IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS 2017.Doi Number

On the Effectiveness of Tools to Support Infrastructure as Code: Model-Driven versus Code-Centric

Julio Sandobalin[1,2], Emilio Insfran[2], and Silvia Abrahão[2]

[1]Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Quito 17-01-2759, Ecuador
[2]Instituto Universitario Mixto de Tecnología Informática, Universitat Politècnica de València, Valencia 46022, Spain

Corresponding author: Emilio Insfran (e-mail: einsfran@dsic.upv.es).

This research is supported by the Ministry of Science, Innovation, and Universities under grant TIN2017-84550-R (Adapt@Cloud project), Spain. The work of Julio Sandobalin was supported by the Escuela Politécnica Nacional, Ecuador.

ABSTRACT Infrastructure as Code (IaC) is an approach for infrastructure automation that is based on software development practices. The IaC approach supports code-centric tools that use scripts to specify the creation, updating and execution of cloud infrastructure resources. Since each cloud provider offers a different type of infrastructure, the definition of an infrastructure resource (e.g., a virtual machine) implies writing several lines of code that greatly depend on the target cloud provider. Model-driven tools, meanwhile, abstract the complexity of using IaC scripts through the high-level modeling of the cloud infrastructure. In a previous work, we presented an infrastructure modeling approach and tool (Argon) for cloud provisioning that leverages model-driven engineering and supports the IaC approach. The objective of the present work is to compare a model-driven tool (Argon) with a well-known code-centric tool (Ansible) in order to provide empirical evidence of their effectiveness when defining the cloud infrastructure, and the participants' perceptions when using these tools. We, therefore, conducted a family of three experiments involving 67 Computer Science students in order to compare Argon with Ansible as regards their effectiveness, efficiency, perceived ease of use, perceived usefulness, and intention to use. We used the AB/BA crossover design to configure the individual experiments and the linear mixed model to statistically analyze the data collected and subsequently obtain empirical findings. The results of the individual experiments and meta-analysis indicate that Argon is more effective as regards supporting the IaC approach in terms of defining the cloud infrastructure. The participants also perceived that Argon is easier to use and more useful for specifying the infrastructure resources. Our findings suggest that Argon accelerates the provisioning process by modeling the cloud infrastructure and automating the generation of scripts for different DevOps tools when compared to Ansible, which is a code-centric tool that is greatly used in practice.

INDEX TERMS Infrastructure as Code, DevOps, Model-Driven Engineering, Controlled Experiments, Crossover Design, Linear Mixed Model.

1. INTRODUCTION

One of the most critical challenges in many of today's organizations is how to deliver a new idea or software artifact to customers as fast as possible. Furthermore, since requirements and timelines are constantly changing, owing principally to time-to-market, the information exchanged between the development team and operation staff must be accurate, readily available, easily found and, ideally, delivered continuously in real-time. In order to confront these challenges, a new movement denominated as DevOps (Development and Operations) is promoting the continuous collaboration between developers and operations staff by means of a set of principles, practices and tools so as to optimize the software delivery time [1]. DevOps implies a significant transformation in IT culture, focusing on rapid IT service delivery through the adoption of agile methodologies and lean practices in the context of a system-oriented approach [2]. In this context, software deployments are typically a huge source of problems and garner much

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

attention from management when releases are delivered late or a critical defect makes it to production [3]. Furthermore, because the deployment process is the boundary between the developers and operations staff in the software delivery cycle, practitioners recommend starting the DevOps journey with this process [3]. In this scenario, automating the provision of the infrastructure accelerates the deployment process in the software delivery cycle. Practitioners and researchers are consequently using the Infrastructure as Code (IaC), which is an approach for infrastructure automation based on practices originating from software development that emphasizes the use of consistent and repeatable routines for infrastructure provisioning [4]. The idea behind the IaC approach is that of both writing and executing code in order to define, deploy and update the infrastructure [5].

Cloud Computing has simultaneously become the primary pay-as-you-go model used by practitioners and researchers to obtain an infrastructure in a short time. Cloud Computing is composed of hardware-based services, in which hardware management is highly abstracted and the infrastructure capacity is highly elastic [6]. According to Brikman [5], the use of cloud computing along with the IaC approach is leading to certain changes, such as:

- Rather than managing data centers, many companies are moving to the cloud.
- Rather than investing heavily in hardware, many operations teams are spending all their time working on software.
- Rather than racking servers and plugging in network cables, many sysadmins are writing code.

The DevOps community provides a considerable number of IaC tools with which to orchestrate cloud infrastructure provisioning. In this scenario, developers and operation staff use definition files or scripts to specify the creation, updating and execution of the cloud infrastructure resources. However, since IaC is based on software development practices, it is possible to use different approaches, such as code-centric or model-driven development, to write/model and execute the infrastructure resources on various cloud platforms. We identify two stages in the IaC process for reasons of understanding: definition and provisioning. The former writes/models the infrastructure resources that will be provisioned on a cloud platform, while the latter employs IaC tools to execute the infrastructure and hence orchestrate cloud infrastructure provisioning. In this work, we focus on the IaC approach in terms of defining the cloud infrastructure resources to be provided.

In a previous work [7], we proposed an infrastructure modeling approach and tool called Argon for cloud provisioning, which leverages model-driven engineering and supports the IaC approach. The main contributions of our approach are: i) it abstracts the complexity of managing the particularities of different cloud providers in order to define the required infrastructure by using a domain-specific modeling language called ArgonML (Argon Modeling Language), and ii) it generates the IaC scripts using Argon in order to support cloud infrastructure provisioning. However, to the best of our knowledge, there is insufficient empirical evidence as to whether model-driven or code-centric development is the most effective means to support the IaC approach. This is supported by the results of a recent Systematic Mapping Study (SMS) of IaC research [8]. This study identified only seven empirical studies related to IaC, none of which is focused on evaluating the existing IaC tools. Moreover, all the studies are individual experiments. According to Carver *et al.* [9] and Stanley [10], experiments in Software Engineering need to be replicated in different contexts, at different times and under different conditions before they can produce generalizable knowledge.

The purpose of this paper is, therefore, to report a family of experiments carried out to compare the effectiveness and the user perceptions of two IaC tools: Ansible [11] (a code-centric tool) and Argon [7] (a model-driven tool) in terms of their support for the definition of cloud infrastructure resources. Note that the IaC approach has two stages (i.e., definition and provisioning), but that in this paper we focus solely on the definition stage. We do not consider the provisioning stage since, once the IaC scripts have defined the cloud infrastructure resources, there is no difference at the provisioning level, regardless of whether the scripts were obtained by using a model-driven or a code-centric approach. The cloud infrastructure resources are additionally updated by using the definition (i.e., updating the model or script) and provisioning (i.e., updating the infrastructure in the cloud) stages. The experiments were conducted with 67 Computer Science Master's and Bachelor's degree students at the Universitat Politècnica de Valencia in Spain.

According to Santos *et al.* [12], at least three experiments should be included to be considered a family of experiments. In this paper, we therefore present a family of three experiments with which to assess the effectiveness of Argon versus Ansible as regards supporting the IaC approach. This study makes the following contributions:

1. A baseline experiment and two strict internal replications are presented. Internal replications are experiments carried out by the same experimenters that pursue the same goal as that of the baseline experiment [13]. The value of replications has been widely recognized as a means to achieve a greater validity and reliability of experimental results [12], [14].
2. The data analysis of individual experiments is presented in a unified manner. We used the AB/BA crossover design to configure each experiment and the linear mixed model to analyze the data collected.
3. A meta-analysis aggregating the results obtained after carrying out the individual experiments is presented.
4. A thorough discussion of the results is reported. The practical implications of our results are discussed in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

terms of the IaC approach and from the perspective of both researchers and practitioners.

The theoretical contribution of this work is a TAM-based model for evaluating IaC tools. The practical contributions are the application of this model for evaluating two specific IaC tools (Ansible and Argon) and the empirical evidence regarding which tool is more effective in supporting human users when defining the cloud infrastructure.

This paper is organized as follows. In Section 2, we discuss the literature concerning existing studies that compare model-driven with code-centric approaches and empirical studies related to IaC. In Section 3, we present an overview of the tools being compared in this study, while in Section 4 we present the family of experiments by providing an overview of the baseline experiment along with the design and execution of the two replications. This section also highlights the AB/BA crossover design used in the experiments. In Section 5, we present the data analysis of the individual experiments, whereas the results of the family of experiments are discussed in Section 6. The threats to validity are discussed in Section 7, and our conclusions and future directions are presented in Section 8.

2. RELATED WORK

There has been a considerable amount of interest in managing cloud infrastructure resources in recent years, and several approaches and strategies have emerged to support it. For example, Amazon Web Services provides tools with which to define, manage and execute their infrastructure resources, such as CloudFormation [15] and OpsWorks [16]. On the one hand, the DevOps community has developed several tools whose purpose is to manage the infrastructure provisioning of different cloud providers, such as Ansible [11] and Terraform [17], and tools with which to install and manage software in existing servers, such as Chef [18] and Puppet [19]. On the other, researchers have focused their efforts on improving infrastructure provisioning and software deployment by following different approaches. In particular, the model-driven approach supports automation according to the IaC approach. For instance, some of the research efforts made to manage infrastructure resources, and software deployment based on model-driven techniques, are CloudMF [20], MUSA Deployer [21] and MORE [22]. Additionally, Bernal *et al.* [23] propose a UML-based framework that can be used to model a cloud system, including the underlying infrastructure, the user resource requirements and their interactions with the cloud provider.

