modelica, are not able to create value from the research work in the codebase mostly due to a monolithic, standards incompliant approach. One can attribute the problem to the lack of SOA design. This is also related to the porting of business logic from old programming languages, like Fortran 77, which contributes much syntactic noise. SOA allows the service specification and modularity in a multi-lingual environment. The SOA rules outlined in Bell (2008) and Schroth and Janner, (2007) could potentially serve to improve the separation of roles. According to these authors, the researcher is not a developer and vice versa. Both roles have their own domain

87 specific knowledge and skills and through SOA the developer can create a standardised interface
88 that exposes the business logic engineered by the researcher. This can be done through protocols
89 and standards, something that SOA reinforces.

90 In the purview of the above, the aim of this work has been to apply modern trends in software
91 development to an existing codebase in the field of EO and land surface modeling to outline an
92 SOA migration path, display its benefits and improve the software quality. SimSphere is an ideal
93 example to be chosen as a case study as it is widely adopted in research, its use is rapidly
94 expanding worldwide, and, the software toolkit is minimal enough to have a proof-of-concept,
95 while rich enough to display our approach. More specifically, the existing GUI code is removed
96 from the model and then SOA design patterns are applied in order to create a command line
97 implementation suitable for cloud deployments or High Performance Computing (HPC) execution,
98 which subsequently exposes the model functionality as a service.

99

100 **2. SimSphere Description**

101 SimSphere is a one-dimensional land biosphere model. It simulates a series of physical processes
102 that take place as a function of time in a column that extends from the soil root zone up to a level
103 higher than the surface vegetation canopy. Three main systems within the models' architecture,
104 include the *physical*, the *vertical* and the *horizontal* layers (**Figure 1**). The *physical* components
105 ultimately determines the microclimate conditions in the model, grouped into three categories,
106 *radiative, atmospheric and hydrological*. The *vertical structure* components, effectively correspond
107 to the Planetary Boundary Layer (PBL) and are divided into three layers - a surface mixing layer, a
108 surface of constant flux layer and a surface vegetation or bare soil layer, where the depths of the
109 first layer is somewhat variable with time, growing throughout the day as sensible heat is added
110 from below. The depth of the constant flux and vegetation layers are set in the model input,
111 although the depth of a bare soil transition (between soil and air) layer is variable in time
112 depending on the wind speed and the surface roughness. The substrate layer refers to the depth
113 of the soil over which heat and water is conducted. The processes and interactions simulated by
114 SimSphere develop over a 24-h diurnal cycle at a chosen time step, starting from a set of initial
115 conditions given in the early morning (at 05.30 am local time) with a continuous evolving
116 interaction between soil, plant and atmosphere layers. A number of input parameters are
117 required to parameterise the model, categorised into 7 defined groups (**Table 1**). Model provides
118 predictions as a function of time for a total of more than 30 variables (**Table 2**). A detailed
119 description of SimSphere architecture can be found in Gillies (1993). The current version of the
120 model is globally distributed from Aberystwyth University, UK (http://www.
121 aber.ac.uk/simsphere).

122

123 **3. Existing Structure and Architecture of SimSphere**

124 The development of SimSphere-SOA has primarily been motivated by efforts to increase the
125 usability of the original software model. In its currently distributed version of SimSphere, the GUI
126 code was written as a presentation and configuration layer to an Extensible Markup Language
127 file. This XML file provided the configuration layer of the data, whilst the persistency layer that
128 transformed user data to the XML file and back was written by hand using custom data types. A
129 component to a server based proprietary execution environment was also developed, but
130 SimSphere software is typically used as a desktop application. The interface to the Fortran 77
131 computational logic was done through a function which took all the arguments used for the
132 computation as inputs. In **Figure 2** is provided an overview of the current architecture. From it

133 and the usage of the software, it can be observed that there is room for further improvements in
134 the original SimSphere software toolkit and in particular:

  ➢ There is no way for exporting the model inversion data.

  ➢ The configuration (that configures the internal data-structures with user input) and
    validation layer (that examines user input for errors) is manually written which is
    error prone.

  ➢ The UI couples tightly with the validation and persistency layer (that loads
    specification and save results) . No third party interface for other UIs is possible.

  ➢ There are two parts in the validation layer. The first is manually written and the other
    is written on old technology (Document Type Definition, DTD).

  ➢ The application is not available for headless installations because of the UI coupling.

  ➢ The application cannot be used remotely through Web Services.

  ➢ Uses legacy unsupported code.

  ➢ The application cannot be run in HPC or in batch mode.

147 There are also a number of other possible issues to address on the model distributed version not
148 evident in **Figure 2** that mainly pertain to maintainability. Typically current deprecated methods
149 do not work as expected and make functionalities of the software unusable in current versions of
150 Java. It is also obvious from the code base that an entangling of roles leads to major shortcomings
151 in the development process.

4. SimSphere-SOA Developments

154 The newly developed code within SimSphere-SOA was structured to follow the SOA principles.
155 For the computational logic layer an orchestrator was created, behind a mega-function interface,
156 which uses the services of the persistency layer to read the XML and export the results of the
157 computation to a Comma Separated Values (CSV) formatted file (Shafranovich, 2005). The new
158 model architecture is shown in **Figure 3**. As an SOA approach, SimSphere-SOA provides the
159 service of simulation by consuming messages in XML format and producing messages in CSV
160 format, allowing them to be consumed by a third party application. Using the self-documenting
161 configuration and validation layer, various applications can be designed around the Application
162 Programming Interface (API) as illustrated in **Figure 4**.

163 There are two endpoints per CSV. The first one takes the XML file as input and produces a CSV of a
164 time based simulation for a whole day. The first column is the simulation time in 15-minute (or
165 higher) steps, whereas the remaining columns contain the evolution of various geophysical
166 quantities as samples at these time instances. The second type of simulation runs various
167 scenarios for a specific time of a particular day encoded as Fractional Vegetation Cover (FVC) and
168 Surface Moisture Availability pairs. These two quantities vary in the 0.1 floating point range. The
169 user can specify steps to sweep the two ranges independently, and for each combination of values,
170 quantities of interest to the remote sensing community are computed. The results are exported as
171 a CSV in the desktop version; however the application can run also as a server where it exposes
172 the two endpoints as shown in **Table 3.** The request headers should be Content Type:
173 application/xml and   Accept: text/csv. This last step can be used for fast model inversion.

