Models in the Cloud: Exploring Next Generation Environmental Software Systems

SAMREEN, Faiza <http://orcid.org/0000-0002-9522-0713>, SIMM, Will, BLAIR, Gordon, BASSETT, Richard and YOUNG, Paul

Published version

SAMREEN, Faiza, SIMM, Will, BLAIR, Gordon, BASSETT, Richard and YOUNG, Paul (2020). Models in the Cloud: Exploring Next Generation Environmental Software Systems. In: Environmental Software Systems. Data Science in Action. IFIP Advances in Information and Communication Technology . Springer, Cham.

Models in the Cloud: Exploring Next Generation Environmental Software Systems

Will Simm[1(✉)], Gordon Blair[1], Richard Bassett[2], Faiza Samreen[1], and Paul Young[2]

[1] School of Computing and Communications, Lancaster University, Lancaster, UK
w.simm@lancaster.ac.uk
[2] Lancaster Environment Centre, Lancaster University, Lancaster, UK
http://www.ensembleprojects.org

Abstract. There is growing interest in the application of the latest trends in computing and data science methods to improve environmental science. However we found the penetration of best practice from computing domains such as software engineering and cloud computing into supporting every day environmental science to be poor. We take from this work a real need to re-evaluate the complexity of software tools and bring these to the right level of abstraction for environmental scientists to be able to leverage the latest developments in computing. In the Models in the Cloud project, we look at the role of model driven engineering, software frameworks and cloud computing in achieving this abstraction. As a case study we deployed a complex weather model to the cloud and developed a collaborative notebook interface for orchestrating the deployment and analysis of results. We navigate relatively poor support for complex high performance computing in the cloud to develop abstractions from complexity in cloud deployment and model configuration. We found great potential in cloud computing to transform science by enabling models to leverage elastic, flexible computing infrastructure and support new ways to deliver collaborative and open science.

Keywords: Cloud computing · Environmental modelling · Data science

1 Introduction

Models in the Cloud is a three year project which sets out to explore the opportunity for a paradigm shift in the support offered by cloud computing for the execution of complex environmental models. Cloud computing is having a major and transformative impact on many areas of society, including in smart cities, eCommerce and eGovernment. Significant advances are being made in these areas and cloud computing is proving to be of significant benefit in minimising up-front investment, achieving economies of scale, supporting elasticity in the

I. N. Athanasiadis et al. (Eds.): ISESS 2020, IFIP AICT 554, pp. 216–227, 2020.
https://doi.org/10.1007/978-3-030-39815-6_21

underlying computational/storage capacity and out-sourcing of the infrastructure management [6].

These advances present a new opportunity for configuring computing on demand, and provide a fabric of services that can be configured and combined to support new modes of working. In visioning the future, consider that the code for a software model of the environment is constrained by the computer system it is to be deployed upon, traditionally a desktop computer or a high performance parallel computing environment. Here, the experiment is designed to fit the available computer. Now consider that cloud computing effectively allows you to design a computer around your experiment, removing constraints in a flexible, on demand, dynamic environment, then the potential opportunities for science are vast.

In his call to action for software engineers, Easterbrook [3] specifically identifies 'Computer-Supported Collaborative Science' as a fundamental way in which software engineering can contribute to addressing the grand challenge of climate change. He identified that through supporting earth system models with software engineering tools and techniques we can accelerate the process of getting scientific ideas into working code.

