Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

New Directions and Software Tools Within the Process Systems Engineering Ecosystem

S. Burroughs[a], B. Lincoln[b], A. Adeel[a], I. Severinsen[c], A. Lee[d,e], O. Amusat[f], D. Gunter[f], B. Nicholson[g], M. Apperley[a], B. Young[c], J. Siirola[g], and T. G. Walmsle[b*]

[a] Ahuora – Centre for Smart Energy Systems, Department of Software Engineering, University of Waikato, Hamilton 3240, New Zealand
[b] Ahuora – Centre for Smart Energy Systems, School of Engineering, University of Waikato, Hamilton 3240, New Zealand
[c] Department of Chemical and Materials Engineering, University of Auckland, 5 Grafton Road, Auckland, 1010, New Zealand
[d] National Energy Technology Laboratory, Pittsburgh, PA 15236, United States of America
[e] NETL Support Contractor, Pittsburgh, PA 15236, USA
[f] Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States of America
[g] Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, 87185, United States of America
* tim.walmsley@waikato.ac.nz

ABSTRACT

Process Systems Engineering (PSE) provides the advanced conceptual framework and software tools to formulate and optimise well-considered integrated solutions that could accelerate the sustainability transition within the industrial sector. The landscape of advanced PSE is poised to undertake a considerable transformation with the rise in popularity of open-source and script-based software platforms with predictive modelling capabilities based on modern mathematical optimization techniques. This paper highlights three leading equation-based platforms—IDAES[1], Modelica[2], and GEKKO[3] — that are increasingly utilised for the modelling, simulation, and optimisation of complex systems within the advanced PSE domain, alongside the strengths and limitations of each approach. Following this, we present a framework through which emerging techniques within the domain of Software Engineering could be leveraged to address these limitations, with a vision of improving the accessibility and flexibility of complex modelling tools for industrial partners.

Keywords: Pyomo, Industry 4.0, Process Design, Simulation, Process Synthesis

INTRODUCTION

Process Systems Engineering (PSE) emphasises the ability to appropriately model, analyse, and optimise systems that relate to the design, retrofit, and operation of processes throughout its life cycle. Chemical plants often comprise multiple interacting processes, each of which involves complex, non-linear dynamics. Accurate models are necessary to establish the economic viability and chemical feasibility of processes of a plant and any changes. This is particularly relevant when considering the increase in global regulation requiring adherence to specific targets for greenhouse gas emissions and pollution reduction and the integration of renewable energy.

From the inception of PSE as a field of study, mathematical models often based on first-principles relationships have been developed to capture the complex interactions within unit operations and between components of plants. Relationships can also be data-driven using classical regression or modern machine learning methods, such as physics-informed neural networks (PINN) [1-2]. These models form systems of equations involving

[1] https://idaes.org/
[2] https://modelica.org/
[3] https://machinelearning.byu.edu/

integer variables, continuous variables, convex/non-convex relationships, and so on, that can be simultaneously solved to find the simulated solution (DOF=0) or optimal (and near-optimal) solution (DOF>0).

The PSE modelling approach is now well established and continues to progress with new advancements. Of particular note, several solvers have been developed specifically to solve formulated mathematical modelling problems, with both open source (e.g., IPOPT) and commercial solutions (e.g., BARON, Gurobi) accessible to researchers and industry [3]. These solvers represent verified programmes that enable practitioners to focus more on developing representative models, and less on the mechanics of how they are solved.

In a further step, Algebraic Modelling Languages (AML)s have also emerged to facilitate the faster development of models through systematic methods for expressing the equations using familiar mathematical notion and language. GAMS in the 1970's pioneered this level of abstraction, creating a comprehensive framework for capturing algebraic expressions. Similar open-source libraries built within modern programming languages are now seeing an upswing in use [4]. Of particular focus within this paper are the IDAES, Modelica, and GEKKO frameworks, each of which adopts an object-oriented approach to define properties and behaviours of modelled components, although with differing levels of abstraction [4-8]. To this end, IDAES and GEKKO are developed as open-source packages within Python, with components following a class structure that allows for the complex development of a system through the creation of modelled objects. In contrast, Modelica utilises a stand-alone language that follows the Modelica language specification.

