A Domain Specific Language for the Design of Artificial Intelligence Applications for Process Engineering

Lelio Campanile[1], Luigi Piero Di Bonito[1], Marco Gribaudo[2], and Mauro Iacono[1(✉)]

[1] Dip. di Matematica e Fisica, Università degli Studi della Campania "L. Vanvitelli", viale Lincoln 5, 81100 Caserta, Italy
{lelio.campanile,luigipiero.dibonito,mauro.iacono}@unicampania.it
[2] Dip. di Elettronica, Informatica e Bioingegneria Politecnico di Milano, via Ponzio 51, 20133 Milan, Italy
marco.gribaudo@polimi.it
https://www.deib.polimi.it/ita/home, https://www.matfis.unicampania.it

Abstract. Processes in chemical engineering are frequently enacted by one-of-a-kind devices that implement dynamic processes with feedback regulations designed according to experimental studies and empirical tuning of new devices after the experience obtained on similar setups. While application of artificial intelligence based solutions is largely advocated by researchers in several fields of chemical engineering to face the problems deriving from these practices, few actual cases exist in literature and in industrial plants that leverage currently available tools as much as other application fields suggest. One of the factors that is limiting the spread of AI-based solutions in the field is the lack of tools that support the evaluation of the needs of plants, be those existing or to-be settlements. In this paper we provide a Domain Specific Language based approach for the evaluation of the basic performance requirements for cloud-based setups capable of supporting chemical engineering plants, with a metaphor that attempts to bridge the two worlds.

Keywords: Artificial intelligence · process engineering · domain specific language · performance evaluation · cloud computing

1 Introduction

Current regulations for the safeguard of the environment require a strict optimization of plants that implement chemical processes, in order to reduce waste, pollution and risk and to optimize the efficiency of each stage and each subsystem. The nature of these plants, that include components with non-linear behavior and that may be one-of-a-kind elements of a chemical plant, challenges the consolidated tuning and control techniques and suggests the application of ad-hoc, self-adapting and time-variant controls, possibly considering a bal-

E. Hyytiä and V. Kavitha (Eds.): VALUETOOLS 2022, LNICST 482, pp. 133–146, 2023.
https://doi.org/10.1007/978-3-031-31234-2_8

anced tuning of parameters at both the subsystem and the system level. Domain experts of the process engineering field advocate the exploitation of consolidated approaches that may match these needs, based on Artificial Intelligence (AI), or, even more interesting, Explainable (XAI) or Trustworthy AI (TAI), that are novel in this sector and are still at a very experimental level.

Besides the extremely important problem of the adaptation of existing approaches and the possible discovery of new solutions, another problem is the evaluation and planning of the resources needed to implement the chosen AI-based solution. Cost parameters are of paramount importance in a market that is generally oriented to mass production, as the effort in training and applying the AI-based solution should be carefully taken into account and should actually lower the overall running costs of the plant. In fact, while domain experts have full control and expertise on design, maintenance and running costs of the plant, the evaluation of the same parameters on AI systems, that is an issue for computer systems engineers and software engineers as well, is generally far from their core professional interest.

The cost of the AI component of the system might be estimated considering the number of signals to be monitored, the frequency of sampling, the amount and intensity of control actions that are needed and the AI algorithms that are chosen, so that apparently the problem might be considered of secondary importance and solvable by an external intervention; but the nature, the dynamicity, the need for possible reconfigurations during ordinary operations, the multiplicity of use, the specificity of each single process plant and component make the problem not trivial, so that one should consider that the integration between the process engineers and the AI engineers in the team would benefit of the availability of design tools that allow a profitable and natural dialogue between the two parts.

In order to support (and promote) AI applications in the management, control and optimization of process plants, we propose a Domain Specific Language (DSL) for the description of sensitive parts of plants and the IT resources to be used to design the system and to implement the solution. The DSL aims at describing the setup of the system with the purpose of evaluating the needed amount of resources that allows to match the requirements of the plant and to quantify the needed amount of resources needed for the training or retraining phase of the plant, in order to guide the decision process for resource planning (private local cloud or public cloud, choice of commercial offer, costs evaluation). The DSL is oriented to provide a visual representation that is familiar both to process engineers and to IT engineers, to ease the collaboration and avoid specification errors and mismatches, with the goal of minimizing the design effort for both the parts of the team and to lower the communication barrier.

This paper is organized as follows: Sect. 2 describes the application domain; Sect. 3 provides related works; Sect. 4 describes the proposed DSL; Sect. 5 presents a case study; conclusions close the paper.