Blair et al. [2] laid out a roadmap of 10 challenges for research in data science of the natural environment. These research challenges cross-cut the themes of data acquisition, infrastructure, methods and policy making and are summarised here: (1) supporting a cultural shift towards more open and more collaborative science; (2) build on cloud computing, extending the levels of abstraction for the domain of science; (3) address complexity more fundamentally and explicitly, e.g. data science techniques to address extremes and emergent behaviours; (4) provide the tools to reify uncertainty and reason about cascading uncertainty; (5) seek adaptive techniques such as adaptive sampling or adaptive modelling driven by uncertainty considerations; (6) seek approaches that deal with epistemic uncertainty in environmental modeling and links with dealing with emergent behavior in complex and irreducible phenomena; (7) seek novel data science techniques, especially those that can make sense of the increasing complexity, variety and veracity of underlying environmental data; (8) seek innovations in modeling by combining process models with data-driven or stochastic modeling techniques; (9) incorporate sophisticated spatial and temporal reasoning, including reasoning across scales; and (10) discover new modes of working, methods and means of organization that enable new levels of cross-disciplinary collaboration.

In this work we primarily address challenge (2) and see running models in a collaborative, elastic cloud environment is fundamental to underpin many of the other challenges. In supporting new modes of science, it is critical to provide tools at the right level of abstraction for uptake by scientists in their work, and it is through the lens of abstraction that this project focuses to ascertain what is in place and what is required to leverage cloud computing.

Abstracting from underlying compute infrastructure and defining interfaces and datastores for environmental models will allow models to be connected and

run more efficiently, and to allow scientists to concentrate on the science, resulting in a better understanding of phenomena and uncertainties which would hopefully be reflected in better policy informed by results. This work forms a core pillar of technology enabling the concept of 'models of everywhere' - a vision to have models of everywhere, models of everything and models at all times, being constantly re-evaluated against the most current evidence [1].

In prior work we published a qualitative study with environmental modellers, and an implementation and associated feedback from modellers on a cloud deployment of a complex weather model [9]. This paper summarises that work, adds further analysis and describes the next steps taken.

The aim of this research is to determine if the principles and approaches exist to leverage cloud computing at a usable level of abstraction, and to inform the development of new tools and practices. This paper contributes the approach we took, the lessons learned from qualitative and experimental work, and adds reflections on lessons learnt to inform taking this work forward.

2 Methodology

As part of an interdisciplinary team of researchers [1], we undertake agile research. This involves relatively short cycles of "Plan - Act - Reflect", with each cycle informing the next. Supporting this work is an evolution of the Speedplay research methodology [4], rooted in participatory design that sees the computer scientists and environmental scientists in equal partnership, developing together. In this research we will go full circle, not just unravelling and understanding the opportunity, but designing and building technologies with end users embedded in the process.

We had end users as collaborators embedded throughout the project, the research team was comprised of experienced computer scientists and environmental modellers. We first did a qualitative study phase with environmental modellers from across the spectrum of the modelling community, this allowed us to up skill in understanding, shared language and get a handle on the challenges faced by modellers. It also helped focus the resources of the study.

We went onto an experimental phase where we began 'learning through doing' - exploring the application of abstraction technologies to this domain, putting models into cloud architectures and learning about the challenges in doing this both from computing and environmental science perspectives. This is the 'Act' part of the cycles in which feedback and reflection is embedded in directing the research.

3 Qualitative Phase

We undertook a study consisting of semi-structured interviews and demonstrations with a diverse group of five environmental scientists engaged in writing

[1] https://www.ensembleprojects.org/.

software to model a variety of environmental systems and processes. The purpose of the study was to gain an understanding of how environmental models are developed and deployed, and the computing tools and architectures which are used. These sessions also up skilled both computer and environmental scientists in their domain languages (e.g. cloud in the sky vs cloud computing, environmental model vs software model) and understanding of each other's domain. Further detail is in Simm et al. [9] but the findings are summarised and analysis extended here.

We selected our 5x participants from those working with a range of model complexities, from small scale statistical models of insect population to community developed global climate models. This was to enable us to gain an understanding of the challenges and opportunities at each scale. We grouped findings under Technical, Scientific and Human headings, for brevity just the main technical findings are summarised here [9]:

Computational Demands and Resources required by models varied from desktop machines to institutional high performance computing (HPC) facilities. When running models on HPC, in depth systems administration skills are required such as file system preparation, shell script writing and good command line familiarity. Data is input and output in flat files not data stores, and sorted and downloaded by FTP and shared by email. Projects are not costed to include projected compute costs or support form experienced software engineers.