176

177 Apart from the various Computer Science (CS) considerations, this new application is an enabler
178 for sophisticated Sensitivity Analysis (SA) tooling in SimSphere. This is very important
179 functionality in terms of future efforts related to performing an all-inclusive the verification of the
180 model. Indeed, SA can help to understand the behavior of a model and in establishing the
181 dependency of the model outputs on its input parameters in how different parts of the model
182 interplay. By means of an SA, irrelevant parts of the model may be dropped or a simpler model
183 can be built or extracted from a more complex one (so-called model lumping), reducing, in some
184 cases significantly, the required computing power in running a model. SA also provides a valuable
185 method to identify critical input parameters and rank them in order of importance. The latter can
186 offer important guidance to the design of experimental programs as well as to more efficient
187 model coding or calibration (e.g. Petropoulos et al. 2009b; 2013b). The new model architecture
188 presented herein allows the creation of a generalised SA scheme which decouples the application
189 from executing various runs manually in order to derive the results. As SimSphere is provided as
190 a service, one needs to change the XML files, create CSV files and run the analysis algorithm via a
191 scripting mechanism. SimSphere-SOA is completely stateless (two executions of SimSphere are
192 independent) and is suitable for HPC environments, since it creates a stateless cluster (as
193 illustrated in **Figure 5**).

194

195 **5. SimSphere-SOA Software Availability**

196 SimSphere-SOA product has been also developed as open source software, as its predecessor
197 SimSphere, and is hence, released under the terms of the GNU General Public License. The
198 software code of this new model version is also freely distributed from the main web site from
199 where the model is distributed globally maintained by Aberystwyth University, UK
200 (www.aber.ac.uk/simsphere).

201

202 **6. Implementation**

203 *6.1. Validation layer*

204 A smaller software tool was created to amend and improve upon SimSphere original model
205 architecture, whilst also ensuring that SOA guidelines and use standards were respected in order
206 to facilitate and the work of the EO community. As a first step, the two validation layers that
207 existed in the currently distributed model version were unified into a single layer in SimSphere-
208 SOA. The two validation layers were very different; one was contained within the application
209 while the other one was external. In order to extract the internal validation layer, amendments to
210 the original code had to be made. Following assessment of the code, the requirements for the
211 unified validation layer which were not met by the existing approach were identified as the
212 following:

213 ➢ The data types of the XML should be specified.

214 ➢ The validation layer should be self-documenting for a developer.

215 ➢ The validation layer should include comments for the non-developer end users.

216 ➢ The validation layer must lie externally to application in order to meet the previous
217 requirements.

218   ➢   The validation layer should be used by the software before using the data for
219   computations.

220   ➢   The validation should be standardised.

221 Having identified these requirements, the XML Schema Definition (XSD) version 1.1 was utilised
222 for the application. Compared to the previous version 1.0, the updated version has the added
223 value of the inclusion of validation, which was previously available as the separate Schematron
224 standard. Using this solution, the validation logic could be externalised as a self-documenting,
225 easily comprehensible compact document. The other benefit is that the same document could be
226 used to create a persistent layer. In the case of SimSphere-SOA, the XSD file contained 3 more
227 validation rules because extra domain specific knowledge from the expert, which is not encoded
228 in the code base, was included. The new validation/persistency approach could also find
229 validation errors that were not possible in the previous approach and were identified as bugs
230 leading to runtime crashes, or worse, incorrect results. Both the newly developed and the existing
231 approach was largely based on data types and their constraints. Perhaps the strongest advantage
232 over the existing SimSphere application is the Look Up Table (LUT) functionality with selection
233 among discrete alternatives and uniqueness provided by XSD 1.0 specification and does not exist
234 in DTD specification. The XSD 1.1 specification facilitated the assertions outlined in **Table 4**
235 which was not available in the previous version. There were also gains in readability. The
236 Longitude element is a typical example. In our case, this is compactly represented as an XSD data
237 type with upper and lower bounds (**Table 5**). In the old approach, this information was encoded
238 in an XML file which resulted in the rule being copied between XML files with the risk of possible
239 typographic mistakes. In the new approach, the XML corresponding element is simplified as one
240 can see from **Table 6**, with evident gains in readability and robustness. The documentation of
241 various XML parameters through the excellent xs3p package allowed for the web presentation to
242 be easily highlighted and formatted.

243

244 *6.2. Serialisation layer*

245 While the Java Architecture for XML Binding (JAXB) API does not support version 1.1 of the XSD
246 (XML Schema) specification, the assertions present in 1.1 could be commented out to create the
247 1.0 conforming document. In this respect, the serialisation layer was automatically generated as a
248 Java package from the 1.0 document. Programmatically, given a conforming XML, the relevant
249 classes specified through the 1.1 XSD version could be filled with the XML data. The corresponding
250 classes in the old version were analysed by the cloc tool (CLOC). The results are shown in **Table 7**.

251 It should be noted that in the XSD, despite the validation and serialisation specification being
252 roughly four times bigger in terms of lines of code in comparison to the DTD approach, as opposed
253 to the manually generated Java it is almost four times smaller in terms of lines of code, and does
254 not require Java expertise. Moreover the validation logic is contained within a single file in
255 contrast to the previous approach which amounted to 39 different files (including DTD). Even if
256 the logic in the XSD were more modularised, spanning multiple files for readability, the overhead
257 would be minimal. The new configuration and validation layer is self-documenting and the
258 validation could be done with the third party library Apache Xerces (Apache Xerces). Notably,
259 application specific validation was not relied upon. There were also significant improvements
260 related to the XML files in comparison to the original approach. The sample XML which was
261 provided with the application had been reduced by 3.5 times in terms of file size in comparison to
262 the original (**Table 7**).

263 With regards to the presentation layer, due to deprecated features and bugs, the GUI code had to
264 be re-written. Instead of using this costly approach in the short term, a standards compliance was
265 adopted to tackle the problem. It was observed that the outputs were double valued, in column
266 form. Given that the model could simulate the 24 hours of a day at 15 minutes resolution, a
267 uniform column based tabulation was available. The most widely used form for such data is the
268 CSV format, which is a standard format supported by dozens of software. Consequently a solution
269 was engineered to convert the internal representation in CSV format for output. The export
270 functionality was not present in the previous approach. Through CSV, the presentation was
271 delegated to other well-established and maintained tools such as LibreOffice.