2 The Application Domain

A chemical plant is an industrial process plant that produces or processes chemicals on a large scale. The general objective of a chemical plant is to create new material wealth through chemical or biological transformation and separation of materials. Chemical plants use specialized equipment, units and technologies in the processes of production, treatment and separation of substances. Classic examples of applications may include polymer, pharmaceutical, food and some beverage production, petroleum refineries or bio-refineries, natural gas processing and biochemical plants, water and wastewater treatment plants. A chemical plant sees the use of different types of equipment, which are mainly fluid systems (e.g., chemical reactors, heat exchangers, liquid-gas, gas-liquid, liquid-liquid contact systems, tanks and separators) and auxiliaries for handling fluids and solids (e.g., piping, pumps, compressors, augers).

Chemical processes can be divided in two main categories: continuous processes and batch processes. Continuous processing plants, as the name implies, operate in an uninterrupted and continuous manner. Following the start-up phase of the plant, it is run in a more or less constant and, hopefully, totally optimum condition. Because the process should be in this ideal condition the majority of the time. Nonetheless, even the steady-state is changing with time. The most prevalent reasons of process operating point variations include changes in process product demand, changes in reactor/separator yield, heat exchanger clogging, and so on. Batch or discontinuous processes have a fixed time. These operations are frequently initiated on demand in order to produce the needed amount of product. Many processes in the food and biochemistry industries, such as fermentation processes, are of this sort. Because the specific chemicals must be produced seldom and frequently in extremely tiny quantities, running the facilities continuously would be financially unsustainable.

Chemical plants are often supplied with sophisticated equipment and a multiplicity of sensors. Sensors primary function is to give data for process monitoring and control. However, approximately two decades ago, academics began to use the huge volumes of data observed and kept in the process sector to construct predictive models. Data-driven models are more related to reality and better explain genuine process conditions since they are based on measurable data within process plants and so depict real process conditions. The primary, and still dominating, application area for these datasets is the prediction of process variables that can only be determined at low sample rates or by off-line analysis. Because these variables are frequently connected to process quality, they are critical for process management and monitoring. For these reasons, it is critical to give more information on these variables at a greater sample frequency and/or with a smaller financial cost. The industry is evolving toward a concept defined as "online prediction" with the use of soft sensors, with the goal of conducting process monitoring and process fault diagnosis. These duties involve detecting the state of the process and determining the source of any divergence from normal circumstances.

As a result, soft sensors can assist process operators in making faster, smarter, and more efficient decisions in order to ensure the long-term growth of the process sector since that many unions and governments have proposed innovative/sustainable manufacturing development plans in recent years. For example, in terms of Process Engineering, the ONU Sustainable Development Goals of clean water and sanitation, cheap and clean energy, responsible consumption and production, climate action, life below water, and life on land must be met by 2030. These objectives could only be met by incorporating smart systems into industrial manufacturing. The process sector has collected a massive quantity of data as a result of the extensive use of distributed control systems during the last few decades. While constructing first-principles models for more complicated processes is becoming more difficult, data-driven process modeling, monitoring, prediction, and control have received a lot of attention in recent years. Later, the process sector grew increasingly aware of the need of extracting knowledge from data, but these data are too large and complex for human modeling. The need for an automated method to model complex systems is increasing. Machine learning algorithms will be crucial in data mining and analytics in the process sector.

2.1 Process Measurements and Control

Process measurements include the application of metrology concepts to the process at hand. The goal is to collect values for the present process conditions and make this information available in a way that can be used by the control system, process operators, or management information systems. The expression measured variable or process variable refers to the process condition under consideration. Process measurements are classified into two types:

1. Constant measurements: a level measuring device that determines the liquid level in a tank is an example of a continuous measurement (e.g., in meters);
2. Measurements that are discrete: a level switch, which displays the presence or absence of liquid at the spot where the level switch is located, is an example of a discrete measurement.

Several process control applications in continuous processes rely on continuous measurements. Several process control applications in batch operations use both discrete and continuous measurements.

Temperature, pressure, flow rate, level, and composition measurement equipment, as well as online physical property measuring techniques, are the most common types of measurement devices utilized in the process industries. Chemical composition is typically the most difficult online measurement. A chemical composition analysis system may easily cost more than 100,000 dollar, and many composition analyzers used for process monitoring and control involve chemical conversion of one or more sample components prior to quantitative measurement. The sampling system includes all of the necessary equipment to provide a clean representative sample of a process stream to a process analyzer and dispose of that sample. When the analyzer is part of an automated control loop, the

sampling system's trustworthiness is just as crucial as the analyzer's or control equipment's. Modern control systems allow for physical separations of several hundred meters between the measuring equipment, the control unit, and the final control element. The measured variable must be transmitted from the measuring device to the control unit, and the controller output must be transmitted from the control unit to the final control element. To make electronic transmission systems less sensitive to interference from magnetic fields, current rather than voltage is employed for the transmission signal. The signal has a range of 4 to 20 mA. There can only be one transmitter in each circuit or "current loop". For most measurement variable transmissions, the lower range is 4 mA and the maximum range is 20 mA [10]. In the process industry, these signals are frequently recorded using a data logger at various collection rates, resulting in a dataset that may be utilized in digital technologies for process control. In terms of database dimension, it is dependent on data logger performance in terms of acquisition rates, so:

- for lower limit an acquisition rate equal to 1 kHz can be considered: this limit primarily depends on both economic factors defined by data logger price and characteristic time of the considered process;
- for upper limit acquisition rates between 0.005–1 Hz can be considered: an analog signal from a traditional sensor is a continuous transmission signal ranging from 4 to 20 mA, but an analog/digital signal from an analyzer might be affected by a delay time determined by the working principle of the analyzer.

3 Related Works

3.1 AI and Chemical Process Plants

A classic example used in literature about the application of AI techniques to supervision and control of chemical process plants is the Tennessee Eastman Process, because of the availability of a large and consistent public dataset. Different applications of machine learning for fault diagnosis and identification based on the Tennessee Eastman Process dataset have been studied, such as in the papers analyzed in the following. [9,12,15] propose a new fault identification and diagnosis methods for manufacturing processes based on AI methodologies (i.e. Bayesian Neural Networks, Auto-associative Neural Networks, Generative Adversarial Networks), which enables (1) fault detection in nonlinear processes and (2) direct identification of faults using easy-to-interpret identification diagrams and fault propagation path analysis. One issue is that the usual dataset used for monitoring the Tennessee Eastman process has just a limited number of data samples, which might be quite restrictive for deep learning analysis. Some implemented datasets of these studies, are 500 times larger than the average datasets and large enough for deep learning. In Fault Diagnosis application, another issue could be the absence of data from essential sensors readings due to hardware or network issues. Results show that these AI methodologies can

reduce the impact on problem detection and identification even in the face of repeatedly missing reading from sensors that are important to the monitoring system proper operation.

In [5,7,14] other examples of Fault Diagnosis applications in the field of process engineering are given, which do not consider the Tennessee Eastman process as a case study. In these studies, other process engineering equipment (such as chemical reactors and separation units) was analyzed in order to develop AI-based methodologies that can improve the performance of the online monitoring system of the units.

3.2 DSL for Performance Evaluation

A comprehensive survey on the use of DSL in the field of Model Driven Engineering (MDE) can be found in [1]. While in the MDE approach exploits metamodeling for the automatic generation of software implementations and/or system configurations, the SIMTHE*Sys* framework [13] supports the definition of DSL for the specification of performance oriented models that may be evaluated by means of different analytical or simulation-based approaches. For example, the SIMTHE*Sys* approach has been used to define DSL for Big Data systems in [2], for multicore CPU and GPU based computing architectures in [6], for lambda computing architectures in [11] and for cloud computing systems in [3], in all cases to support the design and dimensioning of systems in the target domains: these papers have inspired this one.

4 Design of the PE-AI-Perf DSL

We decided to use as running example the Tennessee Eastman process as an example that may support the design choices in designing the PE-AI-Perf DSL, in order to contextualize our work and guide the reader in getting familiar with the domain modeling problems. This example is considered by domain experts to be able to provide a sufficiently complete and sufficiently complex presentation of the target systems, and some quantitative references are provided as well to suggest the terms of the general problem.

4.1 Process Description

The Tennessee Eastman process model is a realistic simulation of a chemical plant that is extensively used as a reference for control and monitoring studies. The procedure is detailed in [8], and the FORTRAN code is available on the internet. Figure 1 depicts the process flow diagram, which includes five primary units: reactor, condenser, compressor, separator, and stripper.

The reaction yields two products from four reactants. An inert and a by-product are also present, for a total of eight components labeled as A, B, C, D, E, F, G, and H. The technique provides for a total of 53 measurements, 41 which are of process variables and 12 which are controlled variables. The Tennessee