In terms of **Computational Skills and Expertise** each of the participants had self-taught programming and systems admin without a formal education in computing. Code is frequently written as a monolith with little thought to code reuse or defining interfaces or structure.

Code Understanding - models are written in scientific languages such as Fortran, Matlab and R, often models are configured and run with Bash scripts and results are analysed using Python and R. Code is often not well commented and is difficult for others to understand, and Integrated Development Environments (IDEs) are not used often except for Matlab and R.

Version Control systems are not widely used, code (and data) being shared by email amongst collaborators rather than using version controlled repositories which leads to confusion about the 'latest version' and why changes are made and by whom.

Fault Tolerance and Resilience is good in large community models, allowing them to restart after a crash. However many models are difficult to debug and understand why crashes have occurred.

(In)efficiency arises from needing to download large data files (sometimes 10's of GiB) to extract the small amount of data for focus as data is stored in flat file format rather than query-able data stores. Code reuse is poor, languages like R do not enforce object orientated styles, and the choice of language is sometimes sub optimal, e.g. R is good at statistical modelling but often used for many other things.

Compatibility - the interfaces for running models are not abstracted from the models - in depth knowledge of the model is required to configure,

parameterise and run models. Data input and output is usually flat file CSV, netCDF and Excel spreadsheet format.

This phase of embedding computer scientists in the environmental modelling domain allowed us to grasp the challenges faced by environmental modellers and understand what appropriate abstractions over computing complexity might be. The main finding of this phase of work from the perspective of this project is that the code of environmental models is deeply entwined with the architecture of the conventional computing systems and the working practices of environmental scientists. There is a great desire to move the science on from this complexity, which we believe could be achieved through abstraction. Indeed, to foster greater understanding of uncertainty in models of environmental systems, models runs may need to be more efficient and run many more times in many more places with approaches such as models of everywhere [1]. Software engineering achieves the separation of concerns through abstraction using modularity, frameworks and defined interfaces; these practices are not often seen in day to day environmental modelling, possibly due to a lack of SE training, but mainly due to the absence of tools at a usable level of abstraction for this domain.

The offline mode of working makes sharing and collaborating difficult, we found our participants indicated closer collaboration would be desirable for science. Code is not released to the community because it is not considered robust or efficient; usually this is because of concerns about input data, misunderstood bugs or the specifics of a system such as configuration of dependencies. A software engineer can see much of this could be resolved through abstraction (e.g. by using pre-built data cleaning libraries rather than unique data cleaning code), changes to code writing style to make debugging easier, and using dependency management tools.

3.1 Opportunities for Abstraction

Computational abstraction alone will not address the problems identified in the qualitative work, social factors are important as well, but beyond the scope of this paper. However we believe abstracting from computational complexity can play a significant role at multiple levels:

1. At the **model code** level: Software frameworks and robust software engineering practices such as defined interfaces and modular code architectures would allow code reuse, reduce dev time and reduce the need for debugging.
2. At the **infrastructure** level: By removing the reliance on the desktop and HPC there is the opportunity to deploy models to new and flexible architectures provided by cloud computing. There are multiple additional benefits here of opening access to experiments for collaborative science, and access to data stores and data science tools already present in cloud offerings.
3. At the **model configuration** level: Complex models have interdependence across a huge volume of configurations, the interaction of which needs to be understood, and sometimes the same value needs to be specified in multiple places.

4. At the **model parameterisation** level: There is a range of methods to parameterise models, often involving multiple model runs before settling on a suitable parameter set.

The first opportunity is perhaps symptomatic of a lack of core software engineering skills within the community writing software models. It cannot be addressed without significant re-engineering of models; our participants pledged to learn about best practice but software frameworks for developing and running models would support this.