272 *6.3. Demonstration of new functions*

273 In order to demonstrate the new functions offered by SimSphere-SOA a simulation
274 configuration file was used for the case of running a simulation for Borgo Cioffi Italian
275 experimental site (latitude 40.617 and longitude 14.933) specifically for the date 17 November
276 2004. After the conversion it could be used the xsd to identify various mistakes (**Figure 6**) that
277 went uncaught by the standard SimSphere distribution. After correcting these mistakes the next
278 challenge was related to the sounding atmospheric profile data provision. In the standard
279 SimSphere distribution, the user has to manually provide such sounding data. They are typically
280 taken from the Department of Atmospheric Science of the University of Wyoming. These come in
281 a standard format. Here in the soundings were acquired for the corresponding date from the
282 nearby weather station, namely LIRE. The provided data were saved to a .csv file and using an
283 accompanying utility in the SimSphere-SOA distribution, namely csv2soundingset.java, it
284 automatically generated the <SoundingSet> element in the correct format and units suitable for
285 SimSphere-SOA (**Figure 7**). This was not possible in the standard version, as the user must
286 employ a manual, error prone procedure. The tool respects the constraint posed by the core of
287 Simsphere which is a limit of 51 Sounding elements. Also, notably the previous version had only
288 11 manually converted measurements. After checking the .xml file again for correctness via the
289 XSD file, it was used to perform simulations. A .csv file was effortlessly generated that encoded the
290 results of the simulation. A standard spreadsheet program was used to create the figures, namely
291 MS Excel. The .xml used to generate the simulation results, namely sample.xml, is available at the
292 corresponding Github project in the folder simsphere_xsds. An example prediction of the time
293 evolution of Sensible Heat Flux of this day is illustrated in **Figure 8**.

294 **7. Conclusions & Future Work**

295 Herein, the applicability of the SOA architecture to a SVAT modeling tool was demonstrated. The
296 innovation of the approach described herein is the exporting of validation logic encoded in code
297 to an XSD. XSD can capture these constraints while the previous DTD validation could not. With
298 XSD more accurate sensitivity analysis and model outputs validation can be performed while the
299 user can use the inline documentation in order to insert valid values. Even if the input is
300 erroneous, by using standard validation interfaces of XML editors the user can accurately describe
301 and comfortably validate the input before submitting it. The validation could be used as a
302 portable interface generator between developers. In the previous SimSphere product edition,
303 manually generated Java code had to be exchanged. In contrast, the new SimSphere-SOA allows
304 the user to analyse the input from the .csv provided as output which are compatible with
305 numerous applications. The executions can be run in headless environments (for example in HPC)
306 in parallel or through shell scripts.

307 SimSphere-SOA development has also resulted in the improvement in the maintainability of the
308 application, code reduction and flexibility of the original model. The improvement in
309 maintainability and code size is due to the automated validation and generation of a persistency

layer, whereas the flexibility owes its improvement to the SOA design. Moreover, the stateless nature of SimSphere-SOA allows it to scale in HPC environments. The stateless is expressed by the creation of two .csv files per XML which can be executed in parallel. While the first .csv allows the observation of time evolution, the second one is very important in running model inversion.

Future work may focus on the GUI re-design using modern practices and technologies that have proved their values. For this purpose a number of options can be explored including Java FX technology (current on-going implementation) or the more standardised approach of HTML5/EcmaScript6 following the web centric trend. A perhaps more pressing issue with SimSphere is related to the maintainability of the scientific investment in computational logic. In order to successfully separate the role of developer and researcher, future work on the model will be required to transform the code base to a researcher friendly DSL like OpenModelica. This development will be of key value in demonstrating a workflow that brings together the developer and the researcher with minimum overlap of responsibilities to each other.

References

Bell, M. (2008). Introduction to Service-Oriented Modeling. Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. p. 3. ISBN 978-0-470-14111-3.

Carlson, T. (2007). An overview of the" triangle method" for estimating surface evapotranspiration and soil moisture from satellite imagery. Sensors, 7(8), 1612-1629.

Carlson, T. N., & Boland, F. E. (1978). Analysis of urban-rural canopy using a surface heat flux/temperature model. Journal of Applied Meteorology, 17(7), 998-1013.

Coon, E.T, J.D. Moulton and S.L. Painter (2016). Managing complexity in simulations of land surface and near-surface processes. Environmental Modelling & Software, 78, 134-149.

Chauhan, N. S., Miller, S., & Ardanuy, P. (2003). Spaceborne soil moisture estimation at high resolution: a microwave-optical/IR synergistic approach. International Journal of Remote Sensing, 24(22), 4599-4622.

Coudert, B., Ottlé, C., & Briottet, X. (2008). Monitoring land surface processes with thermal infrared data: Calibration of SVAT parameters based on the optimisation of diurnal surface temperature cycling features. Remote Sensing of Environment, 112(3), 872-887.

El-Sharkawi, A., Shouman, A., & Lasheen, S. (2013). Service Oriented Architecture for Remote Sensing Satellite Telemetry Data Implemented on Cloud Computing. International Journal of Information Technology & Computer Science, 5(7), 12.

European Space Agency: Support to Science Element, a Pathfinder for Innovation in Earth Observation, ESA, available at: http://due.esrin.esa.int/stse/files/document/STSE_report_121016.pdf (last access: 10 July 2014), 2012.

Gillies, R. R., Kustas, W. P., & Humes, K. S. (1997). A verification of the'triangle'method for obtaining surface soil water content and energy fluxes from remote measurements of the Normalized Difference Vegetation Index (NDVI) and surface e. International journal of remote sensing, 18(15), 3145-3166.

Gillies, R.R., 1993. A physically-based land sue classification scheme using remote solar and thermal infrared measurements suitable for describing urbanisation. PhD Thesis, University of Newcastle, UK, 121 pp.

Grantz, D. A., Zhang, X., & Carlson, T. (1999). Observations and model simulations link stomatal inhibition to impaired hydraulic conductance following ozone exposure in cotton. Plant, Cell & Environment, 22(10), 1201-1210.

Ireland, G., G.P. Petropoulos, T.N. Carlson & S. Purdy (2015): Addressing the ability of a land biosphere model to predict key biophysical vegetation characterisation parameters with Global Sensitivity Analysis Environmental Modelling & Software, 65, 94-107.

Jung, M., Reichstein, M., Margolis, H. A., Cescatti, A., Richardson, A. D., Arain, M. A., ... & Williams, C. (2011). Global patterns of land-atmosphere fluxes of carbon dioxide, latent heat, and sensible heat derived from eddy covariance, satellite, and meteorological observations. Journal of Geophysical Research: Biogeosciences (2005–2012), 116(G3).