The advance of AMLs has increased the accessibility of mathematical modelling, as demonstrated by the broad range of libraries developed within the Modelica community. However, broad adoption of such approaches is still limited by learning curve required to gain familiarity with the axioms and syntax of each respective framework, as well as broader experience in formulating models well suited to optimisation and proficiency with programming languages. Models are also constrained by the computational resources that are available to the solver.

At the opposite end of the abstraction spectrum lies process simulators such as Aspen Plus, Aspen HYSYS, gPROMS, CadSIM, DW Sim, and many more. Each of these simulators provide a "drag and drop" approach to adding/removing/modifying operations to a flowsheet while in the background models are being generated and solved. Advanced users can customise nearly all aspects of the flowsheet but not the mathematical model. Furthermore, once a complex flowsheet is built, there is no possibility to export the mathematical model to form a light-weight representation that could be solved independently.

In this paper, we discuss the IDAES, Modelica, and GEKKO frameworks and propose that advances within the domain of Software Engineering can be leveraged to better address current deficiencies in abstraction, computational resourcing, and modern software application development practices. Nearly all simulators predate modern software engineering practices (e.g., only DW Sim has an interactive web-based platform). As a result, their adoption of such practices and advances are constrained. To this end, we present a framework through which complex modelling capabilities can be enhanced and integrated into a unified and accessible software tool that follows a modular architecture, allowing for the integration of individual capabilities, developed as stand-alone modules, into a central system. This framework has a focus on User Experience (UX) based design, with data structures that are obscured from the user behind a UI, allowing for information to be stored in a standardised form that can be filtered for use across multiple analysis modules. Significantly, this framework leverages containerisation and cloud orchestration technologies to improve accessibility and performance by removing conflicts that arise when producing software solutions for heterogeneous hardware configurations and Operating Systems.

MATHEMATICAL OPTIMISATION FRAMEWORKS

The frameworks explored within this paper are free-to-use and have established their capacity for the modelling, analysis, and optimisation of complex systems via the formulation of mathematical models created via an abstract AML syntax that are then passed to open source and commercial solvers. While the specifics of how this is accomplished differs between frameworks, as does the domain focus and accessibility model, there are strengths and limitations that are common across all approaches and speak to the broader need to improve accessibility and extendibility such that the benefits of mathematical models can be fully leveraged by those within the chemical process engineering domain.

IDAES and GEKKO are standalone packages, with modelling components developed within the Python programming language. GEKKO utilises its own AML formulation to achieve this, while the IDAES Core Modelling Framework (IDAES CMF) is built on top of Pyomo, a general-purpose Python AML. As Python packages, the creation of models within GEKKO and IDAES is achieved via the instantiation of defined classes as modelling objects. Models are specified as high level objects, from which complex behaviours can be captured through the incorporation of defined constants, parameters, and variables.

The intra-connectivity of models is established by relating model components through connections, intermediates, and equations in the case of GEKKO and arcs, expressions, and constraints in the case of IDAES. Once defined, models are optimised for a given objective function [9-11,6,8].

Modelica in contrast utilises its own language with a standardised approach for defining variables and behaviour. There are inherent trade-offs between these approaches. On the one hand, a proprietary syntax like that developed for tools within the Modelica framework ensures the consistency of modelled components and the compatibility of analysis techniques. Conversely, the Python based approach of IDAES and GEKKO presents a lower barrier to entry for users who are already familiar with the underlying syntax and concepts within Python as a higher level programming language, while also retaining the ability to integrate broader functionality within the Python ecosystem into the modelling process.

A key distinguishing feature between the frameworks is the primary use cases that they support. GEKKO is intended primarily for the dynamic optimisation and control of linear and non-linear systems, with an APMonitor executable providing solving capabilities. Specific libraries can be imported into GEKKO to extend this base functionality to incorporate Deep Learning and Machine Learning, as well as a chemical library that defines a flowsheet class, chemical compounds and thermodynamic properties. The relationship between GEKKO and its chemical library extension is somewhat analogous to that of IDAES and Pyomo, with Pyomo as a domain agnostic optimisation package and IDAES as a standalone library that leverages Pyomo capabilities to model problems within the domain of chemical engineering [12]. A core difference is that as a standalone modelling package, IDAES is much more feature-rich and comprehensive, with the core library providing capabilities to model unit operations and properties within chemical processes [8]. Specific libraries are also available for domain specific processes such as power generation, materials processing, and water purification, as well as further modelling extensions that support surrogate modelling and parameter estimation approaches and the incorporation of deep learning models trained through standard frameworks such as PyTorch and Keras [13]. As a well-established stand-alone modelling language, the ecosystem that has developed around Modelica is broad. Modelica has thus emerged as a framework for use in simulation of multi-domain dynamic systems, with libraries adhering to the Modelica specification released under both open source and commercial licences. This reflects the core strength of Modelica, being the ability to integrate dynamics across multiple domains into a single model for a given system [5,7,14].