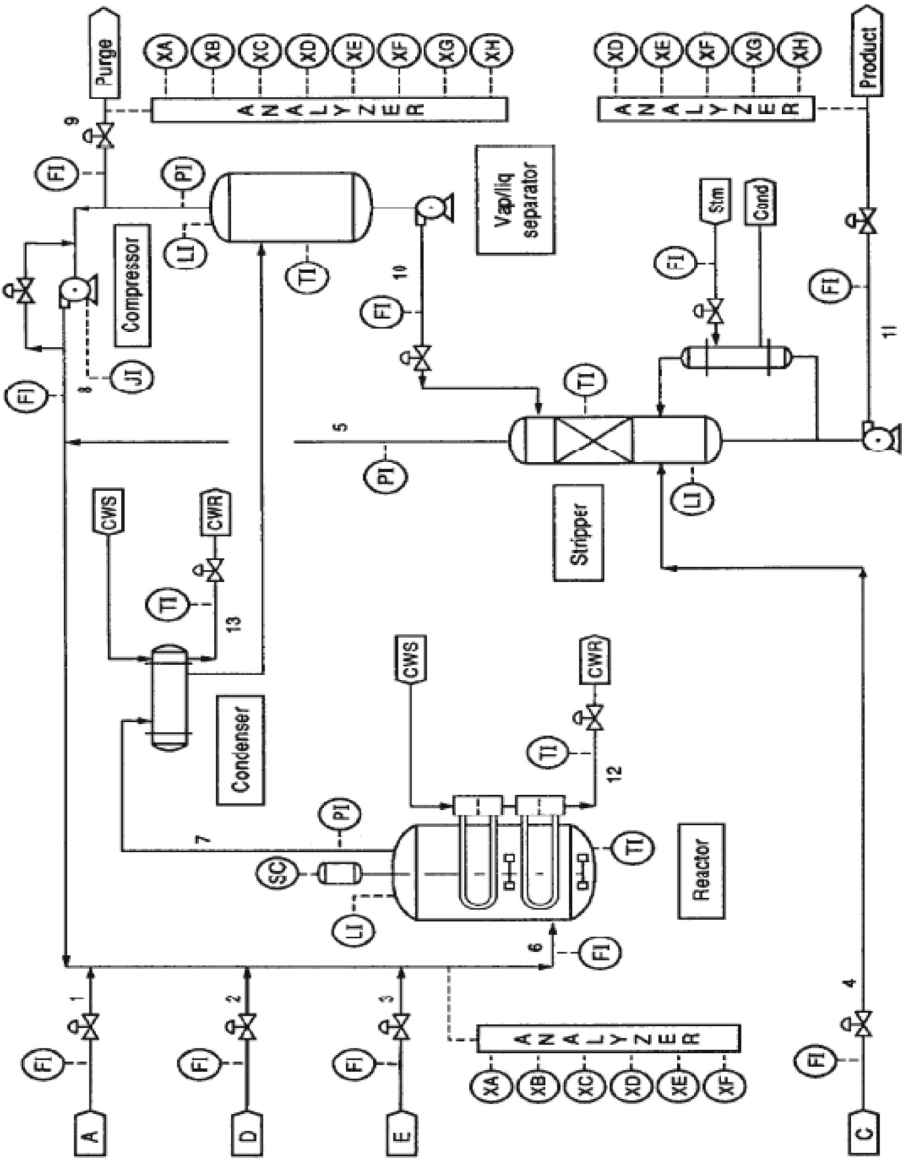

Fig. 1. The Tennessee Eastman Process Process Flow Diagram in [8].

Eastman Process simulation includes 21 preprogrammed defects (classes/labels) with a variety of disturbance types and locations and under typical working conditions, the system is in production mode, with a sample period of 3 min.

4.2 Definition of the Elements of the DSL

For our purposes, from the process side the analysis of a plant is oriented to understand which devices produce data that will contribute to build a dataset and will be involved in the supervision and control tasks and, specifically, which sensors and actuators are related to each device, which devices collect and transmit data: we are interested in identifying data types, sampling and operation frequencies, sampling distribution, dataset size, possible bandwidth constraints. From the computing system side, we are interested in modeling the data storage needs, the computational needs of the AI solutions workloads, the bindings between workloads and devices, the mapping between workloads and Virtual Machines (VM), active (or available) VMs and their configurations.

As there is a variability of devices in different projects, the possibility of building new devices by aggregation of elementary elements is preferable to the definition of a fixed set of devices (this solution is also suitable for the definition of soft sensors, for the purpose of this work). As the PE-AI-Perf DSL is built on top of the SIMTHE*Sys* framework, this is supported by native features.