Lynn, B. H., & Carlson, T. N. (1990). A stomatal resistance model illustrating plant vs. external control of transpiration. Agricultural and Forest Meteorology, 52(1), 5-43.

Mannschatz, T, T. Wolf & S. Hulsmann (2016): Nexus Tools Platform: Web-based comparison of modelling tools for analysis of water-soil-waste nexus. Environmental Modelling & Software, 76, 137-153.

North, M. R., Petropoulos, G.P., Rendall, D.V., Ireland, G.I. & J.P. McCalmont (2015): Evaluating the capability of a land biosphere model in simulating land surface processes: results from different European Ecosystems. DOI: doi:10.5194/esdd-6-217-2015 Earth Surface Dynamics Discussions.

Petropoulos, G., Carlson, T. and Wooster, M. J. (2009a): An Overview of the Use of the SimSphere Soil vegetation Atmospheric Transfer (SVAT) Model for the Study of Land Atmosphere Interactions, Sensors, 9, 4286-4308.

Petropoulos, G., Wooster, M. J., Kennedy, K., Carlson, T.N. and Scholze, M. (2009b): A global sensitivity analysis study of the 1d SimSphere SVAT model using the GEM SA software, Ecol. Model., 220, 2427-2440.

Petropoulos, G. P., Konstas, I., and Carlson, T. N (2013a): Automation of SimSphere Land Surface Model Use as a Standalone Application and Integration with EO Data for Deriving Key Land Surface Parameters, European Geosciences Union, 7–12 April 2013, Vienna, Austria.

Petropoulos, G. P., Griffiths, H., and Tarantola, S. (2013b): Sensitivity analysis of the SimSphere SVAT model in the context of EO-based operational products development, Environ. Modell. Softw., 49, 166–179, 2013c.

Piles, M., Camps, A., Vall-Llossera, M., Corbella, I., Panciera, R., Rüdiger, C., ... & Walker, J. (2011). Downscaling SMOS-derived soil moisture using MODIS visible/infrared data. Geoscience and Remote Sensing, IEEE Transactions on, 49(9), 3156-3166.

Piles, M., G.P. Petropoulos, G. Ireland & N. Sanchez (2016): A Novel Method to Retrieve Soil Moisture at High Spatio-Temporal Resolution Based on the Synergy of SMOS and MSG SEVIRI observations. Remote Sensing of Environment [in press].

397   Ridler, M. E., Sandholt, I., Butts, M., Lerer, S., Mougin, E., Timouk, F., … & Madsen, H. (2012).
398   Calibrating a soil–vegetation–atmosphere transfer model with remote sensing estimates of
399   surface temperature and soil surface moisture in a semi-arid environment. Journal of
400   Hydrology, 436, 1-12.

401   Schroth, C., & Janner, T. (2007). Web 2.0 and SOA: Converging Concepts Enabling the Internet of
402   Services. IT Professional 9, Nr. 3, pp. 36-41, IEEE Computer Society.

403   Shafranovich, Y. (October 2005). "Common Format and MIME Type for CSV Files". Network
404   Working Group (RFC 4180)

