PIACERE Integrated Development Environment

Eliseo Villanueva
Prodevelop SL
evillanueva@prodevelop.es

Ismael Torres
Prodevelop SL
itorres@prodevelop.es

Eneko Osaba
TECNALIA, Basque Research and Technology Alliance (BRTA)
eneko.osaba@tecnalia.com

Sergio Canzoneri
DEIB, Politecnico di Milano
sergio.canzoneri@mail.polimi.it

Andrea Franchini
DEIB, Politecnico di Milano
andrea.franchini@polimi.it

Lorenzo Blasi
Hewlett Packard Italiana s.r.l
lorenzo.blasi@hpe.com

ABSTRACT

This article presents a model-driven engineering (MDE) integrated development environment (IDE) to assist the DevSecOps (Development Security and Operations) process. This tool has been developed within the PIACERE H2020 project, which proposes a framework composed of a set of tools developed to support all phases of the DevSecOps life cycle including modeling, test/validation, build/generate, deployment, operate and modeling. PIACERE IDE is an Eclipse based tool, that acts as the front-end for this framework, and plays a key role in integrating other PIACERE tools. The IDE allows developers to access the different tools in a simple and unified way.

CCS CONCEPTS

• **Software and its engineering**; • **Software notations and tools**; • **Development frameworks and environments**; • **Integrated and visual development environments**;

KEYWORDS

IDE (integrated development environment), DevSecOps, Eclipse, IaC (Infrastructure as Code)

ACM Reference Format:
Eliseo Villanueva, Ismael Torres, Eneko Osaba, Sergio Canzoneri, Andrea Franchini, and Lorenzo Blasi. 2023. PIACERE Integrated Development Environment. In *3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum (ESAAM 2023), October 17, 2023, Ludwigsburg, Germany*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3624486.3624507

1 INTRODUCTION

The interest in automating the deployment of cloud solutions and their integration with a DevOps methodology has been a growing topic of interest in recent years. However, the heterogeneity of technologies and software development methods in use remains a challenge for researchers and practitioners. Moreover, security is often not taken into consideration throughout the whole process.

The PIACERE Project [1] tries to solve the previous challenge providing tools, methods, and techniques to enable most organizations to fully embrace the IaC (Infrastructure as Code) approach, through the DevSecOps philosophy [2]. PIACERE tools enable the automation of deployment, configuration, and monitoring tasks. PIACERE solution consists of an integrated DevSecOps framework (Figure 1) to develop, verify, release, configure, execute, and monitor infrastructure as code. In addition, the proposed solutions are cloud vendor independent, allowing solutions to be deployed on the most relevant cloud providers today. The PIACERE project ends in November 2023, so the tools presented may still undergo some modifications and improvements, although they are expected to be minimal.

The pivotal tool of the project is the IDE, it supports the different DevSecOps activities using a single integrated environment and reduces the learning curve for new teams.

In the PIACERE project a novel DevSecOps Modelling Language (DOML), has been developed to hide the specificities and technicalities of the current solutions. Moreover, PIACERE also provides a ICG (Infrastructural Code Generator) tool to translate this DOML language into source files for different existing IaC languages [3, 4]. With the combination of these tools, the time needed for creating infrastructural code for complex applications will be reduced by increasing the productivity

1.1 DOML: DevSecOps Modelling Language

The aim of these tools is to provide the DevSecOps teams the tools and environments to simulate, package, release and configure an optimized deployment of the IaC.

DOML [5] is an end-user declarative language enabling the modelling of provisioning, deployment, and configuration of complex infrastructural software. It aims at describing cloud applications that are agnostic of the specificities and technicalities of the different providers and current Infrastructure-as-Code tools, and it has been designed to be easy to read and write for non-expert users.

A DOML model can then be translated into executable Infrastructure-as-Code languages, thanks to the PIACERE ICG tool: currently, supported languages include Terraform for provisioning and Ansible for configuring the infrastructure.