A complete scenario can be defined by means of the following DSL elements:

- *Sensor*: it represents an elementary data source installed on a device, and is characterized by a *type* property, describing the numerical type of read values, a *timedistribution* property, describing the time distribution with which the sensor is sampled and its parameters, to shape the input stream to the control task, and a *sizedistribution* property, describing the distribution of the length of the packets generated (measured in bytes);
- *Actuator*: it represents a variable to be produced by the computing part of the system, and is characterized by a *distribution* property, describing the time distribution with which the actuator should receive input from the control task, to shape the output stream of the control task;
- *Device*: it represents one of the devices of a plant, and groups the *sensor* and *actuator* elements representing the installed ones; being a SIMTHE*Sys* submodel, it also refers to contained *sensor* and *actuator* elements with proper properties;
- *Datalogger*: it represents a data logger in the plant, and is characterized by a *storagesize* property, defining the total amount of data that can be logged before a download operation is needed, a *maxsamplefreq* property, defining the highest sampling frequency supported, a *maxinputs* property, defining the number of available input lines to connect *sensor* elements, a *netbandwidth* property, defining the maximum connection bandwidth on the computer network used to interface with the computing architecture of the system;
- *NetworkConnection*: it represents the connection between the plant plane and the data center, being it local to the site or remotely connected, and is characterized by a *totalbandwidth* property, defining the maximum connection bandwidth on the computer network used to interface with the computing architecture of the system, a *technology* property that describes the relevant technical parameters of the connection;

- *Workload*: it represents a single AI solution used for a device or a set of devices, and is characterized by an *requiredCPU* property, defining the computing needs for a single run of the algorithm on a single input in terms of the number of CPU operations, a *requiredGPU* property, defining the computing needs on a GPU in terms of the number of GPU operations;
- *VM*: it represents a virtual machine available to run a *workload*, and is characterized by a *cores* property, defining the number of available virtual cores, a *corespeed* property, defining the operational speed of a core, a *GPUs* property, defining the number of available GPUs, a *GPUspeed* property, defining the operational speed of a core, a *totalstorage* property, defining the amount of the available storage;
- *MapArc*: it represents the mapping between a *Device* and a *Datalogger*, a *Datalogger* and a *NetworkConnection*, a *NetworkConnection* and a *Workload*, a *Workload* and a *VM*;
- *BindArc*: it represents the mapping between a *Workload* and a *Device*, to specify which *Device* elements are related to a *Workload* element. This arc is characterized by a *weight* property, used to define how many instances of the corresponding data are required to run on instance of the considered workload.

Graphical symbols for the elements are depicted in Fig. 2.

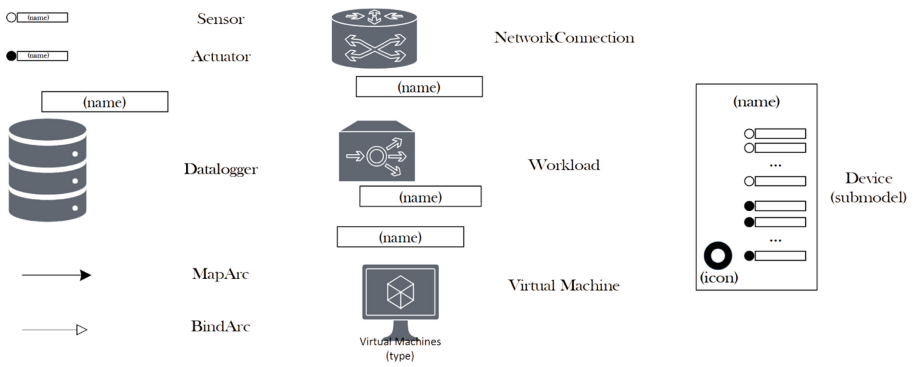

Fig. 2. The symbols for the language elements.

5 A Case Study

In this preliminary work, we do not consider *Actuator*s, and we consider that all data produced by *Sensor*s at a *Device* is aggregate and transmitted as single data blocks thorough the *Network Connection*s. We also focus on workloads that can be entirely run on CPU, disregarding the GPUs. To simplify the description of the semantics of the resulting model, we perform an intermediate translation

of the considered case study into a conventional GSPN/QN model, that we analyze using JMT - Java Modelling Tools [4]. Figure 4 shows the result of this translation process for the DSL model shown in Fig. 3. We also do not consider storage and focus on network and computation requirements.

The equivalent GSPN/QN model is characterized by $\#D + \#W$ classes, one for each *Device* ($\#D$ in total), and for each *Workload* (up to $\#W$). Each *Device* is converted in a QN source, that produces jobs of the corresponding class, at the arrival rate defined in the *timedistribution* parameter of the DSL object.

Dataloggers are converted into FCFS single server queues, where the service time of each class is derived from the *sizedistribution* of the sensor, and the *netbandwidth* property of the logger. Similarly, *NetworkConnection* DSL nodes are converted into processor sharing single server queues, where the service time of each class is derived from the *sizedistribution* of the sensor, and the *totalbandwidth* property of the router.