The second is one area in which we chose to focus our efforts by deploying models to the cloud to explore the opportunity, and look for tools at the right level of abstraction to enable an environmental modeller to configure and deploy a suitable infrastructure for their experiment.

The third is another area we looked at, and felt their was an opportunity for techniques such as Model Driven Engineering (MDE) to build a software model of the complexities in configuration of environmental models.

The fourth we have considered and propose that techniques from AI and machine learning can reduce the parameter space and help the scientist to select the most suitable parameters for their experiment, but only if models exist in an on demand, flexible cloud infrastructure supported by fit for purpose data stores and powerful data science machines.

4 Experimental Phase

This phase describes approaches to abstracting over (i) the complexity in the configuration and deployment of an environmental model (abstraction 3 above) and (ii) the complexity in cloud deployment of the model (abstraction 2 above). We created a tool for the configuration experiments, and a cloud deployment of the Weather Research and Forecasting (WRF) model. Here, we raise the level of abstraction through reducing the complexity of configuration and deployment, leveraging the on demand scale-ability of cloud architecture.

4.1 WRF

The Weather Research and Forecasting (WRF) model [10] is a large community-based endeavour (around 40,000 users), supported by the National Center for Atmospheric Research (NCAR). The model is primarily used for atmospheric research and forecasting across a wide range of scales (thousands of kilometres to meters). The diverse range of extensively validated science WRF can simulate includes regional climate, air quality, urban heat islands, hurricanes, forest fires, and flooding through coupling with hydrological models.

WRF is chosen as a case study here for the following reasons: (i) WRF installation is viewed as a barrier to use; (ii) cloud resources will enable WRF users to conduct simulations beyond current capability [7]; (iii) WRFs open-source nature and portability; and (iv) benefits will impact WRFs large community user base.

4.2 Configuration and Collaboration

Model Driven Engineering (MDE) is an approach to managing complexity in software systems and to capture domain knowledge effectively [8]. Domain knowledge is captured in a software model of the system, this model is configured using Domain Specific Languages (DSLs) that relate to the application domain and potentially the underlying platform features. Code is then generated by the model using transformation approaches to configure and deploy the system being modelled. This approach has been applied successfully in a variety of areas [8] including in industry settings [5].

In this work we investigated using this approach to develop DSLs to allow scientists to describe an experiment, with the underlying software model managing the generation of code to configure the environmental model appropriately, deploying the model to appropriate flexible cloud infrastructure and returning results. In exploring the tools available, we found they are not able to support such a vision at the current state of readiness. There are so many complexities and flexibility required in configuring models like WRF for the many different uses, that to hard code a set of rules into a DSL was not an approach that was likely to be successful.

Instead we decided to use a general purpose language to manage the configuration and deployment, and proposed embedding a future 'learning' approach that would match experimental configurations to infrastructures, and would be able to recommend an appropriate architecture for an experimental configuration. In our qualitative phase we found our participants were familiar with languages such as R and Python, often used to process data and produce visualisations.

In taking this MDE-lite approach, we produced a Python object based model of a WRF experiment configuration, that allows configuration using standard Python constructs. The Python package f90nml[2] allows the object to generate Fortran 90 namelist files that are used to configure WRF, these are auto-uploaded to the WRF instance for deployment. This provides a layer of abstraction over the skills required to login to an instance, navigate the file system, and edit the Fortran namelist file without introducing errors. No configuration dependencies are managed in this first iteration, but this can be added to the Python model in future.

Code notebooks such as Jupyter[3] are now widely used in the data science community to collaborate and annotate data analysis. They are online environments, often hosted in the cloud (but not necessarily publicly available) that allow code to be edited and run with multiple collaborators. They allow mixed mode documentation, with in line code and output visualisation from many modern programming languages, including Python and R. They are able to pull in data from outside sources and write out data to attached cloud native stores or via API.

[2] https://pypi.org/project/f90nml/.
[3] https://jupyter.org/.