Although model-driven techniques are popular in academia, their introduction into industrial practices appears to be slow [24], [25], [26], [27]. Some of the obstacles to this are possibly the difficulty of convincing practitioners of the advantages of Model-Driven Development (MDD) [28], the lack of appropriate tools or the lack of evidence regarding its effectiveness in real-world scenarios. In this respect, the main advantages of MDD are improvements to productivity, portability, maintainability and interoperability [29]. Nevertheless, these claims demand empirical evidence if they are to become facts that may help practitioners adopt an MDD strategy. Researchers have, therefore, started to conduct experiments in order to compare, in this case, model-driven versus code-centric approaches. Some representative studies in this line are those of:

- Martinez *et al.* [30], who compare the model-driven versus code-centric development concerning their potential adoption by junior software developers engaged in the development of the business layer of a Web 2.0 application. In [31], the same authors expand their empirical evidence by conducting another experiment to compare the performance and satisfaction of junior software maintainers while performing maintainability tasks on Web applications with two different development approaches, one of which is model-driven (the OOH4RIA approach), and the other of which is a code-centric approach based on Visual Studio .NET.
- Ricca *et al.* [28], who conducted experiments with UniMod, a state-based tool for model-driven development, which was compared with Java-based code-centric programming in a software maintenance scenario. The goal of the experiment was to analyze the effect on the time required to perform the maintenance tasks, the correctness of the artifacts modified and the efficiency.
- Parra *et al.* [32], who experimented by comparing gestUI, a model-driven method intended to deal with gestures, with a code-centric method whose purpose is to include gesture-based interaction in user interfaces, in order to evaluate the usability of the two methods in terms of effectiveness, efficiency and satisfaction.

The results of the above-mentioned experiments generally show that model-driven approaches improve the participants' performance and perceptions. The participants felt more comfortable when using models and they perceived that model-driven approaches are easy to use and useful in the contexts in which they have been used.

With regard to empirical studies related to IaC, an SMS of IaC research carried out by Rahman *et al.* [8] identified four topics that have been researched in the area of IaC: a framework/tool for IaC, the adoption of IaC, empirical studies and testing. Only seven empirical studies related to IaC have been found. These studies focus on the following topics:

- Testing: publications presenting approaches for the testing of IaC scripts. Ikeshita *et al.* [33] presented a method for the efficient checking of idempotence by combining the testing and static verification approaches. The IaC scripts are transformed into a formal model and a graph that has paths representing all the test cases required to judge which path of the execution graph is likely to be redundant is constructed. In contrast,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

Hummer *et al.* [34], [35] assessed the idempotence of IaC scripts using a model-based testing framework in which an abstracted system model was utilized to derive state transition graphs that were subsequently employed as a basis on which to systematically generate test cases.

- Co-evolution: this is related to the study of the co-evolution of IaC scripts with other software artifacts. In this context, Jiang *et al.* [36] performed an empirical study of 256 OpenStack projects to assess the co-evolution relationship between the IaC scripts and the other categories of files in a project, i.e., source code, test code and build scripts.
- Code quality: this is related to the code quality of IaC scripts. Sharma *et al.* [37] analyzed 4,621 Puppet repositories containing 8.9 million lines of code in order to assess and detect implementation and configuration smells. They consequently proposed a catalog of 13 implementations and 11 design configuration smells, in which each smell violates recommended best practices for configuration code. In contrast Weiss *et al.* [38] proposed an interactive system with which to evaluate and repair system configurations and hence bridge the gap between the shell and system configuration languages. They used imperative configuration repair, which is a program synthesis-based technique that allows an IaC script to be automatically repaired given a sequence of shell commands to guide the desired system state.
- Practitioners' survey: Parnin *et al.* [39] carried out a survey at the Continuous Deployment Summit on the Facebook campus in July 2015. This summit focused on 10 adages, which represent a working set of approaches and beliefs that guide current practice and establish a tangible target for empirical validation by the research community. It is worth mentioning that "configuration is code" is an adage that enables the continuous deployment of infrastructure resources and software applications.

Most of the existing empirical studies assessing model-driven versus code-centric approaches provide a global view of the benefits of model-driven approaches. However, to the best of our knowledge, only a few studies [33], [34], [35], [36], [37], [38], [39]) have evaluated the IaC approach in an empirical manner. This is in line with the results of the previously mentioned SMS of IaC research [8]. This SMS identified research works that focus on assessing both the IaC scripts and the repositories in which these scripts are stored. However, these studies do not compare existing IaC tools systematically, nor do they present controlled experiments involving human participants. The authors of the SMS identified 32 IaC-related publications from 9,387 search results and, the number of publications is, therefore, low when compared with other studies. The authors provide two possible explanations for this: (1) IT organizations have not adopted IaC on a wide scale and, as a result, empirical studies related to their experiences and challenges have not been reported; and (2) IT organizations that have adopted IaC are not open as regards sharing their experiences.

In order to improve the body of knowledge regarding empirical studies in IaC, this paper presents a family of three controlled experiments carried out to assess the effectiveness of tools with which to support the IaC as regards model-driven and code-centric approaches.

3. TOOLS BEING COMPARED

This section provides an overview of the infrastructure provisioning tools selected as treatments. We selected Argon [7] because this tool has been proposed in order to abstract the complexity of working with different tools for infrastructure provisioning. It implements a domain-specific language so as to model the characteristics of the cloud infrastructure and provides transformation engines with which to automate the infrastructure provisioning for different cloud providers. We selected Ansible [11] as the control treatment because it is currently widely used by industry and academia to define the infrastructure in scripts and then deploy it in different cloud providers [40], [41]. Both Ansible and Argon define the cloud infrastructure, although their notations and abstraction levels are significantly different.

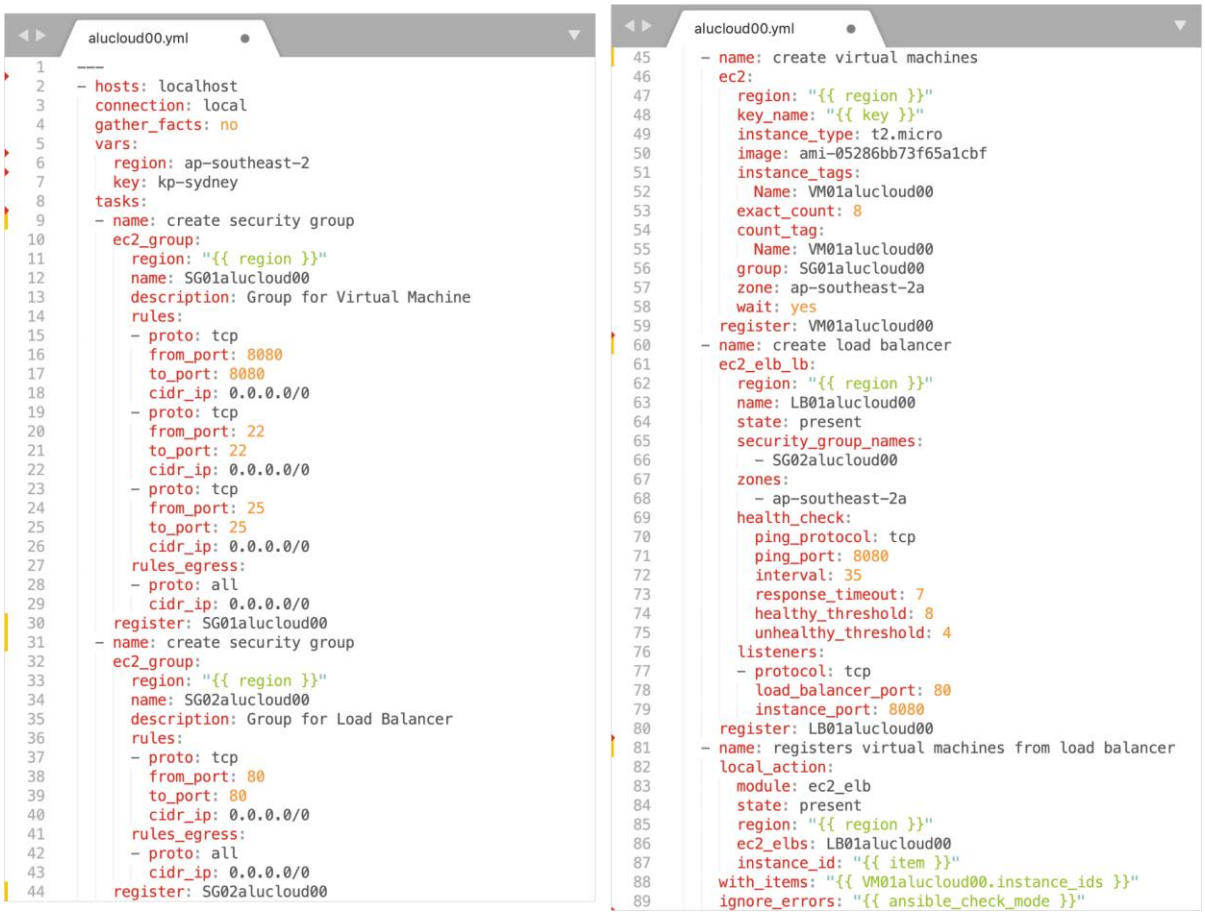
3.1. ANSIBLE: A CODE-CENTRIC TOOL

The purpose of Ansible [11] is to both define the final state of infrastructure resources and orchestrate cloud infrastructure provisioning. Ansible exposes a domain-specific language that is used to describe the state of infrastructure resources, which signifies writing code with which to specify each infrastructure element for a particular cloud provider.