405
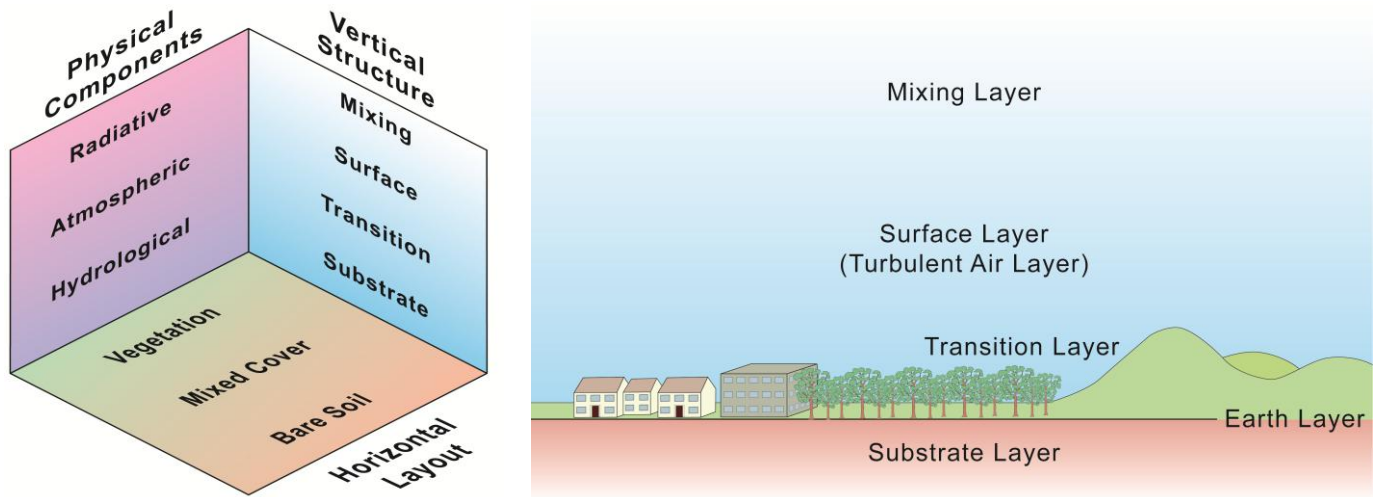
Fig 1: Basic structure of SimSphere, with the different model components summarised
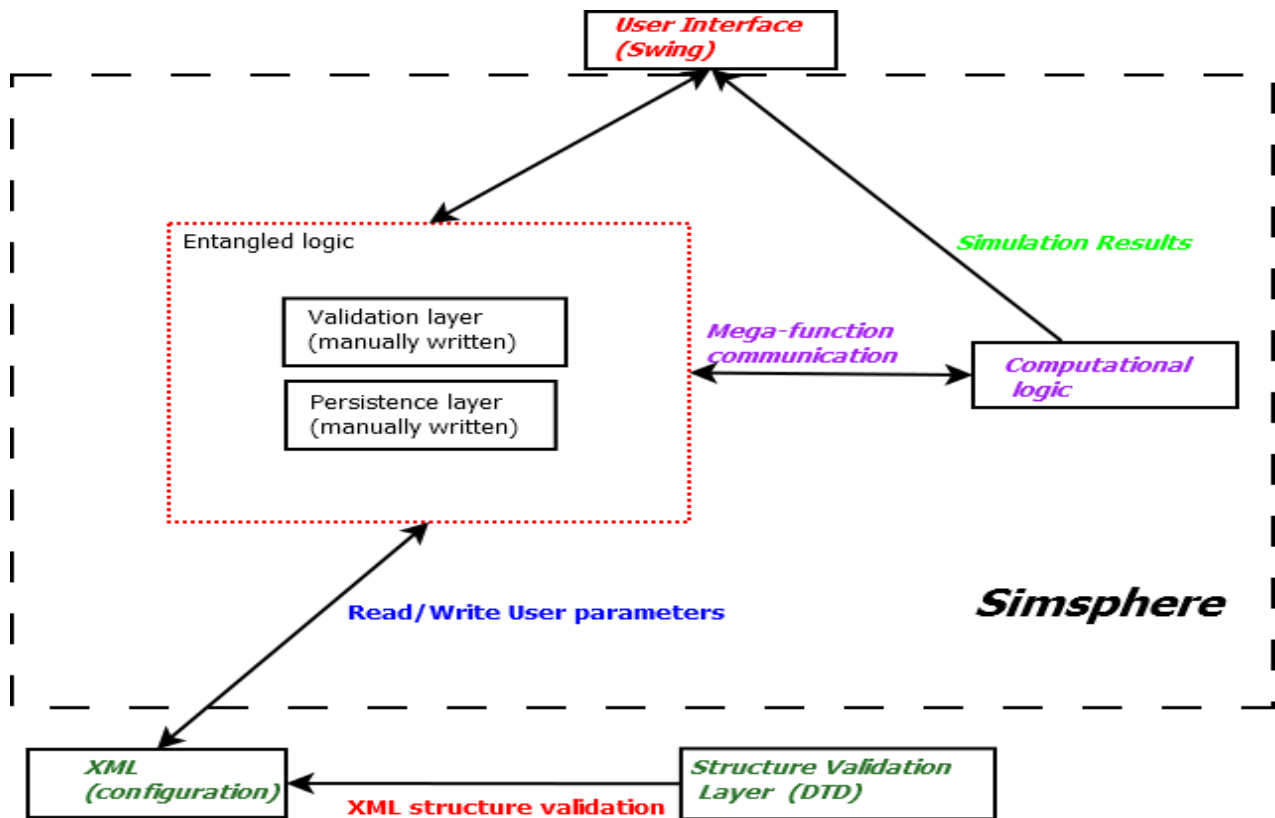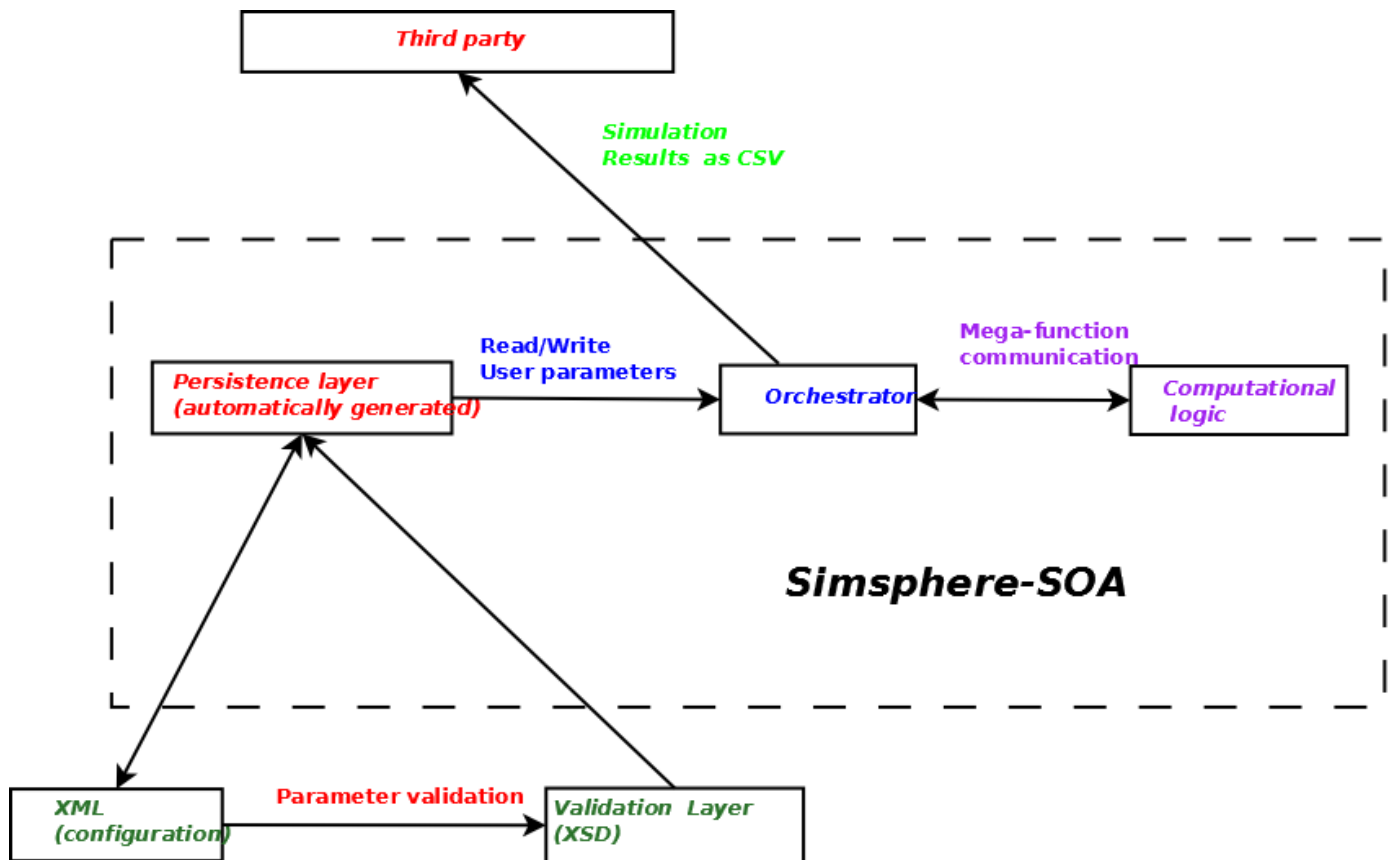
Fig. 2. Original SimSphere architecture