*VirtualMachine*s are transformed into multiple server processor sharing queues, where the number of servers corresponds to the *cores* parameter of the node. Service time distributions are computed from the *corespeed* parameter of the node and from the *requiredCPU* parameter of the workload.

Workload nodes creates a GSPN submodel with one place and two immediate transitions, that aggregates all the data coming from the sensors and required by the considered algorithm. In particular, following the connections defined by the corresponding *BindArc*s, the immediate transitions ending with $\_T$ consume one or more tokens of the classes according to the parameter *weight* of the binding arcs, and produces one token of the destination workload class in the queue modeling the virtual machines. It is worthless analyzing old data: if newer data arrive before the previous value has been consumed, the value is replaced. This is modeled by the immediate transitions ending with $\_D$, that can fire in as many modes as data types. Let as call w the corresponding weight associated to the arc connecting the data type with the workload. To replace older data with newer versions, the input arcs to the immediate transition have weight $w + 1$, while the output arcs have w for the corresponding firing mode.

Table 1 shows the parameters used in the example scenario. In particular, execution times are measured in ECU[1], and the values associated with distribution parameters define their average.

We start studying the effect of the number of VMs deployed in the cloud on both the response time and throughput of the system: results are shown in Fig. 5. With a single VM the system is unstable, and the response time tends to the infinity. In this case, the throughput of the system reduces, due to the full utilization of the cloud. System begins being stable with at least two VMs: in this case, however, queuing occurs very frequently, affecting the response time.

[1] The ECU, EC2 Computing Unit, is a measure defined by Amazon to compare computing power of a VM with respect to its reference VM on Amazon AWS cloud, that can be considered as equivalent to a 1.0–1.2 GHz 2007 Intel Xeon or AMD Opteron CPU; it is now less popular than when it was introduced, but we consider it fit to the context and the purpose of this paper.

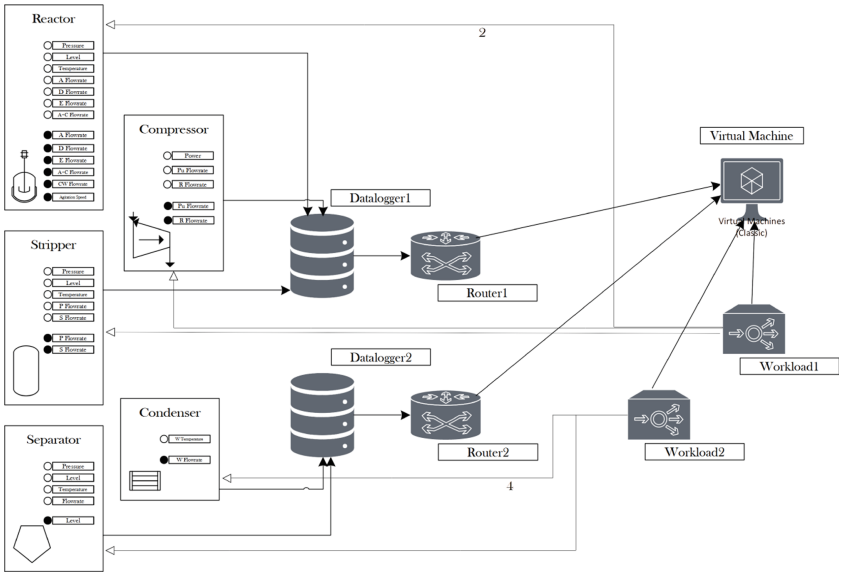

Fig. 3. The model for the example plant (online in-the-loop operations).

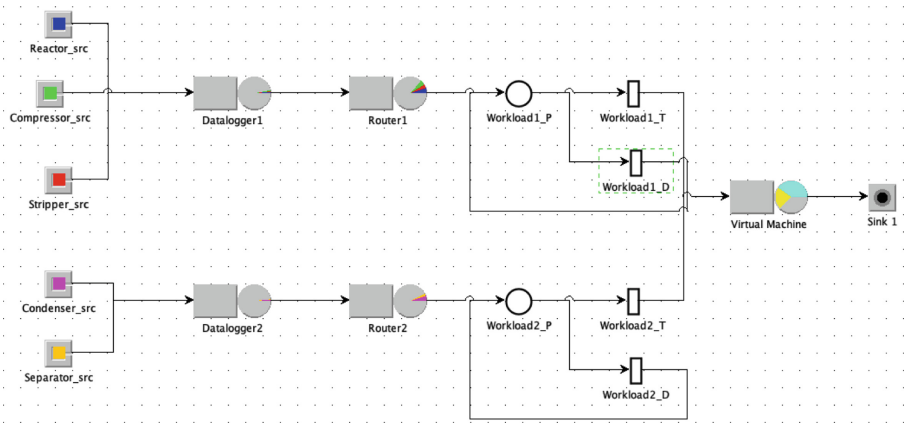

Fig. 4. The equivalent QN-PN model for the example plant.