Ansible uses a script called *playbook* that describes the provisioning configuration, along with an ordered list of *tasks* to be performed by a specific cloud provider. The *playbook* syntax is built on top of YAML, which is a data format language that was designed to be easy to read and write. Each *playbook* is composed of one or more *plays* in a list. A *play* maps some well-defined tasks onto a group of remote hosts, that is, virtual machines in a specific cloud provider. Finally, a *task* is a call to an Ansible module to create and manage infrastructure resources in remote hosts.

Fig. 1 shows an excerpt of the *playbook* (script) in Ansible for the MODAFIN company (MODAFIN for short) [42]. MODAFIN specializes in IT applications for financial services, and its main product line is a proprietary solution for stock market operations, cash administration and lending management. MODAFIN is concerned with the exponential traffic of requests received by software applications installed in Amazon Web Services. MODAFIN has solved this issue by using a load balancer to distribute the workload among several virtual machines.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access



```
alucloud00.yml
1  ---
2  - hosts: localhost
3    connection: local
4    gather_facts: no
5    vars:
6      region: ap-southeast-2
7      key: kp-sydney
8    tasks:
9    - name: create security group
10     ec2_group:
11       region: "{{ region }}"
12       name: SG01alucloud00
13       description: Group for Virtual Machine
14       rules:
15       - proto: tcp
16         from_port: 8080
17         to_port: 8080
18         cidr_ip: 0.0.0.0/0
19       - proto: tcp
20         from_port: 22
21         to_port: 22
22         cidr_ip: 0.0.0.0/0
23       - proto: tcp
24         from_port: 25
25         to_port: 25
26         cidr_ip: 0.0.0.0/0
27       rules_egress:
28       - proto: all
29         cidr_ip: 0.0.0.0/0
30     register: SG01alucloud00
31   - name: create security group
32     ec2_group:
33       region: "{{ region }}"
34       name: SG02alucloud00
35       description: Group for Load Balancer
36       rules:
37       - proto: tcp
38         from_port: 80
39         to_port: 80
40         cidr_ip: 0.0.0.0/0
41       rules_egress:
42       - proto: all
43         cidr_ip: 0.0.0.0/0
44     register: SG02alucloud00
```

```
alucloud00.yml
45   - name: create virtual machines
46     ec2:
47       region: "{{ region }}"
48       key_name: "{{ key }}"
49       instance_type: t2.micro
50       image: ami-05286bb73f65a1cbf
51       instance_tags:
52         Name: VM01alucloud00
53       exact_count: 8
54       count_tag:
55         Name: VM01alucloud00
56       group: SG01alucloud00
57       zone: ap-southeast-2a
58       wait: yes
59     register: VM01alucloud00
60   - name: create load balancer
61     ec2_elb_lb:
62       region: "{{ region }}"
63       name: LB01alucloud00
64       state: present
65       security_group_names:
66         - SG02alucloud00
67       zones:
68         - ap-southeast-2a
69       health_check:
70         ping_protocol: tcp
71         ping_port: 8080
72         interval: 35
73         response_timeout: 7
74         healthy_threshold: 8
75         unhealthy_threshold: 4
76       listeners:
77       - protocol: tcp
78         load_balancer_port: 80
79         instance_port: 8080
80     register: LB01alucloud00
81   - name: registers virtual machines from load balancer
82     local_action:
83       module: ec2_elb
84       state: present
85       region: "{{ region }}"
86       ec2_elbs: LB01alucloud00
87       instance_id: "{{ item }}"
88     with_items: "{{ VM01alucloud00.instance_ids }}"
89     ignore_errors: "{{ ansible_check_mode }}"
```

FIGURE 1. The Ansible playbook for the MODAFIN Company.

The *playbook* (see Fig. 1) begins with --- (line 1), which indicates the start of an infrastructure script for Ansible. A *play* has three sections: the *host* section, the *variables* section, and the *task* section. The host section defines in which remote host (line 2) of a cloud provider the *play* will be executed, and how the *play* will be implemented (line 3).

The variable section defines the variables that will be used for the entire play on remote hosts. In this case, we define the region variable (line 6) to specify the region code in which the infrastructure will be provisioned in Amazon Web Services, along with the key pair (line 7) used by Ansible to obtain access to the region selected.

In the task section, all infrastructure elements detailed will be executed, in order, one at a time, against a remote host matched in the host section. The first task specifies a security group for virtual machines (from line 9 to 30), which works like a firewall to enable both inbound connections through port 8080, port 22 and port 25, in addition to all outbound connections. The second task details a security group for the load balancer (from line 31 to 44), which enables both inbound connections through port 80 and all outbound connections. The third task defines a virtual machine (from line 45 to 59) in which all the hardware characteristics are specified, such as the type of instance (processor, RAM, storage, etc.), the image code (operating system) or the region and availability zone in which the virtual machines will be deployed. The fourth task describes a load balancer (from line 60 to 80), which distributes the workload among several virtual machines. The load balancer has a health check element (from line 69 to 75), which validates that virtual machines connected to the load balancer are available. The load balancer also has a listener element (from line 76 to 79), which checks connection requests made to the load balancer. Finally, the fifth task registers (from line 81 to 89) all virtual machines created for the load balancer in order to distribute the workload among them.

3.2. ARGON: A MODEL-DRIVEN TOOL

Argon [7] is an infrastructure modeling tool for cloud provisioning that leverages model-driven engineering and supports the IaC approach. Argon abstracts the complexity of working with different cloud providers through the use of a domain-specific modeling language (DSML). Moreover, Argon uses the DSML to model the infrastructure resources and then generate the corresponding scripts with which to manage different tools (e.g., Ansible, Terraform, etc.) for cloud infrastructure provisioning. Furthermore, Argon was

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

built by following a plugin-based architecture, and it therefore works in the Eclipse Modeling Framework [43].

Fig. 2 shows an infrastructure model modeled with ArgonML for the CEC University (CEC for short) [6]. CEC offers massive open online courses by means of a virtual platform. The problem with CEC's virtual platform is the high demand for specific courses, which is causing an overload in the servers in which the courses are running.

CEC could purchase new servers to create a cluster and thereby solve its problem. However, these new servers will not work if there is no demand for courses. The solution is, therefore, to migrate the infrastructure toward Amazon Web Services and hence pay for using the infrastructure deployed. In this scenario, CEC should use a load balancer to distribute workloads among several virtual machines.

Fig. 2A shows the infrastructure model in which a virtual machine (VM01alucloud00) element is connected to a security group (SG02alucloud00) that works as a firewall. Note that the virtual machine (VM01alucloud00) element defines the number of virtual machines to be created. The security group (SG02alucloud00) has three inbound rules (Port_9090, Port_443 and Port 22) and one outbound rule (Port_ALL). The inbound rules enable incoming connections to virtual machines, and the outbound rule allows outgoing connections from the virtual machine to external servers. In addition, a zone (ue-west-2b) element specifies the availability zone in which virtual machines (VM01alucloud00) will be deployed. The load balancer (LB01alucloud00) distributes the workload among virtual machines (VM01alucloud00) and has a listener (L01alucloud00) element and a health check (HC01alucloud00) element. The former checks connection requests to the load balancer, while the latter validates that virtual machines connected to the load balancer are available. In addition, the load balancer (LB01alucloud00) has a zone (ue-west-2b) element, which specifies the availability zone in which it will be deployed. Finally, the load balancer (LB01alucloud00) has a security group (SG01alucloud00) with one inbound rule (Port_80) and one outbound rule (Port_ALL).

Furthermore, Fig. 2B shows the palette of the infrastructure elements that are used to model an infrastructure model. Note that each infrastructure element has its own properties. Fig. 2C shows the infrastructure model properties in which the *File name* property specifies the script name that will be generated, the *Key name* property allows the key pair to be written in order to enable secure access to Amazon Web Services, and the *Region* property allows the definition of the region code in which infrastructure resources will be provisioned.