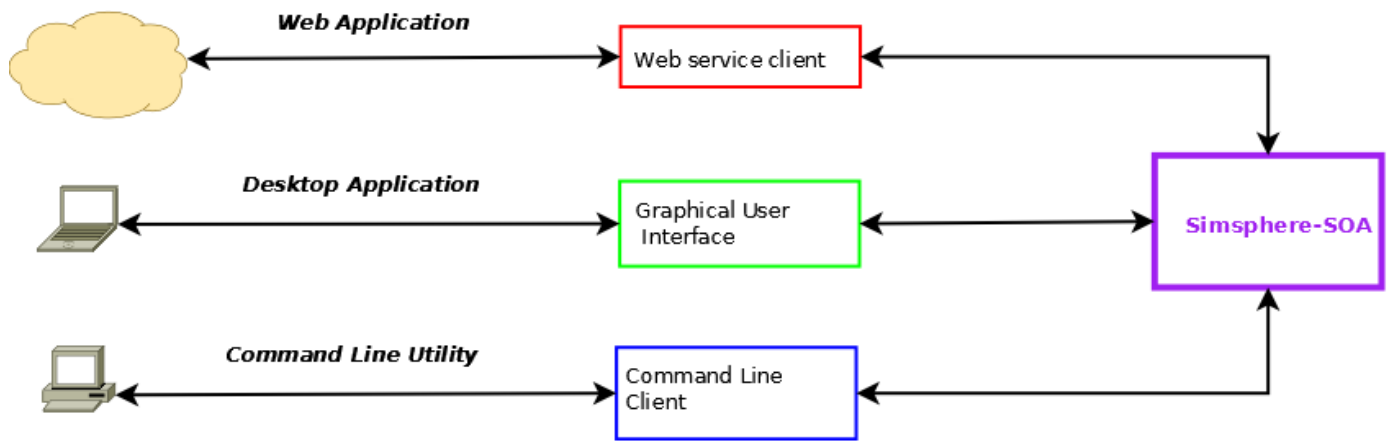Fig. 3 The new friendly architecture of SimSphere-SOA
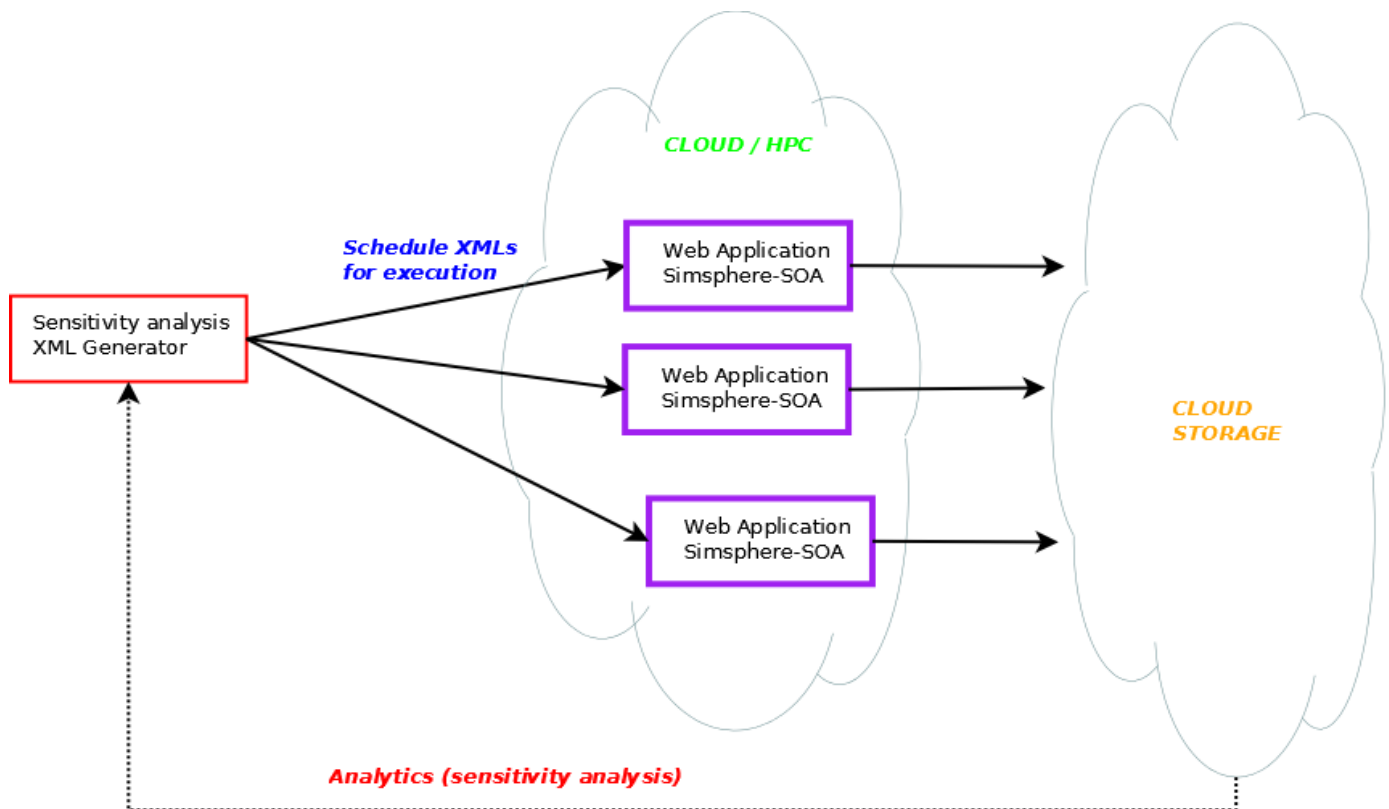
Fig. 4 Example of SimSphere-SOA applications