DOML provides several modelling perspectives in a multi-layer approach. An application can be described in four layers: application layer, abstract infrastructure layer, concrete infrastructure layer and optimization layer. The separation between the abstract and the concrete layer, allows developers to describe the structure of the cloud applications in an abstract manner, mapping the different

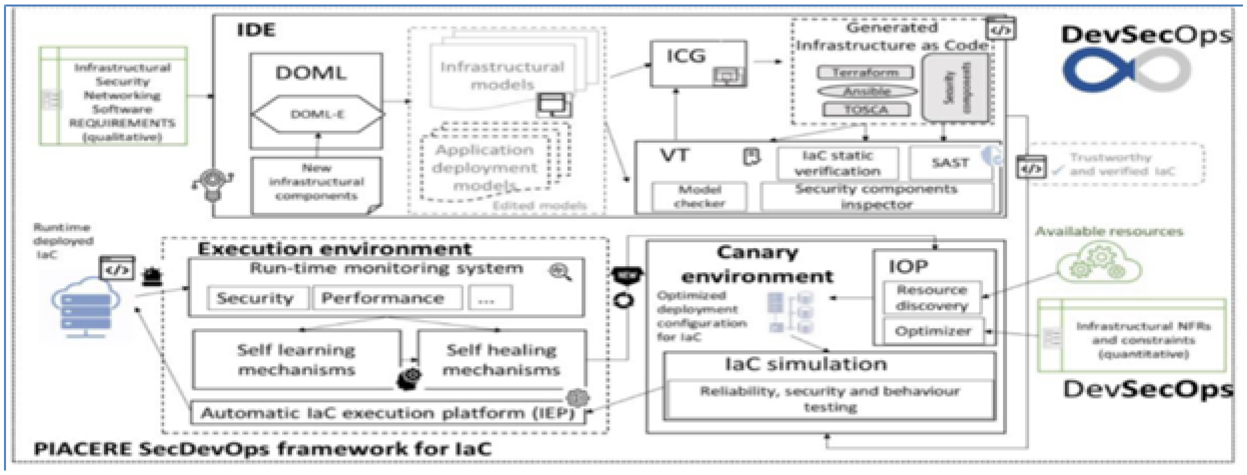
IDE
DOML
DOML-E
New infrastructural components
Infrastructural Security Networking Software REQUIREMENTS (qualitative)
Infrastructural models
Application deployment models
Edited models
ICG
Generated Infrastructure as Code
Terraform
Ansible
TOSCA
Security components
VT
Model checker
IaC static verification
SAST
Security components inspector
DevSecOps
Trustworthy and verified IaC
Runtime deployed IaC
Execution environment
Run-time monitoring system
Security
Performance
...
Self learning mechanisms
Self healing mechanisms
Automatic IaC execution platform (IEP)
Canary environment
Optimized deployment configuration for IaC
IOP
Resource discovery
Optimizer
IaC simulation
Reliability, security and behaviour testing
Available resources
Infrastructural NFRs and constraints (quantitative)
DevSecOps
PIACERE SecDevOps framework for IaC

Figure 1: PIACERE Framework overview.

software components to the infrastructure elements, enabling the usage of different concretizations, even on different providers, to match one particular deployment.

Moreover, DOML is meant to be extensible: the set of extension mechanisms that allow users to create new concepts from the existing ones takes the name of DOML-E.

As for the internal structure, DOML consists of a metamodel and a syntax. The metamodel was built on top of Eclipse Modeling Framework (EMF), relying on the Emfatic textual syntax to generate an Ecore metamodel. Thanks to the usage of Emfatic, the metamodel is easier to handle and the process to modify it, defining new concepts or extending the existing ones, is simpler.

The syntax was developed starting from the metamodel, using the Eclipse Xtext framework. However, in order for it to be more flexible and suitable for the purpose of translating models into Infrastructure-as-Code languages, the syntax was not automatically generated from the Ecore using the Xtext wizard; instead, it was written in a custom manner, still maintaining a strong alignment with the metamodel. Thanks to the usage of Xtext, users are provided with an editor (incorporated in the IDE) which guides them in the building of syntactically correct models, through code completion suggestions and error warnings.

Finally, Xtext also provides the tools to convert DOML models to an XML-based format, named DOMLX, which is used as an internal representation of DOML for most of the PIACERE tools. Such a serialized representation of models' data structures allows the tools to be independent of the specific syntax used to generate a model, leading to a more flexible approach to updating DOML.

By the end of the project, the reverse conversion from DOMLX to DOML format will be supported, allowing the transformation of DOML models from an older version of the language to a newer one, thus guaranteeing backward compatibility.

2 PIACERE DESIGN TIME TOOLS

PIACERE tools operate in two different scenarios: design time and run time. In this article, only the design time tools will be discussed, as they are the ones that allow the user to model, validate, optimize, and generate their IaC and have it ready for deployment.

2.1 IDE: Integrated development environment

The IDE, as mentioned above, allows developers to access all PIACERE tools. The development of this Integrated Development Environment has been carried out using consolidated tools such as Eclipse EMF that allow a simple integration of models such as those required by the DOML language (integrated via plug-in) and allow the use of REST APIs in an easy way to integrate some of the tools in the IDE.