For our purposes a Jupyter code notebook provides an ideal space for environmental scientists to control modelling experiments. Whilst it would not be possible to run a complex model such as WRF in the notebook, it is possible to configure a run experiments via system APIs. The model can be configured using our Python object based configuration tool, and the experiment run outside of the notebook (either on the same machine or an external 'cloud burst') with output data returned to the notebook for collaborative analysis. The notebook allows the experiment to be documented (supporting scientific reproducibility) and shared, extending the reach of the experiment.

4.3 Cloud Deployment of WRF

We built a scripted cloud deployment of WRF, and evaluated usability, performance and cost metrics. This installation script dealt with dependencies and the configuration and compilation required to run WRF on the Microsoft Azure cloud platform. The script is automated as far as possible, however in a number of places unavoidable user input is requested from the WRF installer.

The architecture of the WRF model itself is complex, described by [10]. Our standard cloud configuration consisted of a Message Passing Interface (MPI) supported cluster of 9 standard compute nodes from the Microsoft Azure Dsv3-series each having a 3.2 GHz Intel Xeon E5-2673 v4 (Broadwell) processor. One node is a master node taking care of all the compilation and providing a means of sharing the storage and computation with all the nodes.

We used a predefined image of Ubuntu Server 16.06 LTS for each of the cluster machines and each node has 8 processors with 32 GiB RAM and temporary storage of 64 GiB that is considered a secondary storage for each compute node. We used the GNU Fortran and GCC compilers. The cluster provides primary storage of 100 GiB shared amongst nodes via the Network File System (NFS). The shared location contains all the simulation related input/output data and files required for WRF configuration as well as compilation. All the cluster nodes and storage are deployed in Western Europe under one secure virtual network and have friction-less access to enable data sharing and execution of MPI jobs.

An expert user group of 6 regular WRF users agreed that our automated WRF deployment successfully abstracted over the major hurdle of initially installing the model. They helped to develop the use cases that we used in developing the work: (A) removing barriers to entry for new users, allowing them to immediately run experiments; (B) users wanting to run the model in a standard way to feed results into other models; and (C) power users wishing to deploy massively in parallel and without waiting for institutional HPC queue times [9].

4.4 Mechanisms for Cloud Computing Configuration

To build on this scripted installation, we investigated key mechanisms to abstract over the complexities of configuring compute architecture, by exploring different modes of cloud deployment. We wanted to retain as much flexibility in the deployment whilst retaining usability for our use cases.

Portal Configuration. The scripted installation described above requires the manual configuration of the cluster from the Azure web portal, or defined in code using their Infrastructure As Code (IAC) offering, which are general purpose interfaces and require a deep understanding of the desired infrastructure. Because of this knowledge required, it is perhaps not a suitable interface for our use cases except perhaps (C) the power user who may have an understanding of the relative merits of different cluster configurations and be able to fine tune to their experiment manually. Since the WRF installation is not fully automated, it still requires user interaction.

Containerisation is a mechanism whereby the software environment including dependencies for a particular application is defined in code. These containers are infrastructure agnostic, so can be deployed to any suitable provider. However the MPI architecture of the WRF model makes it unsuitable for containerisation - MPI is the mechanism by which messages are transferred between nodes, and we found little support for this in existing technologies. This situation is changing, with a number of providers beginning to offer support for MPI [4].

LibCloud is a Python library for interacting with many cloud providers, abstracting from their specific IAC offering. This allows a provider agnostic configuration of a cloud infrastructure from a notebook, however it is again general purpose and does not encompass every offering from every provider. It is designed for the computer scientist, and does not really abstract from the complexity of deploying a cloud computing system, just from the differences that individual cloud providers might have for their standard machines. It also does not get around the user interaction required for installing WRF so perhaps only suitable for use case (C).

Infrastructure As A Service (IAAS) allows the replication of predefined and configured machine images, allowing standard node with WRF installed to be instantiated on demand. In Azure this can be in their machine image library for anyone to use, and other providers have similar facilities.