While there are specific applications for which each framework is well suited - real-time optimisation and control for GEKKO, chemical process systems for IDAES, and multi-domain system modelling for Modelica – there are broader strengths and limitations that are applicable to all EOM frameworks. An obvious strength is the comparative performance of mathematical modelling approaches vs traditional simulation, due largely to the commercial and open source solvers that are available for computing solutions for well formulated models [3]. This is particularly relevant when considering the non-linear relationship between model complexity and solution space, for which the solving performance of traditional first principles based simulation engines becomes a clear bottleneck [15]. More significantly for chemical process engineering is the degree of customisation and extensibility that is available to an end user through mathematical modelling. Within traditional simulation engines, users are generally unable to access the expressions that govern the behaviour of modelled components. Furthermore, it is challenging to define and integrate new components. In contrast, using an AML it is possible to build new unit operations or property packages to suit the individual needs of a modelled system. This is exemplified within IDAES through their modular property package and reaction frameworks, as well as their generic *UnitModelBlockData* class, each of which can be implemented with custom configurations. In cases where dynamics are unknown for a given component within a system, surrogate models can be developed through machine learning techniques to capture the desired behaviours [16].

Common underlying limitations for all frameworks can broadly be captured under the scope of usability, accessibility, and extendibility. While the adoption of programming language-like syntax has greatly improved the usability of EOM frameworks, users are still faced with a steep learning curve to master the syntax and implementation of each AML, as well as the broader concepts and logic associated with the use of programming languages and environments. This is particularly challenging when considering IDAES wherein multiple layers of inheritance are implemented on top of base Pyomo classes, requiring a high degree of familiarity with the underlying source code to gain the full benefits. Furthermore, upon mastering the syntax and concepts of each AML, there is still a significant effort required to ensure that models are adequately specified with the correct number of degrees of freedom and can be initialised [17]. This limitation is recognised, with frameworks implementing approaches to abstract away the complexity of the underlying AML behind a User Interface (UI). In the case of GEKKO and IDAES this is limited to visualising and manipulating flowsheets after they have already been defined programmatically, although IDAES has also taken steps to provide users with Command Line Interface (CLI) diagnostics [17]. Modelica has open source and commercial tools to

provide a UI, however the open source UI of OpenModelica is somewhat unintuitive and outdated, while access to commercial versions is naturally constrained by cost. This issue of commercial vs open source development is also a core consideration for accessibility and extendibility. While the Modelica language specification has contributed to the development of an ecosystem of analysis packages, the creation of multi-domain models is predicated on the willingness and ability of the end user to pay for each of the commercial libraries that they require. Furthermore, to gain the best user experience when using these libraries, further costs must be borne to access relevant commercially developed UIs. Consequently, while the established nature of Modelica as a framework and its standardisation as an AML contributes to its extendibility, accessibility and collaboration is limited by its inherent commercial realities. GEKKO and IDAES are less established as frameworks but are entirely open source in their development, meaning that functionality is well suited to customisation and extension for user specific cases. Furthermore, the Python based nature of GEKKO and IDAES means that they are inherently compatible with other libraries that exist within the Python ecosystem, which is immense.

As a result of these observations, there is clear scope for the development of a generalised and intuitive, user experience-driven platform that is framework agnostic. Such an approach would provide the usability benefits of traditional process simulators, while retaining the extendibility and performance provided by mathematical frameworks. There are clear benefits that would result from such development, foremost of which is the ability to leverage the strengths of each framework and compare the resultant analysis for a given model. This benefit is not limited to EOM frameworks, as data within such a platform could be stored in a traditional database rather than existing explicitly within the context of any individual AML. Consequently, further forms of analysis that are not EOM based (such as Pinch Analysis or Heat Exchanger Network synthesis) could be performed on the same data, allowing for greater access to and integration of insights from such techniques. This would be of great benefit to chemical engineers seeking to design or analyse processes, as it would remove a lot of the time that is spent manually creating models across different frameworks and updating each with the results of another, as is the case in the current status quo.