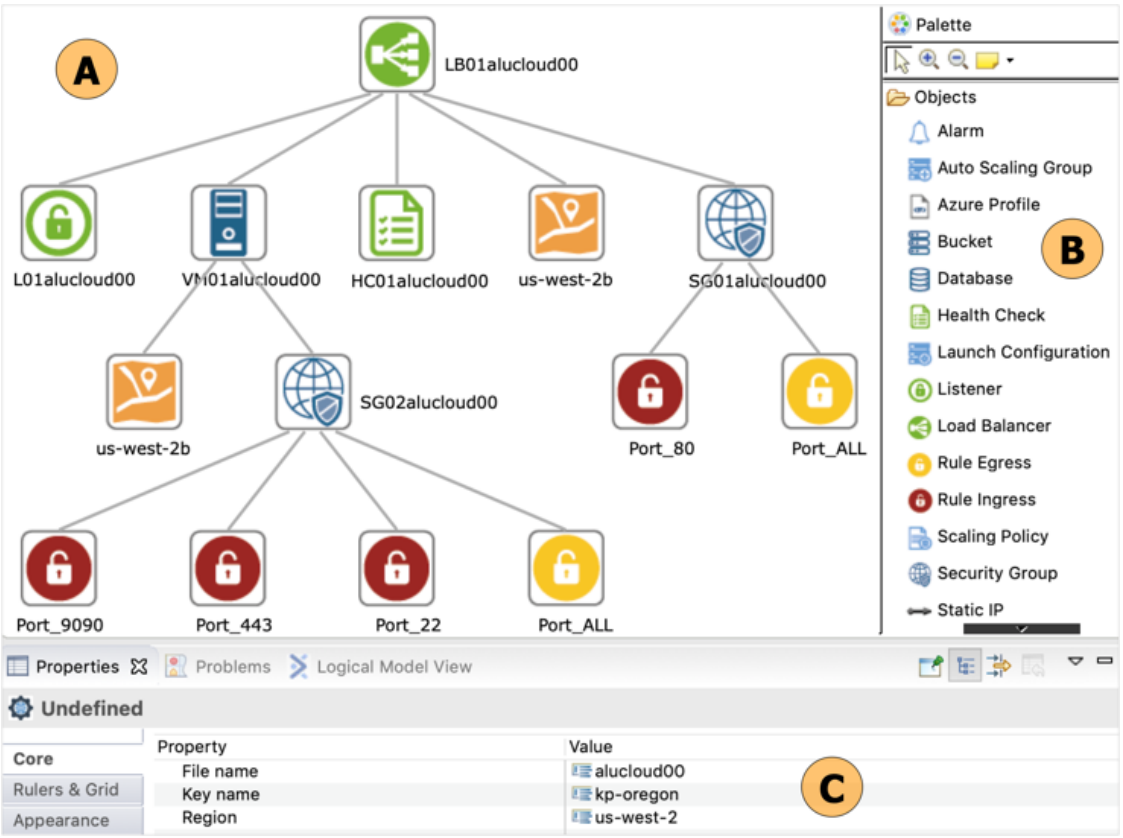


FIGURE 2. The infrastructure model for the CEC University in Argon.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

3.3. COMPARISON OF TOOLS

Both Ansible and Argon provide support in order to define the final state of cloud infrastructure resources. On the one hand, Ansible allows the specification of infrastructure elements in a script using a scripting language (i.e., YAML). On the other, Argon allows the modeling of the infrastructure resources in an infrastructure model and, from this model, generates the corresponding scripts required to manage different provisioning tools such as Ansible, Terraform, etc.

Table 1 presents a mapping between the infrastructure elements of Ansible and Argon. In this comparison, it is worth mentioning that both Ansible and Argon can define the infrastructure for different cloud providers. In this work we, therefore, focus on specifying the cloud infrastructure for Amazon Web Services.

Argon provides more infrastructure elements than Ansible because it proposes a generic infrastructure model in which all elements are specified in detail. However, Ansible includes one or more infrastructure elements in a module. For instance, the ec2_group module allows the configuration of a security group, along with its inbound and outbound rules. Moreover, the ec2_elb_lb module allows the configuration of a load balancer, along with its listener and health check elements. The remaining elements are matched, one by one, between Ansible modules and Argon infrastructure elements.

TABLE 1. COMPARING ANSIBLE MODULES AND ARGON INFRASTRUCTURE ELEMENTS

Argon	Ansible
ec2	Virtual Machine
ec2_group	Security Group
-	Inbound Rule
-	Outbound Rule
ec2_eip	Static IP
ec2_elb_lb	Load Balancer
-	Listener
-	Health Check
rds	Database
s3_bucket	Bucket
ec2_lc	Launch Configuration
ec2_asg	Auto Scaling Group
ec2_scaling_policy	Scaling Policy
ec2_metric_alarm	Alarm

4. THE FAMILY OF EXPERIMENTS

According to Basili *et al.* [14], a family of experiments is a group of experiments that pursue the same goal and whose results can be combined into joint results that are potentially more mature than those that can be achieved in isolated experiments. In addition, Santos *et al.* [12] contribute to this definition by adding that at least two treatments should be compared within all experiments on a common response variable and at least three experiments should be included within the family. We therefore conducted a family of three controlled experiments in order to provide empirical findings on the comparison of two treatments, that is, Ansible versus Argon. The family of experiments was defined according to the framework proposed by Ciolkowski *et al.* [44], and each experiment was designed and executed by following the experimental process proposed by Wohlin *et al.* [45].

4.1. GOAL

A family of experiments has to set a goal in order to allow the effective analysis of the individual results [44] and to define the scope of the experiments. According to the Goal-Question-Metric (GQM) paradigm [46], the goal of our family of experiments is to *analyze* the definition of cloud infrastructure resources specified by Ansible and Argon *for the purpose of* assessing them *with respect to their* effectiveness, efficiency, perceived ease of use, perceived usefulness, and intention of use *from the viewpoint of* novice software engineers *in the context of* Computer Science Master's and Bachelor's degree students.

Although cloud computing practitioners would have been preferable, we focused on the profile of novice software engineers since we aim to provide an infrastructure modeling tool for cloud provisioning that will help less experienced users to specify high-quality infrastructure models. The research questions addressed are:

- **RQ1:** Which IaC tool is more effective when defining the cloud infrastructure? This is concerned with the correctness of the definition of the cloud infrastructure resources.
- **RQ2:** Which IaC tool is more efficient when defining the cloud infrastructure? This is concerned with the correctness of the definition of the cloud infrastructure resources in relation to the time spent.
- **RQ3:** Which IaC tool is perceived to be easier to use?
- **RQ4:** Which IaC tool is perceived to be more useful?
- **RQ5:** Which IaC tool is more intended to be used?

4.2. CONTEXT SELECTION

The context of this study is the definition of two cloud infrastructures created by novice software engineers. The context is defined by (i) the experimental objects (i.e., the infrastructure resources to be specified); (ii) the IaC tools selected, and (iii) the selection of participants.

4.2.1. EXPERIMENTAL OBJECTS

The infrastructure resources to be specified were selected and adapted from literature:

- O1 – *MODAFIN Company* [42]: the purpose is to solve the exponential traffic of requests received by software applications installed in Amazon Web Services through the use of a load balancer, which distributes the workload among several virtual machines. An excerpt of the MODAFIN *playbook* (script) in Ansible is shown in Fig. 1.
- O2 – *CEC University* [7]: the purpose is to pay for the use of an infrastructure deployed in Amazon Web Services in order to solve the high demand for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

specific courses that is causing an overload on the virtual platform (servers) located in the CEC University. In this case, a load balancer should distribute the workload among virtual machines, which will be created or destroyed on demand. An excerpt of the infrastructure model of the CEC system modeled with Argon is shown in Fig. 2.

4.2.2. IAC TOOLS

Ansible is widely used by practitioners to define, update and execute the infrastructure resources in cloud computing. It uses the YAML scripting language to define the instructions necessary to specify infrastructure elements in a playbook (i.e., script). In this case, the participants were asked to write the instruction needed to specify a load balancer that distributes the workload among virtual machines (e.g., Fig. 1 shows the specification for the MODAFIN system). Moreover, both the load balancer and the virtual machines had to be connected to a security group. First, the playbook had to begin with three dashes that would indicate the start of the script, after which the participants had to specify remote hosts on which the script would be run, and how it would be executed. They also had to define the variables to be used throughout the entire playbook. Finally, they had to define Ansible modules with which to create and manage the infrastructure elements.

Argon, meanwhile, provides a domain-specific modeling language with which to model the infrastructure resources. Argon was developed by our research group and is, therefore, an experimental prototype. In this case, the participants were asked to model a load balancer that would distribute the workload among several virtual machines (e.g., Fig. 2 shows the infrastructure model for the CEC University). In addition, the participants had to model infrastructure resources, such as security groups, zone, health check and listener. First, an infrastructure model had to be created in the Eclipse Modeling Framework, after which each element had to be dragged from the palette of infrastructure elements and dropped onto the canvas. It was also necessary to make the connection among elements. Finally, it was necessary to fill in both the infrastructure diagram properties and the infrastructure elements properties.

4.2.3. SELECTION OF PARTICIPANTS

According to Kitchenham *et al.* [47], students are the next generation of software professionals and are, therefore, relatively close to the population of interest. Furthermore, Höst *et al.* [48] investigated the appropriateness of final-year students as subjects and concluded that, if well trained, they can be considered as appropriate experimental subjects. The following group of participants was consequently selected:

- Master's students enrolled on the Software Engineering Master's degree program at the Universitat Politècnica de València (UPV). These students attended the Empirical Software Engineering course throughout the academic year 2017-18. The purpose of this course is to provide students with the knowledge required to plan, conduct and present the results of empirical studies.
- Undergraduate students enrolled on the Computer Engineering Bachelor's degree at the UPV. These students attended the Model-Driven Software Development course throughout the academic year 2017-18. The purpose of this course is to provide students with the knowledge of the construction of software models at different levels of abstraction as the main artifacts in software development.
- Undergraduate students enrolled on the Computer Engineering Bachelor's degree at the UPV. These students attended the Requirements Engineering course throughout the academic year 2017-2018. The purpose of this course is for students to understand both the needs of users and the domain and context in which the software system will be used to elicit, analyze, negotiate and document software requirements.

We did not establish a classification of the participants on the basis of their cloud infrastructure provisioning experience, since neither the Bachelor's nor the Master's degree students had a previous background in defining cloud infrastructure provisioning.