In summary, the IAAS approach was selected as being most suitable for our WRF cloud deployment as it means deployment is instantaneous with no user interaction required in getting a system running WRF running, so good for use cases (A) and (B). However this reduces flexibility for (C) in terms of virtual machine specification as it is tied to a specific standard machine type offered by our cloud provider. Without re-engineering the WRF installer, or WRF itself it was not possible at this time to create a cloud-native, provider agnostic system without user input in the installation.

4.5 Experimental System

The IAAS approach whilst reducing ultimate flexibility in infrastructure, allowed us to explore how WRF might be configured and deployed from a notebook environment. We built a demonstrator using the Azure SDK for Python to configure and deploy a WRF cluster from within a Jupyter notebook running on an

[4] https://www.stackhpc.com/the-state-of-hpc-containers.html.

instance of Azure data science machine and so far a number of test experiments have been run.

Ongoing Work: The next stage in developing this system is to integrate the WRF namelist configuration tool to the same notebook, and return results as an attached data store to the machine. The same Jupyter notebook can then be used to sort output data and prepare data visualizations. In this way the whole experiment can be orchestrated through a single collaborative notebook interface, which is version controlled and can be archived easily. Figure 1 shows this visually.

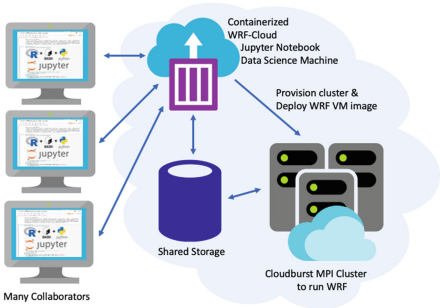

Fig. 1. WRF-cloud system architecture

5 Reflections

The key successes of this project were in using the WRF case study to navigate available technologies to create useful abstractions not only to enable the scientist to focus on science, but in support of Easterbrook's vision of the next generation of computer supported collaborative science [3]. A vision where models form part of a service fabric of technologies that can be recombined and run an unlimited number of times to explore the specifics of place and rationalise about uncertainty. The work underpins Blair's ten challenges to environmental data science community [1] and leverages the elasticity and flexibility of cloud computing to provide on demand, scale-able computing.

Adapting WRF to a cloud computing infrastructure and attempting to leverage the latest advances was bound to face some challenges. WRF has been developed over decades and consists of about half a million lines of Fortran code. Fortran is especially good at large scale numerical simulation, however it is not a typically supported language in the modern software industry and hence cloud computing platforms. The MPI libraries for parallelising code were written for Fortran and C++, and support for these is low in cloud native technologies such as containerisation, although this is changing as more cloud providers consider supporting scientific applications. Containerisation allows complete abstraction from underlying compute architecture, and would make deployment on new generations of machines possible. As such it was not possible during the project to leverage these cutting-edge technologies to run WRF, and we had to step back to an IAAS approach of using machine images to explore our ideas.

Installing and compiling WRF on new machines is a chore, and requires an advanced computing skill set. Our demonstrator of an auto-install of WRF to the cloud was an eye-opener for many of our participant scientists. The ability to simply spin up a virtual machine pre-installed with WRF is beginning to be used in teaching labs, instead of requiring students to go through the long error-prone process. This was a successful exercise in abstraction from complexity.

We found the tools developed by the MDE research community to be unable to capture the complexity of WRF deployment without reducing the flexibility required by the wide range of use cases - they required hard coding a set of configurations. However we designed a Python tool to allow the scientist to retain full control of the experimental configuration, without risking errors in the generation of a Fortran namelist configuration file, that in future could manage configuration dependencies and abstract from configuration complexities.