FRAMEWORK FOR A COMPREHENSIVE PLATFORM

There are two broad classes of problems that present bottlenecks to the uptake of robust equation-based modelling techniques: usability and compatibility. The modelling frameworks explored within this paper are all reliant on the user's familiarity with programming languages and development environments. Furthermore, each framework has its own respective axioms and syntaxes, with sophisticated capabilities inaccessible without significant legwork on the part of the user. This has contributed to the lack of extensive adoption of such frameworks outside of the research space.

Compatibility challenges extend beyond the scope of equation-based modelling, reflecting an issue that is more fundamental to modelling and optimisation in general: multiple forms of analysis and degrees of abstraction are often desirable, and it is difficult to integrate the insights from one approach into the formulation of another without considerable effort on the part of the end user. We propose a solution to these issues in the form of a comprehensive platform developed along software engineering methodologies.

Addressing Usability Concerns

The fundamental issue at the core of usability challenges is the barrier to entry presented by the complex and abstract nature of modelling frameworks and syntax, as well as the inherent challenges in debugging models and interpreting results. We propose that to address this, a UX-focused UI should be developed independently from any single modelling framework. Interfaces should also be designed with a view of integrating further analysis and modelling functionality in the future, ensuring that the final solution does not suffer from the clutter that is associated with frequent retroactive additions. To this end, the design process should begin with initial consultation with intended users to elicit functional requirements and identify pain points with existing approaches, from which multiple draft prototypes should be developed to capture functionality and workflows. These steps should be repeated to select and refine the implementation, following which components should be developed from final prototypes.

The separation of concerns between front-end UI and backend data structures and modelling logic is part of a higher-level modular architecture that is a core feature of this framework. The modular components fall into the following three categories: front-end UIs, analysis, and data structures and APIs. UIs and analysis modules are developed independently of each other. The Front-end should have a focus on intuitive design and data interpretation, while analysis modules should be standalone packages. The link between front and back-end elements is established through a final module, which defines data structures and APIs.

Data structures can capture an exhaustive amount of information about a given system component beyond the scope of any individual modelling framework or form of analysis and can be informed by live sensor data. As further functionality is integrated into the platform,

additional properties can be added to the existing data structures as required, with new APIs managing the communication between the new module and the database. It is thus possible to retroactively integrate new features or properties without breaking existing models. Model data can also be filtered to display different abstractions or engage in multiple analysis functionalities, with each of them manipulating and modifying a shared data source, providing the user with a level of control over the detail and complexity of the UI, allowing for use by both basic and advanced users.

A modular architecture is further beneficial for maintaining code as modifications to the functionality of one module will not break the functionality of another. This supports a 'plug and play' approach for developing additional functionality. Each module is developed as a standalone package with a defined set of required parameters and accompanying tests to verify functionality. Database migrations are applied to integrate new properties into models, with UI components developed to manage interactions with the new module and an API layer developed to facilitate the communication of data between the database and the module.

Leveraging Cloud Computing Strategies

The outlined modular architecture is well suited to deployment across a distributed system in the form of a microservice architecture [18]. This is achieved through a containerisation approach, which packages a process alongside its dependencies into a lightweight container image. Container images also define an Operating System (OS) that is required for the process, enabling processes with different OS requirements to run on the same system. In a microservice architecture, a complex application is separated into its constituent modular elements, each of which is then packaged as a containerised process. Containerisation has several benefits compared to more traditional monolithic application architectures [19]:

- Resource utilisation can be handled at a more granular level as each containerised process has its own respective allocations.
- The number of replicas for a containerised process can be increased or decreased. Requests can be handled by any replica.
- The impact of any given process failing is greatly reduced. functional replicas will automatically take up the jobs assigned to a failed container. Furthermore, containers can be quickly recreated as needed.

Kubernetes is a container orchestration platform that facilitates the management of containerised applications across a cluster of networked computational resources by encapsulating them within a higher-level "pod". This is achieved through a self-adaptive approach that leverages a control approach to match the running state of the application to an expected state. In practice, this means that Kubernetes allows for self-healing behaviour, wherein failed processes are automatically restarted, and autoscaling, where the number of container replicas for a given process will automatically increase and decrease in response to dynamic demand. A microservice architecture deployed to a Kubernetes cluster is visualised within figure 1.