Fig. 5 SimSphere-SOA in cluster mode

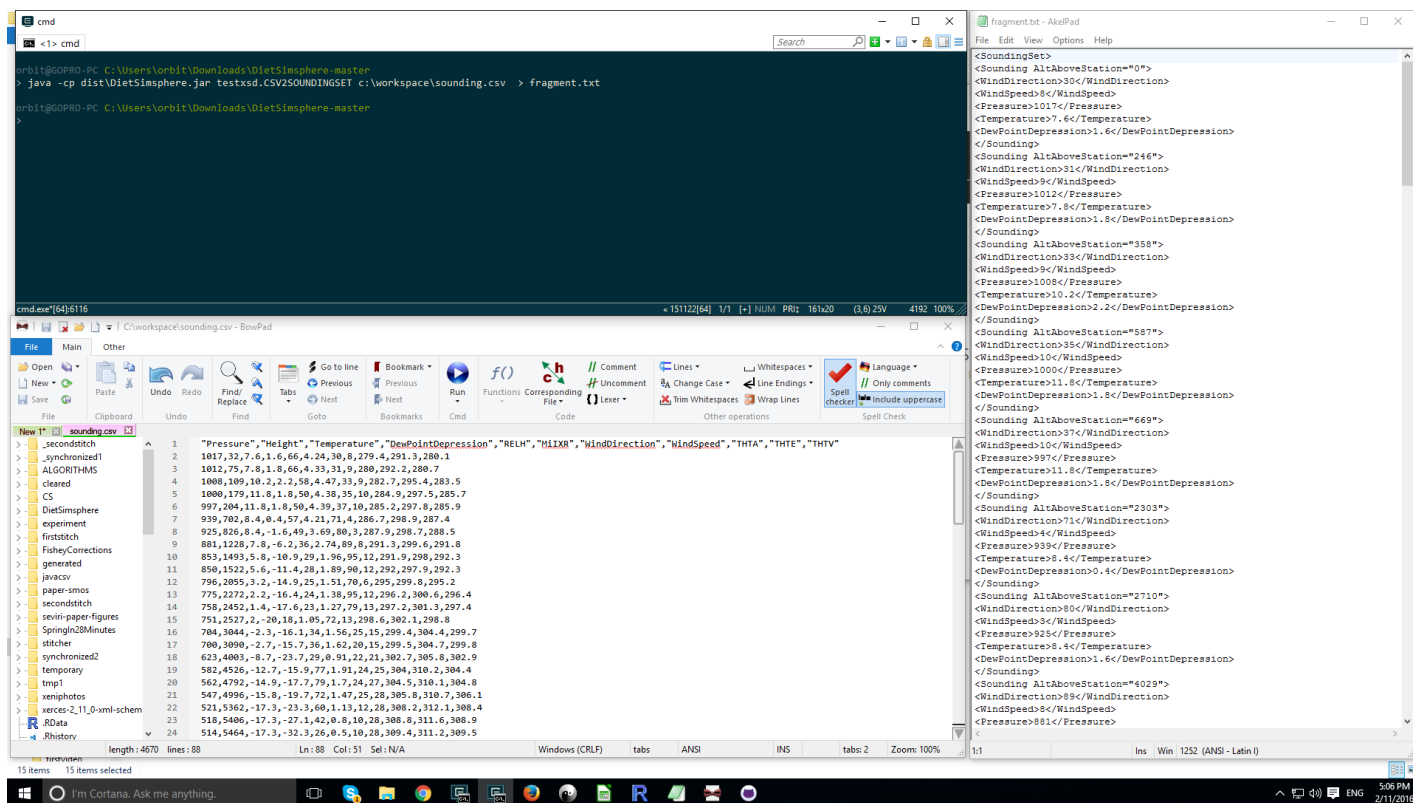Fig. 6 Validation failure on converted xml values.

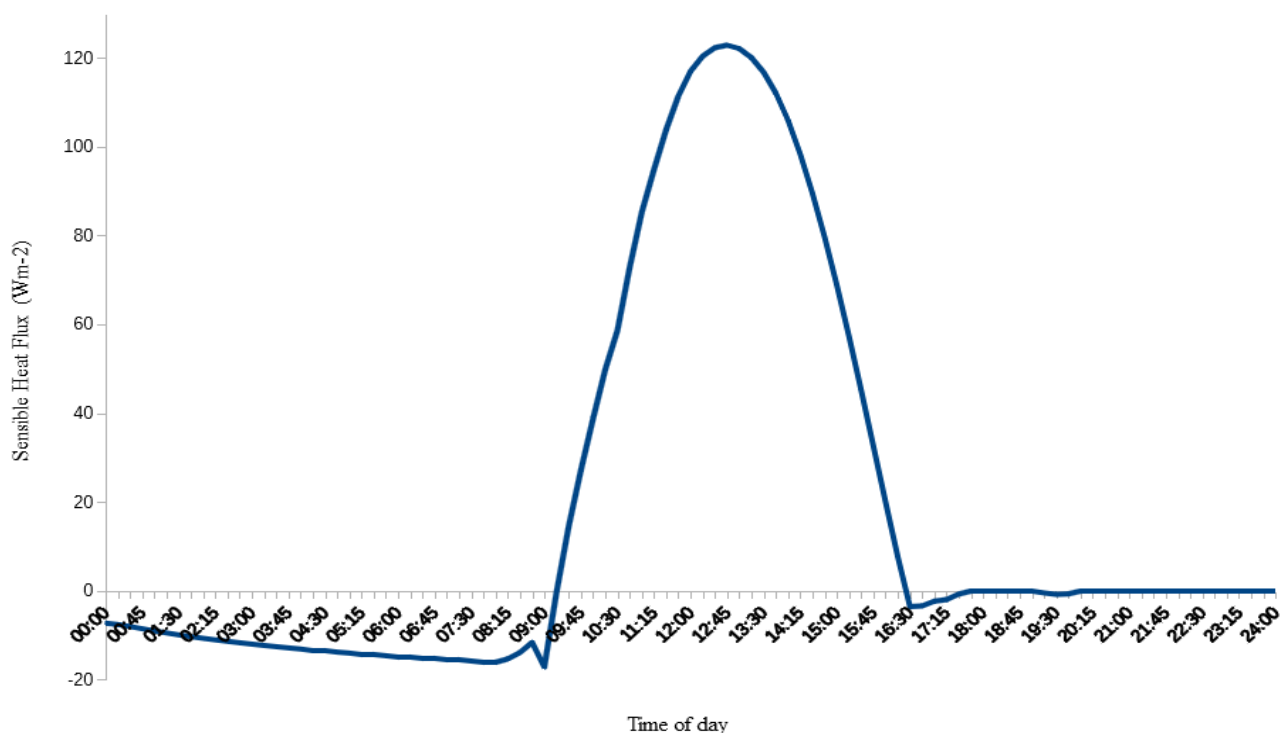Fig. 7 XML fragment generation from imported sounding data.