4.3. DESIGN OF INDIVIDUAL EXPERIMENTS

Fig. 3 summarizes the family of experiments. The figure includes the context of each experiment, the number of participants involved and the place in which the experiments took place. The figure also shows the order in which the experiments were carried out.

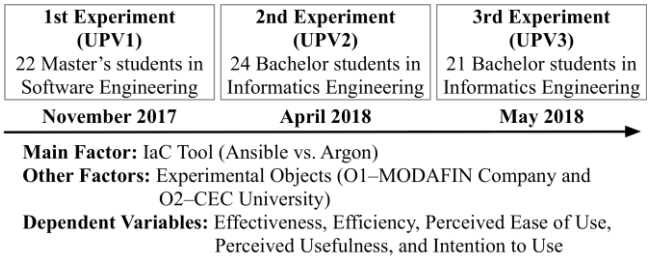


FIGURE 3. Overview of our family of experiments.

Since experimental conditions are difficult to control, one way in which to satisfy the statistical requirements of replications is by running internal replications (in the same place and by the same experimenters) [49]. Having more internal replications of the same experiment considerably reduces the Type I error, and identical replications are also required in order to be able to estimate the effect size under study [49]. A Type I error (α-error, false positives) occurs when the null hypothesis (H_0) is rejected in favor of the alternative hypothesis (H_1), when the 'null' hypothesis is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

actually true. The effect size indicates the magnitude of the observed effect or relationship between variables.

Our family was, therefore, composed of the baseline experiment conducted with the Master's degree students (UPV1). Before conducting this experiment, a pilot experiment was conducted with 12 Ph.D. students to evaluate the experimental materials regarding the experimental procedures, instructions and the task completion time. It should be noted that Ph.D. students played no part in the controlled experiments.

In order to verify the results obtained in the baseline experiment, we conducted two strict replications according to the guidelines proposed in [50]. The first replication was conducted with undergraduate students enrolled on the Model-Driven Software Development course (UPV2), while the second replication was also conducted with undergraduate students, but those enrolled on the Requirements Engineering course (UPV3). These replications were operational, as we varied some dimensions of the experimental configuration [13], particularly the population. This allowed us to verify whether the results were independent of the participants' profile.

4.3.1. BASELINE EXPERIMENT (UPV1)

The objective of this experiment was to evaluate whether the participants using our proposed tool (i.e., Argon) to model an infrastructure model would perform better (in terms of effectiveness and efficiency) and report better perceptions than when using Ansible.

4.3.1.1. CONTEXT SELECTION

We used the experimental objects explained in Section 4.2.1, along with the IaC tools described in Section 4.2.2. Table 2 presents the infrastructure requirements used to specify the cloud infrastructure. The requirements were taken and adapted from literature [7], [42]. Note that the requirements are independent of both the experimental objects and the IaC tools. Moreover, the experimental objects are from different application domains, which do not require specialized knowledge to understand them, but have a similar complexity.

4.3.1.2. PARTICIPANTS

The experiment involved 22 Master's students enrolled on the Software Engineering Master's degree program at the Universitat Politècnica de València. The participants' prior knowledge and expertise were evaluated through the use of a pre-questionnaire. 14 participants reported that they had professional experience in software development, varying between 1 to 4 years, with an average of 2 years, but that they had no previous knowledge of cloud computing or the infrastructure provisioning process. The participants were chosen by means of convenience sampling. They attended the Fall 2017 course on Empirical Software Engineering with a focus on evaluating infrastructure provisioning approaches. The participants were asked to carry out the experimental task as part of the laboratory exercises of the course.

TABLE 2. REQUIREMENTS TO SPECIFY THE CLOUD INFRASTRUCTURE.

No.	Requirement
R1	The infrastructure should be deployed in a specific region of Amazon Web Services.
R2	A security group should enable TCP incoming connections to virtual machines through three specific ports. Moreover, the security group should enable all outgoing connections from virtual machines.
R3	A security group should enable TCP incoming connections to a load balancer through port 80. Moreover, the security group should enable all outgoing connections from the load balancer.
R4	A set of virtual machines should be launched in a specific availability zone in a selected region of Amazon Web Services. Each virtual machine should have an explicit virtual CPU and RAM. Moreover, each virtual machine should have a specific operating system and web server installed.
R5	A load balancer should distribute the workload among virtual machines. The load balancer should respond to client requests and validate that all virtual machines connected are available. Moreover, the load balancer should work in a specific availability zone in a selected region of Amazon Web Services.
R6	The load balancer uses a health check element to validate the state of virtual machines connected through check intervals using the TCP protocol and a specific port number. It is necessary to wait for a particular time (in seconds) to notify an error check. The load balancer should receive a precise amount of consecutive errors in order to change a virtual machine to an "unhealthy" state, whereas it should receive a specific number of state verification probe successes in order to change a virtual machine to a "healthy" state.
R7	The load balancer uses a listener element to respond to client requests through the use of the TCP protocol and the port 80, in addition to distributing all workloads to virtual machines by means of a specific port.
R8	Virtual machines should be registered to the load balancer, in such a way that the load balancer can distribute the workload among all available virtual machines.

4.3.1.3. SELECTION OF VARIABLES

The independent variable (or factor) was the IaC tool, which has two levels: Ansible and Argon. The former is a code-centric tool used to specify the infrastructure resources in scripts, whereas the latter employs a model-driven approach to model the infrastructure resources.

The Method Evaluation Model (MEM) [51] was used as a theoretical basis on which to design the experiment. According to this model, there are two types of dependent variables: performance-based variables that measure how well the participants perform the experimental task (i.e., define the infrastructure resources), and perception-based variables that measure the participants' beliefs and attitudes toward the use of the IaC tools.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

There are two performance-based variables:

- **Effectiveness,** which measures the degree to which an IaC tool achieves its objectives.
- **Efficiency**, which measures the effort required to use an IaC tool.

There are three perception-based variables:

- **Perceived Ease of Use (PEOU)**, which refers to the degree to which a participant believes that learning and using a particular IaC tool will be effortless.
- **Perceived Usefulness (PU)**, which refers to the degree to which a participant believes that a particular IaC tool will be effective in achieving its indented objectives.
- **Intention to Use (ITU)**, which is the extent to which a participant intends to use a particular IaC tool.

In order to operationalize the performance-based variables, we used the ISO/IEC 9126-4 [52] to obtain the metrics required to measure effectiveness and efficiency. Moreover, in order to operationalize the perceptions-based variables, we adapted the measurement instrument proposed by Davis *et al.* [53] to measure the perceived ease of use, perceived usefulness and intention of use. Table 3 summarizes the metrics used to measure each dependent variable.

TABLE 3. SUMMARY OF THE DEPENDENT VARIABLES.

Name	Measure	Scale
Effectiveness	$\frac{\text{Number of Correct Requirements}}{\text{Total Number of Requirements}}$	Ratio
Efficiency	$\frac{\text{Effectiveness}}{\text{Time}}$	Ratio
PEOU	5-point Likert scale	Ordinal
PU	5-point Likert scale	Ordinal
ITU	5-point Likert scale	Ordinal

Since the experimental task is a set of requirements to be coded in Ansible or modeled in Argon we, therefore, calculated *effectiveness* as the number of correctly defined requirements divided by the total number of requirements proposed. This reflects the correctness of the definition of cloud infrastructure resources specified by the participants. Moreover, *efficiency* was calculated as the Effectiveness of that participant divided by the Time that she/he took to perform the task. This reflects the proportion of requirements achieved for each unit of time. Efficiency increases with increasing effectiveness and a reduction in task time. In fact, we defined the following aggregation metric to decide whether or not a task (i.e., set of requirements) was valid:

- **All-or-nothing metric:** we considered that a requirement was correct only if it was defined correctly. A requirement, therefore, had only two possible values: success or failure (1 or 0).

We proposed the *all-or-nothing* metric because the infrastructure resources defined in the experiment had to work in a particular cloud provider, and the infrastructure would not, therefore, be provisioned in the cloud provider if any of the requirements had been incorrectly defined.

In contrast, in order to operationalize the perception-based variables, we used a survey questionnaire adapted from [51] to measure PEOU, PU and ITU. The questionnaire items were formulated by using a 5-point Likert scale and adopting the opposing-statement question format. Various items within the same construct group were randomized to prevent systemic response bias. Table 4 presents the items employed to measure these variables (See Appendix A).

TABLE 4. ITEMS USED TO MEASURE THE PERCEPTION-BASED VARIABLES.

Type	Question statement
PEOU1	I found the procedure for using the IaC tool complex and difficult to follow.
PEOU2	Overall, I found the IaC tool difficult to use.
PEOU3	I found the IaC tool easy to learn.
PEOU4	I found it difficult to define the cloud infrastructure with the IaC tool.
PEOU5	I found the use of the IaC tool clear and easy to understand.
PU1	I believe that this IaC tool would reduce the effort required to define the cloud infrastructure.
PU2	Overall, I found the IaC tool useful.
PU3	The cloud infrastructure defined using this IaC tool would be more difficult to understand.
PU4	Overall, I think this IaC tool does not provide an effective solution to define the cloud infrastructure.
PU5	Overall, I think this IaC tool makes an improvement to the cloud infrastructure definition process.
PU6	This IaC tool would make it easier for practitioners to define the cloud infrastructure.
PU7	Using this IaC tool would make it easier to communicate the cloud infrastructure definition to other practitioners.
ITU1	I would recommend this IaC tool to define the cloud infrastructure.
ITU2	If I am working at a company in the future, I would like to use this IaC tool to define the cloud infrastructure.
ITU3	It would be easy for me to become skillful in using this IaC tool to define the cloud infrastructure.