The Eclipse IDE has been customized with suitable plug-ins that integrate the execution of the different tools, in order to minimize learning curves and simplify adoption. Not all tools are integrated in the same way. Several integration patterns, focusing on the Eclipse plugin architecture, have been defined. Thanks to the different integration patterns applied, there is independence between the IDE and the integrated tools, facilitating their decoupling and allowing them to be updated very quickly without the need to release a new version of the IDE.

A new section (Figure 2) in the preferences menu has also been created for the PIACERE Project, where the different access parameters to the API endpoints of each of the tools can be easily modified. PIACERE Tools are accessible through the PIACERE Tools menu (Figure 3) by right-clicking on the different files in the workspace.

2.2 VT: Verification Tools

The Verification Tools of PIACERE consists of two static analysis tools: the DOML Model Checker and the IaC Security Inspector.

The DOML Model Checker is used at design time to validate DOML models and verifying their consistency and correctness. The verification results are presented in a user-friendly interface with details on the issue source and suggestions on how to solve it.

The IDE communicates with it through a REST API, leveraging the DOMLX format, an XMI file containing all the elements and relationships specified in DOML. Using the ECORE metamodel, it

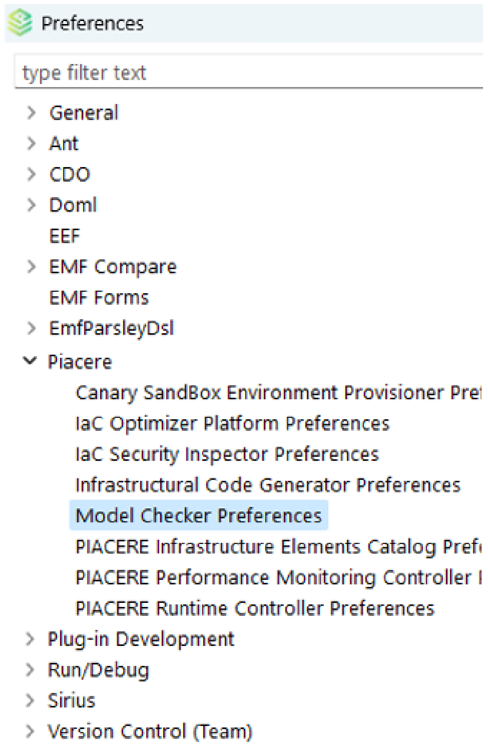
Preferences
type filter text
General
Ant
CDO
Doml
EEF
EMF Compare
EMF Forms
EmfParsleyDsl
Piacere
Canary SandBox Environment Provisioner Pref
IaC Optimizer Platform Preferences
IaC Security Inspector Preferences
Infrastructural Code Generator Preferences
Model Checker Preferences
PIACERE Infrastructure Elements Catalog Pref
PIACERE Performance Monitoring Controller I
PIACERE Runtime Controller Preferences
Plug-in Development
Run/Debug
Sirius
Version Control (Team)

Figure 2: PIACERE Preferences menu section.

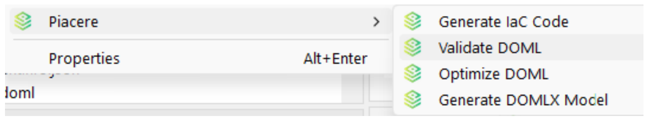

Figure 3: PIACERE contextual Menu.

is possible to access the DOML model in external tools, through libraries such as PyEcore.

The Model Checker is powered by the Z3 Theorem Prover developed by Microsoft Research and was built with Python and deployed in the PIACERE infrastructure as a container.

Users can further customize the checks that Model Checker runs via a domain specific language that allows them to exclude certain requirements (should they cause issue with a specific configuration), enable optional ones, and even write new requirements.

Another feature present in the Model Checker is a tool to check the compatibility with selected cloud service providers, ensuring that certain configurations are supported (e.g.: operating systems, architectures, keypair algorithms) and all the minimum required elements are present. The results are presented as a compatibility matrix.

The IaC Security Inspector is used to ensure the safety of the IaC code generated from DOML after the design phase, through an expansive suite of checks. A set of these checks cover syntactic problems of the languages used various IaC tools (i.e.: Terraform, TOSCA, Ansible...) such as typos and exposed secrets, like hard-coded passwords; another set of checks instead focuses on ensuring that IaC component and their dependencies are not affected by vulnerabilities.