Fig. 8 Sensible Heat Flux predicted by the Simsphere-SOA

List of Tables

Table 1: Summary of the main SimSphere inputs.

NAME OF THE MODEL INPUT	PROCESS IN WHICH PARAMETER IS INVOLVED	MIN VALUE	MAX VALUE
Slope *(degrees)*	TIME & LOCATION	0	45
Aspect *(degrees)*	TIME & LOCATION	0	360
Station Height *(meters)*	TIME & LOCATION	0	4.92
Fractional Vegetation Cover *(%)*	VEGETATION	0	100
LAI *(m^2m^{-2})*	VEGETATION	0	10
Foliage emissivity *(unitless)*	VEGETATION	0.951	0.990
[Ca] (external [CO_2] in the leaf) *(ppmv)*	VEGETATION	250	710
[Ci] (internal [CO_2] in the leaf) *(ppmv)*	VEGETATION	110	400
[03] (ozone concentration in the air) *(ppmv)*	VEGETATION	0.0	0.25
Vegetation height *(meters)*	VEGETATION	0.021	20.0
Leaf width *(meters)*	VEGETATION	0.012	1.0
Minimum Stomatal Resistance *(sm^{-1})*	PLANT	10	500
Cuticle Resistance *(sm^{-1})*	PLANT	200	2000
Critical leaf water potential *(bar)*	PLANT	-30	-5
Critical solar parameter *(Wm^{-2})*	PLANT	25	300
Stem resistance *(sm^{-1})*	PLANT	0.011	0.150
Surface Moisture Availability *(vol/vol)*	HYDROLOGICAL	0	1
Root Zone Moisture Availability *(vol/vol)*	HYDROLOGICAL	0	1
Substrate Max. Volum. Water Content *(vol/vol)*	HYDROLOGICAL	0.01	1
Substrate climatol. mean temperature *(^oC)*	SURFACE	20	30
Thermal inertia *($Wm^{-2}K^{-1}$)*	SURFACE	3.5	30
Ground emissivity *(unitless)*	SURFACE	0.951	0.980
Atmospheric Precipitable water *(cm)*	METEOROLOGICAL	0.05	5
Surface roughness *(meters)*	METEOROLOGICAL	0.02	2.0
Obstacle height *(meters)*	METEOROLOGICAL	0.02	2.0
Fractional Cloud Cover *(%)*	METEOROLOGICAL	1	10
RKS (satur. thermal conduct.*(Cosby et al., 1984)*	SOIL	0	10
Cosby B *(see Cosby et al., 1984)*	SOIL	2.0	12.0
THM (satur.vol. water cont.) *(Cosby et al., 1984)*	SOIL	0.3	0.5
PSI (satur. water potential) *(Cosby et al., 1984)*	SOIL	1	7
Wind direction *(degrees)*	WIND SOUNDING PROFILE	0	360
Wind speed *(knots)*	WIND SOUNDING PROFILE	---	---
Altitude *(1000's feet)*	WIND SOUNDING PROFILE	---	---
Pressure *(mBar)*	MOISTURE SOUNDING PROFILE	---	---
Temperature *(Celsius)*	MOISTURE SOUNDING PROFILE	---	---
Temperature-Dewpoint Temperature *(Celsius)*	MOISTURE SOUNDING PROFILE	---	---

Table 2: Summary of the main s outputs simulated by SimSphere.

SimSphere model Outputs				
Output Name	**Units**		**Output Name**	**Units**
Air temperature at 1.3m	^{o}C		Radiometric Temperature	^{o}C
Air temperature at 50m	^{o}C		Root Zone moisture Avail.	n/a
Air temperature at foliage	^{o}C		Sensibel heat flux	Wm^{-2}
Bowen ratio	n/a		Short-wave flux	Wm^{-2}
$[CO_2]$ on canopy	ppmv		Specific humidity at 1.3m	gKg^{-1}
$[CO_2]$ flux	$micromolesm^{2}s^{-1}$		Specific humidity at 50m	gKg^{-1}
Epidermal water potential	Bars		Specific humidity at foliage	gKg^{-1}
Global O_3 flux	$Ugm^{-2}s^{-1}$		Stomatal resistance	sm^{-1}
Ground flux	Wm^{-2}		Surface moisture availability	n/a
Ground water potential	bars		Vapor pressure deficit	Mbar
Latent Heat flux	Wm^{-2}		Water Use Efficiency	n/a
Leaf water potential	bars		Wind at 10m	Kts
Net Radiation	Wm^{-2}		Wind at 50m	Kts
$[O_3]$ canopy	ppmv		Wind in foliage	Kts
$[O_3]$ flux plant	$Ugm^{-2}s^{-1}$			

Table 3. Web service endpoints of SimSphere-SOA

Endpoint	Description	Method
http://localhost:8080/timebased	Time based	POST
http://localhost:8080/convolution	Convolution simulation	POST

Table 4: Two examples XSD 1.1 assertions used

Description	Assertion
Altitude above station must start at 0	<xsd:assert test="count(./simsphere:Sounding[@AltAboveStation eq 0]) gt 0"/>
Minimum temperature must be below maximum temperature.	<xsd:assert test="@MinTemperature le @MaxTemperature"/>

Table 5: The longitude definition as used by SimSphere-SOA

```
<xsd:element name="Longitude">
  <xsd:annotation>
    <xsd:documentation>
      The longitude in degrees.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="-180"/>
      <xsd:maxInclusive value="180"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Table 6. Comparison of old and new approaches in providing the longitude parameter in XML input

Approach	Code

OLD	`<ParamLabel Format="##0.##" Label="Longitude (deg)" ParmElem="Longitude">`
	`</ParamLabel>`
	
	`<Longitude Value="96.55">`
	`  <BoundedRange MaxType="closed" Maximum="180" MinType="closed"`
	`      Minimum="-180" RangeID="ID102">`
	`  </BoundedRange>`
	`</Longitude>`

NEW	`<Longitude>96.55</Longitude>`

Table 7: Analysis using CLOC of old and new serialization and validation codebases

Approach	Files	Blanks	Comments	Lines of Code	File Size
OLD (Java only serialization)	38	1152	2069	4393 (manual)	
NEW (Java only)	0	N/A	N/A	0 (manual)	
OLD (DTD only)	1	N?A	N/A	311	
NEW (XSD only)	1	N/A	N/A	1145	
OLD XML Input					106KB
NEW XML Input					26KB