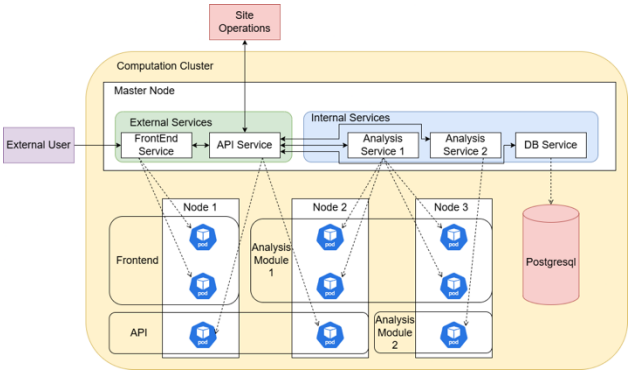

Figure 1: Microservice architecture deployed on Kubernetes.

As data is created or modified within the front-end, information is created or updated within the linked database. Upon engaging analysis functionalities, API calls extract relevant information from the database and send it to the relevant module. The results of the analysis are then returned to the database and displayed within the associated UI component. All interactions between a user and data within the platform are managed via UI components, with the actual data handling managed through APIs.

The integration of Kubernetes allows for an economic model that supports simultaneous commercial and open access releases, without limiting accessibility to core functionality. This can be achieved through the release the end software as a standalone containerised application that can be run on a local machine, or the release of standalone modules that could be integrated into other custom software. Conversely, more advanced users that wish to collaborate with others over shared models or require additional access to computational resources for parallel analysis could pay for access to the Kubernetes version, which is hosted externally. Such a financial model would support the continued development and maintenance of the software, while also ensuring that libraries remain open source and core capabilities are accessible.

Key Directions for Future Research and Development

To adequately fulfil the vision that our presented

framework represents, the software platform must achieve the following outcomes:

- UI development should center around user experience, with primary consideration given to intuitive workflows.
- The platform should be agnostic with regard to modelling frameworks. Backend analysis modules should be able to automatically generate, export, and solve models using a framework chosen by the user.
- It should be freely available for research and development, with modules released as standalone, open source packages.
- It should leverage cloud computing paradigms such as containerisation to achieve optimal performance through scaling actions and avoid conflicts of OS or hardware configurations.

We envisage that such a platform would naturally fill a gap within the existing modelling ecosystem, improving accessibility to such tools and potentially extending in future to include multi-domain level analysis through the integration of additional modules.

CONCLUSIONS AND FUTURE VISION

This paper has outlined the development of abstracted modelling within the process engineering domain, from initial verbose mathematical models to modern process simulation software and mathematical modelling frameworks. Three mathematical frameworks were chosen for their relevance to the chemical process engineering domain, as well as their individual implementation. Strengths and weaknesses for each framework were outlined, establishing advantages of mathematical modelling for performance and customisation while also highlighting the usability concerns associated with the steep learning curves required to gain proficiency. We then presented a framework through which the strengths of traditional process simulators can be combined with the powerful functionality of equation-oriented modelling via the adoption of modern software engineering concepts and methodologies for the development of a comprehensive software solution that allows for the integration of independent modules behind a single UI.

The vision we present for a software solution that follows the implementation path of our proposed framework is that of a centralised, comprehensive platform that serves as an intuitive interface through which users can access complex functionality. This platform would greatly contribute to the accessibility of advanced modelling capabilities by removing the requirements of familiarity with AML syntaxes and programming environments. Furthermore, the abstract nature of the data structures within this platform would also contribute to the integration of multiple disparate forms of analysis into a single shared knowledge base, greatly increasing the ability for engineers to integrate insights from multiple sources into a single model.

ACKNOWLEDGEMENTS

This research has been supported by the program “Ahuora: Delivering sustainable industry through smart process heat decarbonisation”, an Advanced Energy Technology Platform, funded by the Aotearoa New Zealand Ministry of Business, Innovation and Employment.

This project was funded by the United States Department of Energy, National Energy Technology Laboratory, in part, through a site support contract. Neither the United States Government nor any agency thereof, nor any of their employees, nor the support contractor, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Contract No. DEAC02-05CH11231.