4.3.1.4. FORMULATION OF HYPOTHESES

We formulated the null hypotheses on the basis of the dependent variables. It is worth mentioning that the experiment aims to assess the definition of infrastructure resources rather than assessing the tools holistically.

The null hypotheses of the experiment can be summarized as follows:

- **$H1_0$:** Effectiveness (Ansible) = Effectiveness (Argon)
- **$H2_0$:** Efficiency (Ansible) = Efficiency (Argon)
- **$H3_0$:** PEOU (Ansible) = PEOU (Argon)
- **$H4_0$:** PU (Ansible) = PU (Argon)
- **$H5_0$:** ITU (Ansible) = ITU (Argon)

The goal of the statistical analysis was to reject the null hypotheses and possibly accept the alternative ones (e.g., $H1_1 = \neg H1_0$). All the hypotheses are two-sided because we did

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

not postulate that any effect would occur as a result of IaC tool usage.

4.3.1.5. DESIGN

The experiment was designed as an AB/BA crossover design, which has one factor and two treatments. In this context, the IaC tool employed to specify the cloud infrastructure is the factor, and the two treatments are Ansible and Argon. We chose the AB/BA crossover design because it addresses the issue of small sample sizes and increases the sensitivity of experiments. We followed the guidelines proposed by Vegas *et al.* [54] to define the crossover design and, therefore, used fixed factors as period, sequence and carryover.

TABLE 5. AB/BA CROSSOVER DESIGN USED TO CONFIGURE THE EXPERIMENT.

	Object	O1: MODAFIN		O2: CEC University	
	Period	Period 1		Period 2	
Technique	Sequence	Ansible	Argon	Ansible	Argon
Group 1	Ansible-Argon	X	-	-	X
Group 2	Argon-Ansible	-	X	X	-

Table 5 shows a special type of design called a factorial crossover design, which has the same number of periods as treatments, in which all the participants apply each treatment under study once and once only [54]. In this scenario, it is necessary to differentiate between the concepts of period and session. A period is defined by the application of one treatment by one participant to one experimental object, whereas a session is a portion of time taken by a subject to complete (one or more) experimental tasks [54]. We consequently had two periods, since each participant had to perform two treatments and for reasons of the students' class timetable. We, therefore, carried out one period in a session.

Because we had two periods and two treatments, there were two resulting sequences, that is, Ansible-Argon and Argon-Ansible. In the first period, Group 1 solved the experimental object O1 with Ansible, while Group 2 solved the same experimental object with Argon. In the second period, Group 1 solved the experimental object O2 with Argon, while Group 2 solved the same experimental object with Ansible. In this scenario, we do not believe that there is the possibility that any of the sequences would have improved the experimental results, since both Ansible and Argon define the cloud infrastructure on the basis of different principles and inputs.

Two different experimental objects were employed in the experiment (MODAFIN and CEC University), and each experimental object (system specification) described eight requirements that should be implemented by the participants. According to the experiment design, each experimental object had to be used in a different period. As in the case of sequences, we do not believe that there is a possibility that the order of use of experimental objects would have improved the results of the experiment.

Throughout an experiment, if the effect of one treatment carries on after the treatment is withdrawn, then the response to a second treatment may well be partly owing to the previous treatment, and carryover occurs [54]. In this case, in a crossover design it is complicated to identify the sequence effect in addition to the interaction between period and treatment and the possibility that carryover exists should, therefore, be considered. As a result, we acknowledge the carryover effect in the design stage, and its results will be examined in the analysis stage. Finally, the crossover design is balanced for carryover effects because each treatment followed each of the other treatments an equal number of times [55].

4.3.1.6. OPERATION

The procedure employed to run the experiment matches that of the factorial crossover design chosen. The experiment was conducted in five 2-hour sessions. Fig. 4 shows a summary of how the experiment was conducted. The numbers on the diagram represent the steps in the experimental process and, in this case, represent the sessions. The first 2-hour session (Step 1) was used to teach the foundations of cloud computing and those of Ansible and Argon. Note that we focused on teaching the principal concepts that the participants had to use for the experiment rather than explaining cloud computing or IaC tools (i.e., Ansible and Argon) in detail. The second 2-hour session (Step 2) was employed to allow the participants to get acquainted with Ansible, whereas the third 2-hour session (Step 3) was to enable them to get acquainted with Argon. The participants carried out an exercise similar to the experiment. The first three steps focused on teaching and training how to define the cloud infrastructure, because it was new knowledge that the participants had to acquire.

The first period of the controlled experiment was conducted in the fourth 2-hour session (Step 4), whereas the second period was conducted in the fifth 2-hour session (Step 5). The activities carried out in each period are described below:

- First, the participants filled out a survey related to their knowledge and experience with cloud computing and IaC tools.
- Next, the participants were randomly assigned to Group 1 and Group 2, considering that both groups would have the same number of participants. In the first period, Group 1 solved the experimental object O1 (MODAFIN) with Ansible, while Group 2 solved the same experimental object with Argon. In the second period, Group 1 solved the experimental object O2 (CEC University) with Argon, while Group 2 solved the same experimental object with Ansible.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

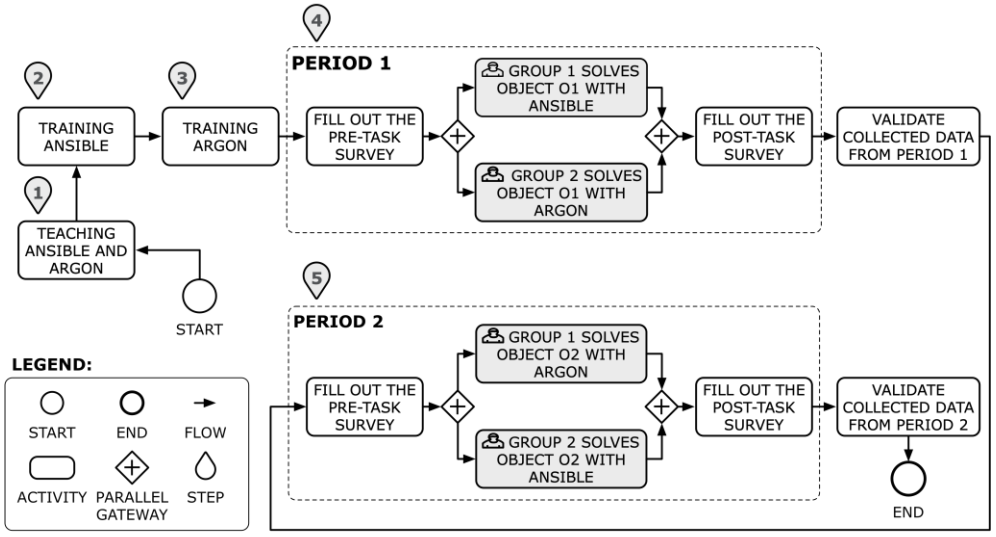


FIGURE 4. **Summary of the operation process of the experiment.**

- Finally, the participants filled out a survey to show their perceptions as regards ease of use, usefulness and intention to use the treatments (i.e., Ansible or Argon). After each period, we checked that the data was complete and whether it had been collected correctly.

4.3.2. SECOND EXPERIMENT (UPV2)

The second experiment in our family was a strict internal replication of the baseline experiment (UPV1). The same experimental protocol was applied but to a different population, signifying that we varied only the participants, while the site, experimenters, design, variables and instrumentation remained the same. We changed only the participants with the purpose of testing the extent to which the experimental results could be generalized. The sample of participants was composed of 24 undergraduate students enrolled on the Model-Driven Software Development course in the third year of their B.Sc., who had consequently obtained knowledge of model-driven techniques. The participants were asked to carry out the experimental task as part of the laboratory exercises on the course.

As in the baseline experiment, it took place in a single room, and no interaction was allowed among the participants. We carried out the AB/BA crossover design of setting two treatments and two periods, and the participants were randomly assigned to two groups.

4.3.3. THIRD EXPERIMENT (UPV3)

The third experiment in our family was a strict replication of the baseline experiment (UPV1). The experimental protocol, site, design, variables, instrumentation and experimenters remained the same. The sample of participants was composed of 21 undergraduate students enrolled on the Requirements Engineering course in the fourth-year of their B.Sc. The participants were asked to carry out the experimental task as part of the laboratory exercises of the course.

The replication took place in a single room, and no interaction was allowed among the participants. We again used the AB/BA crossover design and two treatments with two periods were, therefore, configured and the participants were randomly assigned to two groups.

4.4. EXPERIMENTAL TASK AND MATERIALS

The experimental tasks consisted of specifying the cloud infrastructure using one of the IaC tools selected on two experimental objects. These tasks were structured so as to allow the comparison of both tools in terms of specified infrastructure resources. We provided the participants with the goal and description of the cloud infrastructure and then asked them to specify the corresponding infrastructure resources by following the steps and guidelines of each tool. In this scenario, a set of requirements was provided to the participants in an attempt to exemplify a real problem in which they should use a load balancer to distribute workloads among virtual machines.