The IaC Security Inspector receives the IaC as input and returns the list of vulnerabilities such as configuration errors, warnings, and suggestions as output.

Both tools receive DOMLX as input and produce their output as JSON (useful for inserting the Model Checker in a larger process as an automated step, such as the PIACERE Self-Healing, in the case of the DOML Model Checker) or HTML, which can be displayed to the user with a clean and human-readable UI.

2.3 IOP: PIACERE Optimizer Infrastructure

The main basis of the optimization problem formulated in the project is comprised by a service which must be deployed, and a catalogue of infrastructural elements (IEC). Thus, the main goal is to find the most appropriate IaC configuration to be deployed on the most suitable infrastructural elements that optimally meets the predefined restrictions.

Thus, the principal responsibility of the IOP tool is to find the optimum infrastructure to be deployed considering the data provided as input. This data must be introduced using the previously detailed DOML modelling language, and it should contain two main aspects: the objectives to optimize, and the requirements that the solution must meet.

On the one hand, the objectives considered by the current version of the IOP are i) the cost, ii) the availability, and iii) the performance of the overall deployment. On the other hand, seven different requirements can be deemed by the optimizer: i) a maximum cost for the overall configuration, ii) a minimum performance, iii) a minimum availability, iv) a minimum memory for each of the elements deployed, v) restrict the provider for the selected elements, vi) restrict the region for the elements, and vii) to establish the amount of elements to deploy.

With all this information, the IOP resorts to the well-known NSGA-II [6] and NSGA-III [7] multiobjective algorithms for conducting the matchmaking of the infrastructure. Thus, the IOP success if these artificial intelligence optimization methods can provide the user with the optimized deployment configuration. For this purpose and considering that the problem to solve is a multiobjective one, several solutions are offered and ranked by the IOP to the user.

Finally, within PIACERE, the IOP is employed into two different contexts: the initial deployment of the service, and the redeployment once it is already running. Thus, the IOP has a different role depending on whether it is called from the Design Time phase or the Run Time phase:

- The IOP in the Design Time: in this phase, the IOP is called from the PIACERE IDE. In this regard, and once the input DOML is composed, the IOP service can be called from the IDE just pushing the right click of the mouse over the DOML file and selecting PIACERE -> Optimize DOML. Once the optimization is conducted, the service offers the ranked solutions, which are subsequently use by the ICG for generating the IaC.
- The IOP in the Run Time: in this phase, the IOP is part of the self-healing process, and it is called once the system detects that a re-deployment is needed.

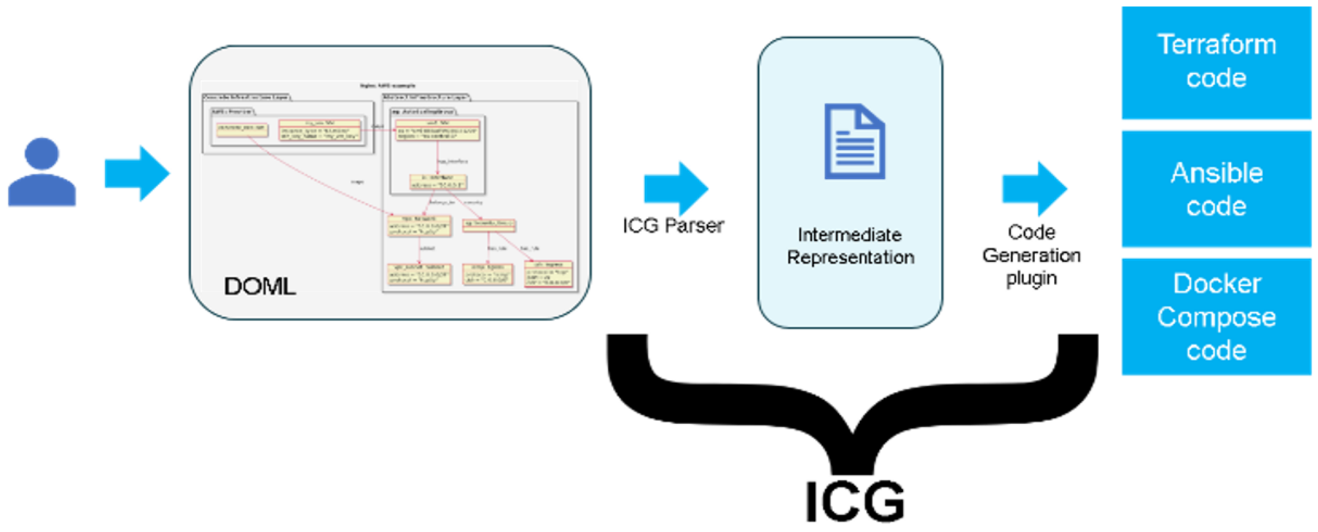

Figure 4: ICG Architecture.