With 4, 8 or 16 VMs response time stabilizes respectively to around 2 s and 1 s for both workloads, reaching performances very close to the considered service times. For this reason, a configuration with 4 VMs seems to be the one with the optimal tradeoff between performance and cost.

Throughput however, although becoming stationary, only reaches around 0.31 and 0.38 executions per second for both workloads. This is severely reduced with respect to the expected 0.5 executions per second which could be achieved with the considered data generation rate. Analyzing the firing modes of the drop

Table 1. Process variables to be manipulated.

Element	Parameter	Value
Reactor	*timedistribution*	$Erlang_4$ (1 s)
Reactor	*sizedistribution*	*Exponential* (100 KB)
Compressor	*timedistribution*	$Erlang_4$ (2 s)
Compressor	*sizedistribution*	*Exponential* (150 KB)
Stripper	*timedistribution*	$Erlang_4$ (2 s)
Stripper	*sizedistribution*	*Exponential* (200 KB)
Condenser	*timedistribution*	$Erlang_4$ (0.5 s)
Condenser	*sizedistribution*	*Exponential* (50 KB)
Separator	*timedistribution*	$Erlang_4$ (2 s)
Separator	*sizedistribution*	*Exponential* (100 KB)
Datalogger 1	*netbandwidth*	100 Mbps
Datalogger 2	*netbandwidth*	100 Mbps
Router 1	*totalbandwidth*	20 Mbps
Router 2	*totalbandwidth*	20 Mbps
Workload 1	*requiredCPU*	*Exponential* (2 $ECU \cdot s$)
Workload 2	*requiredCPU*	*Exponential* (1 $ECU \cdot s$)
Virtual Machine	*cores*	4
Virtual Machine	*corespeed*	1 *ECU*

transitions (Workload1_D and Workload2_D), we can see that around 19.5% of readings for Workload 1, and around 12% of the second workload are lost, regardless of the VM settings.

This problem is due to the jittering in the incoming data: the considered coefficient of variation $c_v = 0.5$ of the Erlang distributions defining the inter-arrival times of the data produced by the sensors is too high, and produces very frequently cases in which some data needs to be discarded to maintain only the most updated samples. Figure 6 shows the effect of varying the coefficient of variation on both the throughput and the probability of dropping some input. When it reduces below 1/16, the effect of jittering becomes negligible, and almost all input readings can be used for controlling the system. It is also interesting to note that Workload 1 is more affected by the problem since it requires the fusion of a larger number of sources, thus increasing the effect of jittering.

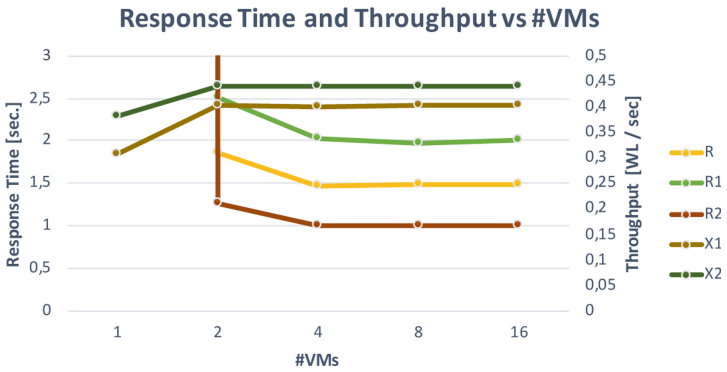

Fig. 5. Response time and throughput as function of the number of VMs deployed.

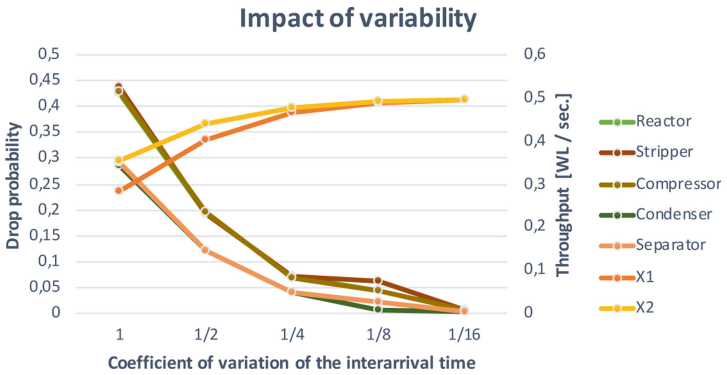

Fig. 6. Drop rate as function of the Coefficient of variation.