We added further abstraction from the cloud provider's interface by bringing these tools together into a now becoming familiar Jupyter notebook, writing code to deploy a WRF system from within the notebook, which can then also be used to analyse results in a collaborative environment that describes the whole experiment. An obstacle we are facing in completing this vision is the poor support of WRF's netCDF output files, which are designed for flat file stores. Other groups are working on this, for example Unidata at UCAR[5].

6 Conclusion

In this project we aimed to find out if the principles and approaches exist to leverage cloud computing at a usable level of abstraction in environmental modelling, and we found we were able to make usable abstractions from complexity in deployment of the WRF model to cloud, and the configuration of WRF. We found a notebook based approach supports new practices in collaborative and open science through the ability to describe and run an experiment in a single notebook document that can be shared and examined by others. Our findings can be generalised to other complex models, specifically MPI based models but learning is applicable to many others.

The abstractions proposed and explored through this project mean that WRF could form part of a cloud service fabric consisting of many models, data stores and data science services that could run as many times and in as many combinations as desired. Add in cloud based machine learning facilities and a system could propose its own infrastructures based on the experiment to be run, as well as help to reduce the workload of scientists in terms of reducing parameter spaces when parameterising models. These abstractions also have potential to be significant when operationalising code which needs to run continuously and effectively, allowing code to be run on diverse and shared architectures.

Future work will involve 'closing the loop' by returning outputs from WRF to the notebook by connecting output data to a cloud native data store, thus enabling the analysis of output within the same notebook, but also powerfully interfacing output to other models as part of the aforementioned service fabric. This work primarily focused on technical factors, but equally there are many social factors at play in defining the software and hardware architectures in use in environmental science. Future work could include exploring the barriers these present to the take up of new computing technologies.

[5] https://www.unidata.ucar.edu/blogs/news/entry/netcdf-and-native-cloud-storage.

Acknowledgements. Thanks to the wider Ensemble team (https://www.ensembleprojects.org/) for their support. This work is supported by "Models in the Cloud: Generative Software Frameworks to Support the Execution of Environmental Models in the Cloud" EPSRC: EP/N027736/1 and a Microsoft Azure AI for Earth grant.

References

1. Blair, G.S., et al.: Models of everywhere revisited: a technological perspective. Environ. Model. Softw. **122**, 104521 (2019). https://doi.org/10.1016/j.envsoft.2019.104521
2. Blair, G.S., et al.: Data science of the natural environment: a research roadmap. Front. Environ. Sci. **7**, 121 (2019). https://doi.org/10.3389/fenvs.2019.00121
3. Easterbrook, S.M.: Climate change: a grand software challenge. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER 2010, pp. 99–104. ACM, New York (2010). https://doi.org/10.1145/1882362.1882383
4. Ferrario, M.A., Simm, W., Newman, P., Forshaw, S., Whittle, J.: Software engineering for 'social good': integrating action research, participatory design, and agile development. In: Companion Proceedings of the 36th International Conference on Software Engineering, ICSE Companion 2014, pp. 520–523. ACM, New York (2014). https://doi.org/10.1145/2591062.2591121
5. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, pp. 471–480. ACM, New York (2011). https://doi.org/10.1145/1985793.1985858
6. Johnson, R.E.: Frameworks = (components + patterns). Commun. ACM **40**(10), 39–42 (1997). https://doi.org/10.1145/262793.262799
7. Powers, J.G., et al.: The weather research and forecasting model: overview, system efforts, and future directions. Bull. Am. Meteorol. Soc. **98**(8), 1717–1737 (2017). https://doi.org/10.1175/BAMS-D-15-00308.1
8. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. Computer **39**(2), 25–31 (2006). https://doi.org/10.1109/MC.2006.58
9. Simm, W.A., et al.: SE in ES: opportunities for software engineering and cloud computing in environmental science. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS 2018, pp. 61–70. ACM, New York (2018). https://doi.org/10.1145/3183428.3183430
10. Skamarock, W., et al.: A description of the advanced research WRF version 3 (2008). https://doi.org/10.5065/D68S4MVH