2.4 ICG: Infrastructural Code Generator

The ICG tool (Figure 4) can be seen as a DOML compiler that generates executable IaC code. A parser extracts all the needed information from the input DOMLX model, creating an Intermediate Representation, and the code generation plugins create the output IaC code in multiple languages, depending on the activity to be automated. For example, Terraform code is created for provisioning virtual machines and network, and Ansible or Docker Compose code for deploying applications on the provisioned infrastructure.

ICG can be used either as command line tool, to be easily used in manual testing or CI/CD pipelines, or as containerized service that exposes a REST API. This API is used to integrate it in the IDE.

Multiple target platforms are supported in the last version of ICG. The generated IaC code can create infrastructure on cloud providers, such as AWS and Azure, but also on premise in Openstack or VSphere environments.

ICG generates IaC code using templates. The provided library includes templates for virtual machines, networks, subnets, security groups (firewalls) and autoscaling groups (based on load balancing).

Being based on templates, ICG can be easily extended to support new target platforms (e.g., GCP) or additional resources (e.g., switch or dbms), by adding the corresponding templates. Another possible extension is to add a new target IaC language (e.g., Kubernetes), but this is slightly harder, as it involves not only providing templates, but also writing (in Python) a new code generation plugin.

The last version of the ICG can also integrate custom code provided by the user. The new ICG API accepts in input a zip package including both the DOMLx model and a directory tree with a predefined structure, containing custom IaC code, e.g., an Ansible playbook. The ICG output integrates the given custom code in the generated structure, with the additional configuration needed to execute it. This functionality allows the user to reuse existing code assets (for this we also call the input directory with custom IaC code as "asset folder").

A further functionality just added to the ICG is the creation, in every output package, of a Gaia-X self-description [8] for the generated code. This allows the generated IaC to be uploaded as a potentially executable service in a Gaia-X repository.

3 CONCLUSIONS

The overall objective of the IDE is to become the main gateway for all developers interested in adopting and following the proposed methodology for deploying DevSecOps applications.

The improvements brought using the IDE to develop applications are: Easy to use IDE, Support for all phases of the software development cycle, Integration with other quality tools.

The full version includes the integration of all the tools specified in the PIACERE project objectives. In addition, the use of the tool has been validated by 3 different use cases in environments of various kinds such as safety on IoT in 5G, public administration, and maritime infrastructures.

ACKNOWLEDGMENTS

This work was supported by the European Union under the H2020 Research and Innovation Programme (PIACERE, grant agreement no: 101000162).

REFERENCES

[1] "PIACERE Project," [Online]. Available: https://piacere-project.eu/.
[2] J. Alonso, R. Piliszek and M. Cankar, "Embracing IaC through the DevSecOps philosophy: Concepts, challenges, and a reference framework," IEEE Software, Special issue: Special Issue on Infrastructure-as-Code Unleashed, pp. 56-62, 2023.
[3] G. Nedeltcheva, A. De La Fuente Ruiz, L. Orue-Echevarria Arrieta, N. Bat and L. Blasi, "Towards Supporting the Generation of Infrastructure as Code Through Modelling Approaches – Systematic Literature Review," in IEEE 19th International Conference on Software Architecture Companion (ICSA-C)., Hawaii, 2022.
[4] M. Chiari, M. De Pascalis and M. Pradella, "Static Analysis of Infrastructure as Code: a Survey," in 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)., Hawaii.
[5] M. Chiari, B. Xiang, G. Nedeltcheva, E. Di Nitto, L. Blasi, D. Benedetto and L. Niculut, "DOML – A New Modelling Approach To Infrastructure-as-Code," in 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023), Zaragoza, 2023.

[6] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, "A fast elitist nondominated nondominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in Parallel Problem Solving from Nature PPSN VI: 6th International Conference, Paris, 2000.

[7] K. Deb and H. Jain, "An evolutionary many-objective optimization," IEEE transactions on evolutionary algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints, vol. 18, no. 4, pp. 577-601, 2013.

[8] "Self-Description of Resources, Service Offerings and Participants within Gaia-X Ecosystems," Gaia-X, 2022. [Online]. Available: https://gaia-x.eu/wp-content/uploads/2022/08/SSI_Self_Description_EN_V3.pdf.