In the case of Ansible, the experimental task consisted of: (1) defining the host and variable sections; (2) defining the task section and then specifying the infrastructure resources, and (3) registering or deregistering the components, which signified connecting the infrastructure elements with each other in such a way that they would work with each other on a particular cloud platform. In the case of Argon, the experimental task similarly consisted of: (1) modeling the infrastructure resources; (2) connecting the infrastructure elements to each other, and (3) filling in the property values of each infrastructure element.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

The experimental materials consisted of a set of documents to support the experimental task, training sessions and pre- and post-questionnaires.

The training material included: (1) slides to teach the foundations of Cloud Computing, Amazon Web Services, and Infrastructure as Code, (2) slides to explain the Ansible tool, along with an example of how to specify the infrastructure elements in a script, and (3) slides to explain the Argon tool, along with an illustrative example of how to model the elements in an infrastructure model.

The experimental material that supported the experimental task of each experiment included:

- Four booklets that covered four possible combinations of experimental objects (O1 and O2) and treatments (Ansible and Argon). The purpose of these booklets was to describe each experimental object, along with its requirements, in addition to describing the infrastructure elements, their property values and their relationships.
- One pre-questionnaire with which to collect the participants' knowledge and skills, and one post-questionnaire with which to gather the participants' perceptions as regards ease of use, usefulness and intention to use.
- A guide explaining the steps required to create a playbook, that is, a script for Ansible. This guide included definitions of the sections of the script and snippets of code in order for the participants to use them as a reference to create their code. The objective of the experiment was that the participants would learn how to create a script rather than memorizing snippets of code.
- A guide explaining the steps required to model the cloud infrastructure resources with Argon. This guide included instructions on how to create an infrastructure project and the guidance required to use infrastructure elements and fill in their properties.
- A guide explaining the regions and availability zones for Amazon Web Services. The load balancer and virtual machines have to specify the location in which they will be deployed.
- A guide containing a list of key pairs required to access the regions of Amazon Web Services. In order to deploy the infrastructure resources in a particular region, it is necessary to specify its code.
- A guide containing a list of the instance type codes, which have the hardware characteristics of virtual machines.
- A guide containing a list of the image codes, that is, the operating system for a virtual machine in Amazon Web Services.
- A virtual machine configured and tested with Ansible and Argon. The aim was to provide to each participant an identical environment in which to work. Each virtual machine was configured with Windows Server 2012R2, Java JDK v1.8, the Eclipse Modeling Framework v4.8, Argon v1.0, and Ansible v2.6.

The post-experimental questionnaire contained a set of closed questions that would allow the participants to express their opinions about Ansible and Argon in terms of their perceived ease of use, perceived usefulness, and intention to use. We ensured the balance of the items in the questionnaire by putting half of the questions in their negative form and arranging all the items in random order so as to reduce the potential ceiling effect that could induce monotonous responses to question items measuring the same construct [51]. The list of items used in the questionnaire is shown in Table 4.

4.5. FAMILY DATA ANALYSIS AND META-ANALYSIS

The results of each experiment were collected using the booklets and the online questionnaire and were then analyzed.

We used descriptive statistics, box plots, and statistical tests to analyze the data collected from each experiment. As is usual, we accepted a probability of 5% of committing a Type-I Error in all the statistical tests. The data analysis was carried out by following the steps shown below:

1. We first carried out a descriptive study of the measures for the dependent variables.
2. The use of a crossover design meant that it was necessary to analyze the experiment factors such as periods, sequences and carryover. In this case, we used the linear mixed model to assess whether these factors had influenced the results.
3. In order to apply the linear mixed model, the residuals had to meet the condition of normality [54]. To ensure that the model was valid, we therefore used the Shapiro-Wilk test to confirm the normality of the residuals.
4. We then applied the linear mixed model to each dependent variable in order to assess whether the period (confounded with the experimental object), sequences (confounded with period*technique and carryover) or technique (treatment) had statistical significance.
5. Since the statistical significance is not sufficient to explain the difference caused by the treatments, we measured the effect size to assess the magnitude of those differences. The effect size of the treatments should be measured only if the period, the sequence or any blocking variables have no bearing, and there is no carryover [50]. Because our family is comparing two groups rather than evaluating the strength of association between two variables, we used Cohen's d to measure the effect size.
6. In order to strengthen the results of each individual experiment, we decided to aggregate them using a meta-analysis. We specifically performed an Aggregated Data (AD) meta-analysis based on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

Cohen's d, as the experimental conditions were similar for all the experiments. This analysis enabled us to obtain more robust results and to extract more general conclusions when considering the set of experiments in the family.

5. RESULTS

In this section, we present the empirical evidence collected from our family, in addition to discussing the results of each experiment by quantitatively analyzing the data according the hypotheses stated. We used the Statistical Package for Social Science (SPSS v24) and R v3.5.2 to obtain the experiment results. A qualitative analysis based on the feedback obtained from the post-task questionnaire is also provided.

5.1. DESCRIPTIVE STATISTICS AND EXPLORATORY DATA ANALYSIS

Table 6 presents a summary of the descriptive statistics (mean and standard deviation) for the performance-based and perception-based variables. Despite the fact that we did not further analyze *Duration*, we have included the variable in this summary in order to provide a preliminary idea of the complexity of the experimental tasks. The *Duration* is the time —in minutes— taken to perform an experimental task. The cells in bold type indicate the participants' values for each variable with the lowest *Duration* (time) and the smallest standard deviation. At a glance, the results show that the participants performed best and also had the best perceptions as regards ease of use, usefulness and intention to use when using Argon. With regard to effectiveness, it can be observed that the measures of central tendency are higher for Argon than for Ansible (the mean values range from 0.818 to 0.869 for Argon and from 0.646 to 0.649 for Ansible). The practical meaning of this is that the number of definitions of valid cloud infrastructure resources was higher when using Argon.

TABLE 6. DESCRIPTIVE STATISTICS.

Data Set	Variable	Ansible		Argon	
		Mean	Std. Dev.	Mean	Std. Dev.
UPV1	Effectiveness	0.648	0.293	0.858	0.160
	Efficiency	0.025	**0.017**	0.042	0.021
	Duration	30.00	8.211	**23.42**	8.622
	PEOU	3.80	0.866	4.49	0.500
	PU	4.08	0.667	4.53	0.494
	ITU	4.06	0.814	4.53	0.606
UPV2	Effectiveness	0.646	0.260	0.818	0.198
	Efficiency	0.022	**0.009**	0.034	0.022
	Duration	30.13	6.442	**27.53**	9.631
	PEOU	3.65	0.813	4.11	0.584
	PU	3.60	0.646	4.07	0.516
	ITU	3.49	0.674	4.18	0.581
UPV3	Effectiveness	0.649	0.359	0.869	0.134
	Efficiency	0.019	**0.011**	0.041	0.020
	Duration	34.51	7.567	**24.31**	8.704
	PEOU	3.80	0.832	4.16	0.695
	PU	3.82	0.601	4.16	0.651
	ITU	3.76	0.518	4.14	0.688

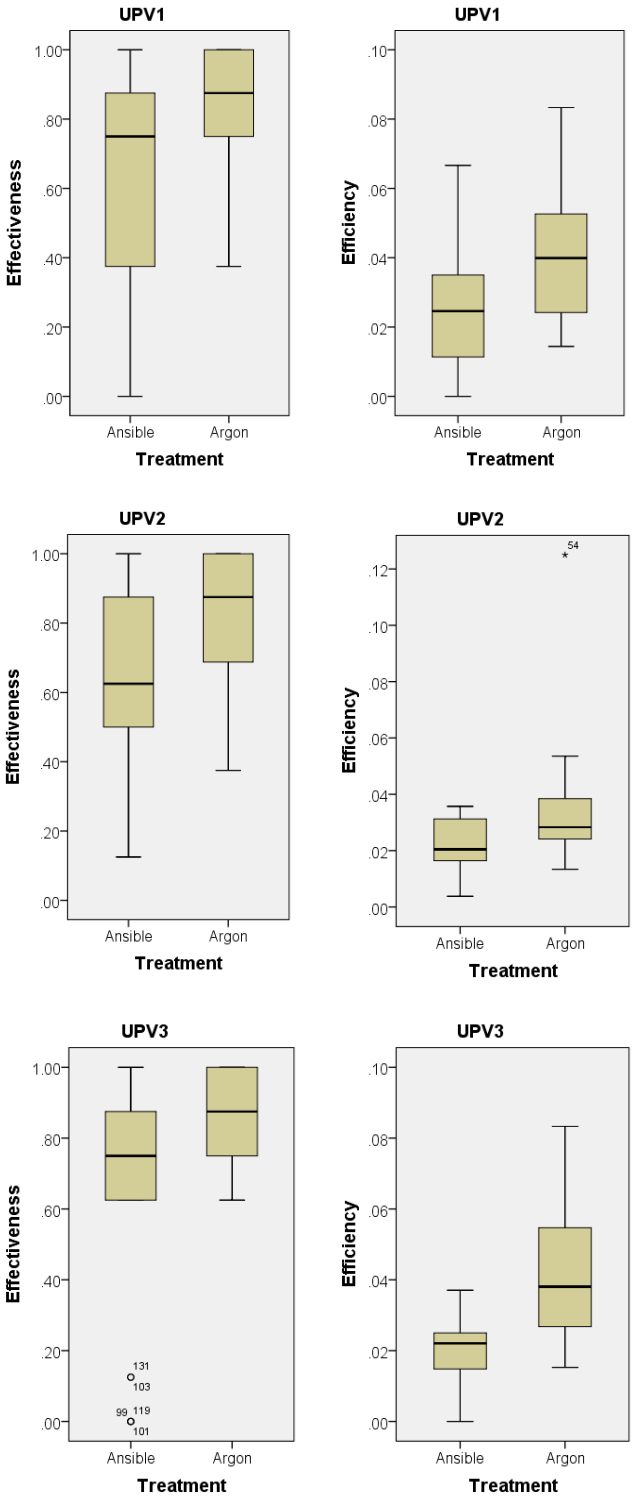


FIGURE 5. Boxplots for treatment effectiveness and treatment efficiency.