6 Conclusion and Future Work

In this paper we presented a DSL for the description of a computing architecture needed to implement an AI-based control system for the processes in chemical plants, oriented to performance evaluation and to support designers in defining and choosing a suitable cloud-based solution. We leveraged our previous results on DSL to obtain a user-friendly description that lowers the barrier between the two fields of chemical engineering and computer science and engineering practice. Future works include the definition of more complex and concurrent solutions for larger plants and more critical processes and a campaign to validate the approach and the solution with companies.

Acknowledgements. This work has been partially funded by the internal competitive funding program "VALERE: VAnviteLli pEr la RicErca" of Università degli Studi della Campania "Luigi Vanvitelli" and is part of the research activity developed within Industrial Ph.D. Programme PON 2014-2020.

References

1. Abouzahra, A., Sabraoui, A., Afdel, K.: Model composition in model driven engineering: a systematic literature review. Inf. Softw. Technol. **125**, 106316 (2020). https://doi.org/10.1016/j.infsof.2020.106316
2. Barbierato, E., Gribaudo, M., Iacono, M.: Modeling apache hive based applications in big data architectures. In: Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools, ValueTools 2013, pp. 30–38. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels (2013). https://doi.org/10.4108/icst.valuetools.2013.254398
3. Barbierato, E., Gribaudo, M., Iacono, M., Jakobik, A.: Exploiting CloudSim in a multiformalism modeling approach for cloud based systems. Simul. Modell. Pract. Theory **93**, 133–147 (2018)
4. Bertoli, M., Casale, G., Serazzi, G.: JMT: performance engineering tools for system modeling. SIGMETRICS Perform. Eval. Rev. **36**(4), 10–15 (2009). http://doi.acm.org/10.1145/1530873.1530877
5. Bhakte, A., Pakkiriswamy, V., Srinivasan, R.: An explainable artificial intelligence based approach for interpretation of fault classification results from deep neural networks. Chem. Eng. Sci. **250**, 117373 (2022). https://doi.org/10.1016/j.ces.2021.117373
6. Cerotti, D., Gribaudo, M., Iacono, M., Piazzolla, P.: Modeling and analysis of performances for concurrent multithread applications on multicore and graphics processing unit systems. Concurr. Comput. Pract. Exp. **28**(2), 438–452 (2016). https://doi.org/10.1002/cpe.3504
7. Dai, Y., Zhao, J.: Fault diagnosis of batch chemical processes using a dynamic time warping (DTW)-based artificial immune system. Ind. Eng. Chem. Res. **50**(8), 4534–4544 (2011). https://doi.org/10.1021/ie101465b
8. Downs, J.J., Vogel, E.F.: A plant-wide industrial process control problem. Comput. Chem. Eng. **17**(3), 245–255 (1993). https://doi.org/10.1016/0098-1354(93)80018-I
9. Dzaferagic, M., Marchetti, N., Macaluso, I.: Fault detection and classification in industrial IoT in case of missing sensor data. IEEE Internet Things J. **9**(11), 8892–8900 (2022). https://doi.org/10.1109/JIOT.2021.3116785
10. Green, D.W., Perry, R.H.: Perry's Chemical Engineers' Handbook, Eighth Edition, 8th edn. McGraw-Hill Education (2008). https://www.accessengineeringlibrary.com/content/book/9780071422949
11. Gribaudo, M., Iacono, M., Kiran, M.: A performance modeling framework for lambda architecture based applications. Futur. Gener. Comput. Syst. **86**, 1032–1041 (2017). https://doi.org/10.1016/j.future.2017.07.033
12. Heo, S., Lee, J.H.: Statistical process monitoring of the Tennessee Eastman process using parallel autoassociative neural networks and a large dataset. Processes **7**(7) (2019). https://doi.org/10.3390/pr7070411
13. Iacono, M., Gribaudo, M.: Element based semantics in multi formalism performance models. In: MASCOTS, pp. 413–416. IEEE (2010)
14. Liu, J., et al.: Explainable fault diagnosis of gas-liquid separator based on fully convolutional neural network. Comput. Chem. Eng. **155**, 107535 (2021). https://doi.org/10.1016/j.compchemeng.2021.107535
15. Sun, W., Paiva, A.R., Xu, P., Sundaram, A., Braatz, R.D.: Fault detection and identification using Bayesian recurrent neural networks. Comput. Chem. Eng. **141**, 106991 (2020). https://doi.org/10.1016/j.compchemeng.2020.106991