REFERENCES

1. Daoutidis, P., Lee, J. H., Rangarajan, S., Chiang, L., Gopaluni, B., Schweidtmann, A. M., ... & Georgakis, C. (2024). Machine learning in process systems engineering: Challenges and opportunities. Computers & Chemical Engineering, 181, 108523.
2. Bishnu, S. K., Alnouri, S. Y., & Al-Mohannadi, D. M. (2023). Computational applications using data driven modeling in process Systems: A review. Digital Chemical Engineering, 8, 100111.
3. Lavezzi, G., Guye, K., Cichella, V., & Ciarcià, M. (2023). Comparative Analysis of Nonlinear Programming Solvers: Performance Evaluation, Benchmarking, and Multi-UAV Optimal Path

Planning. Drones, 7(8), 487.
4. Hedengren, J., & Nicholson, B. (2022). Open-Source Modeling Platforms (No. SAND2022-12379C). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).
5. Ploch, T., von Lieres, E., Wiechert, W., Mitsos, A., & Hannemann-Tamás, R. (2020). Simulation of differential-algebraic equation systems with optimization criteria embedded in Modelica. Computers & Chemical Engineering, 140, 106920.
6. Beal, L. D., Hill, D. C., Martin, R. A., & Hedengren, J. D. (2018). GEKKO optimization suite. Processes, 6(8), 106.
7. Fritzson, P., Pop, A., Abdelhak, K., Asghar, A., Bachmann, B., Braun, W., ... & Ostlund, P. (2022). The OpenModelica integrated environment for modeling, simulation, and model-based development. Mic.
8. Lee, A., Ghouse, J. H., Eslick, J. C., Laird, C. D., Siirola, J. D., Zamarripa, M. A., ... & Miller, D. C. (2021). The IDAES process modeling framework and model library—Flexibility for process simulation and optimization. Journal of advanced manufacturing and processing, 3(3), e10095.
9. Hall, K. K., Walmsley, T. G., Nemet, A., & Walmsley, M. (2022, September). Open-source tool for heat exchanger network synthesis. In PRES'22.
10. Udugama, I. A., Hall, K. K., Bayer, C., Young, B. R., & Walmsley, T. G. (2023). Real-time control and scheduling decision making in fermentation-based food process operations. In Computer Aided Chemical Engineering (Vol. 52, pp. 2381-2386). Elsevier.
11. Eslick, J. C., Zamarripa, M. A., Ma, J., Wang, M., Bhattacharya, I., Rychener, B., ... & Miller, D. C. (2022). Predictive modeling of a subcritical pulverized-coal power plant for optimization: Parameter estimation, validation, and application. Applied Energy, 319, 119226.
12. Nicholson, B., Siirola, J. D., Watson, J. P., Zavala, V. M., & Biegler, L. T. (2018). pyomo. dae: A modeling and automatic discretization framework for optimization with differential and algebraic equations. Mathematical Programming Computation, 10, 187-223.
13. Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C. D., & Misener, R. (2022). OMLT: Optimization & machine learning toolkit. Journal of Machine Learning Research, 23(349), 1-8.
14. Ding, H., Li, W., Duan, C., Xu, H., Ding, P., Zhang, Y., & Hong, G. (2021). A thermal-hydraulic analysis model of printed circuit heat exchangers for system simulation using Modelica. Journal of Nuclear Science and Technology, 58(12), 1296-1307.
15. Valmari, A. (1996, September). The state explosion problem. In Advanced Course on Petri Nets (pp. 429-528). Berlin, Heidelberg: Springer Berlin Heidelberg.
16. Amusat, O. O., Atia, A. A., Dudchenko, A. V., & Bartholomew, T. V. (2024). Modeling Framework for Cost Optimization of Process-Scale Desalination Systems with Mineral Scaling and Precipitation. ACS Es&t Engineering, 4(5), 1028-1047.
17. Lee, A., Parker, R., Poon, S., Gunter, D., Dowling, A., & Nicholson, B. (2024). Model Diagnostics for Equation Oriented Models: Roadblocks and the Path Forward. National Energy Technology Laboratory (NETL), Pittsburgh, PA, Morgantown, WV, and Albany, OR (United States).
18. Pahl, C., Jamshidi, P., & Zimmermann, O. (2020). Microservices and containers.
19. Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., & Steinder, M. (2015, September). Performance evaluation of microservices architectures using containers. In 2015 ieee 14th international symposium on network computing and applications (pp. 27-34). IEEE.