The results also indicate that the standard deviation is higher for Ansible, which indicates that the participants behaved in a more uniform manner when using Argon. With regard to the duration of the experimental task, the measures of central tendency are lower for Argon than for Ansible (the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

mean values range from 23.42 to 27.53 for Argon and from 30 to 34.51 for Ansible).

Fig. 5 shows the boxplots for the treatment effectiveness of each experiment. The results generally indicate that the participants were more effective as regards defining the cloud infrastructure when using Argon. For example, the boxplot for the treatment effectiveness of the UPV1 sample shows that Argon obtained a better result because 75% of Argon effectiveness scored over 0.75, whereas only 50% of Ansible effectiveness scored over 0.75. The boxplots for the UPV2 and UPV3 samples similarly indicate that Argon obtained better results when defining the infrastructure resources. These results could be explained by the fact that Argon employs a model-driven approach which allows participants to focus on specifying the cloud infrastructure at a higher level of abstraction rather than concentrating on the code styles of scripting languages, as is the case of Ansible. Moreover, because the participants had no previous knowledge or experience of cloud computing, another possibility is that Argon could allow participants to improve their understanding of cloud computing concepts and how infrastructure resources can be defined.

The boxplot for the treatment efficiency of the UPV1 sample (see Fig. 5) shows that Argon obtained a better result, because 75% of Argon efficiency scored over 0.024, whereas only 50% of Ansible efficiency scored over 0.024. Likewise, the boxplots for the UPV2 and UPV3 samples indicate that Argon obtained better efficiency results when specifying the infrastructure resources. Since the efficiency variable was calculated as the ratio between the effectiveness and the task duration, one reason why better efficiency was obtained is that the participants improved their effectiveness when using Argon, while another option could be that the participants specified the infrastructure resources faster when using Argon. For instance, Table 6 shows that the mean duration values range from 23.42 to 27.53 minutes for Argon and from 30 to 34.51 minutes for Ansible. This result could be explained by the fact that Argon abstracts the complexity of using scripting languages through the use of a domain-specific language (ArgonML) to model the cloud infrastructure resources.

Fig. 6 shows the boxplots for the sequence effectiveness of each experiment. In this case, the experimental groups are the sequences, that is, the order in which the subjects applied the treatments [54]. The boxplots generally show that neither the S1 sequence (Ansible-Argon) nor the S2 sequence (Argon-Ansible) made a significant difference in terms of effectiveness. For instance, the boxplots for the sequence effectiveness of the UPV1 and UPV3 samples show that 75% of both S1 and S2 scored over 0.75 for effectiveness. However, in the case of the UPV2 sample, S2 obtained a better result than S1 because 75% of S2 effectiveness scored over 0.62, whereas 75% of S1 effectiveness scored over 0.50. Note that the participants in the UPV2 sample attended the course on Model-Driven Software Development and the higher results of this sample could, therefore, be owing to the fact that the S2 sequence had the Argon-Ansible order, and the result obtained when using Argon (a model-driven approach) could consequently have motivated the participants to improve the outcome when using Ansible (a code-centric approach).

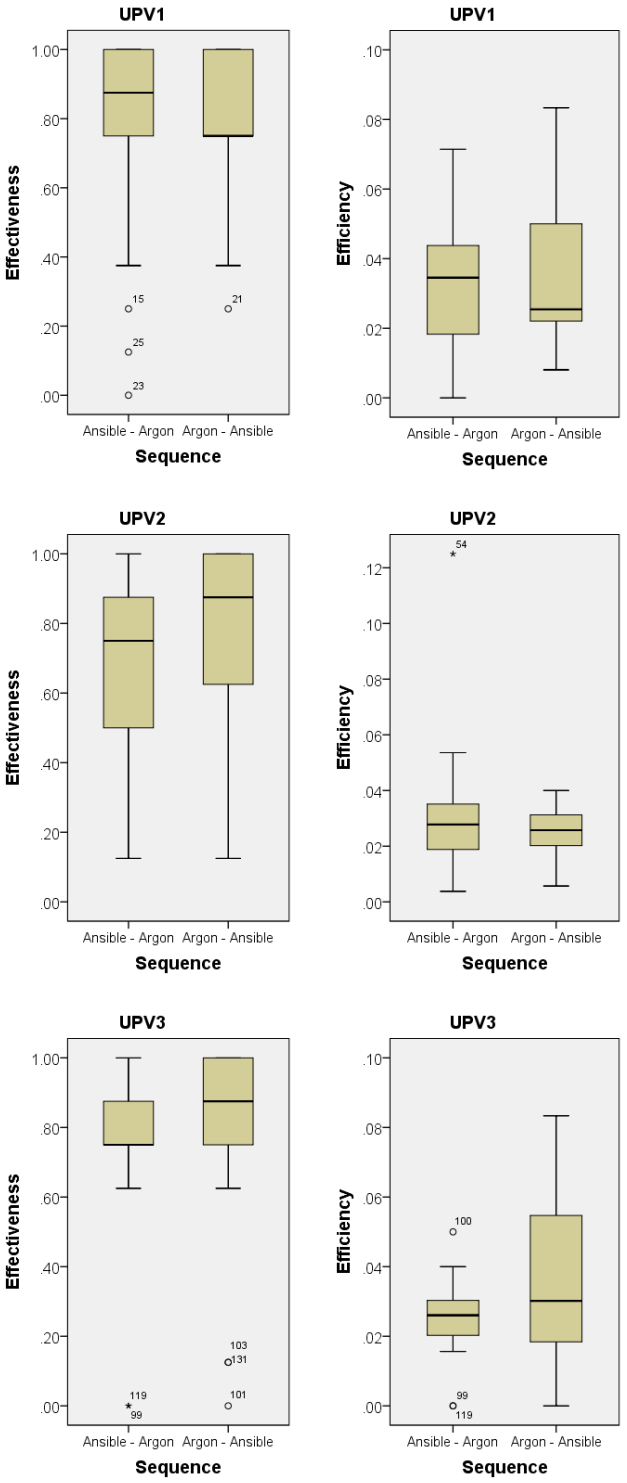


FIGURE 6. Boxplots for sequence effectiveness and sequence efficiency.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

Nevertheless, the difference in the efficiency of the UPV2 sample sequences is small, and it would be necessary to conduct more experiments with participants who have previous knowledge of model-driven development techniques to affirm that an optimal sequence exists.

Fig. 6 also shows the boxplots for the sequence efficiency of each experiment. In this case, the boxplots show a slight difference in the efficiency of each sequence. For example, the boxplot for the sequence efficiency of the UPV1 sample shows that the S1 sequence (Ansible-Argon) obtained a better result than the S2 sequence (Argon-Ansible), because 50% of S1 efficiency scored over 0.034, whereas 50% of S2 efficiency scored over 0.025. The UPV2 and UPV3 samples similarly show different results in their boxplots for sequence efficiency. Since the efficiency is the ratio between the effectiveness and the duration, the time it took to perform the experimental tasks might have affected the efficiency of each sequence. Moreover, the effectiveness of the treatments might have affected the efficiency of each sequence. As shown in the boxplots for sequence efficiency, the efficiency is not clearly comparable with regard to the sequences, and it is for this the reason that the crossover design does not focus its attention on efficiency in order to carry out a statistical analysis of the sequences [54].

Fig. 7 shows the boxplots for the period effectiveness of each experiment. In this case, a period is defined by the application of one treatment by one participant to one experimental object [54]. Note that a period uses only one experimental object at a time. The boxplots generally show that neither the MODAFIN period (O1) nor the CEC period (O2) has better effectiveness. For example, the boxplots for the period effectiveness of the UPV1 and UPV3 samples show that 75% of the effectiveness of both MODAFIN and CEC scored over 0.75. However, in the UPV2 sample, it would appear that MODAFIN attains a better result than CEC, but 50% of the effectiveness of both MODAFIN and CEC scored over 0.75. This signifies that no period has better effectiveness than another. These results also indicate that the two experimental objects have a similar complexity, and that there might be no learning effect.

Fig. 7 also presents the boxplots for the period efficiency of each experiment. In this case, the boxplots show a small difference in the efficiency of the periods. For example, the boxplot for the period efficiency of the UPV1 sample shows that the CEC period (O2) obtained a better result than the MODAFIN period (O1), because 50% of CEC efficiency scored over 0.034, whereas 50% of MODAFIN efficacy scored over 0.025. Likewise, the UPV2 and UPV3 samples show different results in their boxplots for period efficiency. Again, because efficiency is the ratio between effectiveness and task duration, the time taken to perform the experimental tasks might have influenced the efficiency of each period. Moreover, the effectiveness of the treatments might have affected the efficiency of each period. As a result, the efficiency is not clearly comparable with regard to periods and this is, therefore, the reason why the crossover design does not focus its attention on efficiency in order to carry out a statistical analysis of the periods [54].

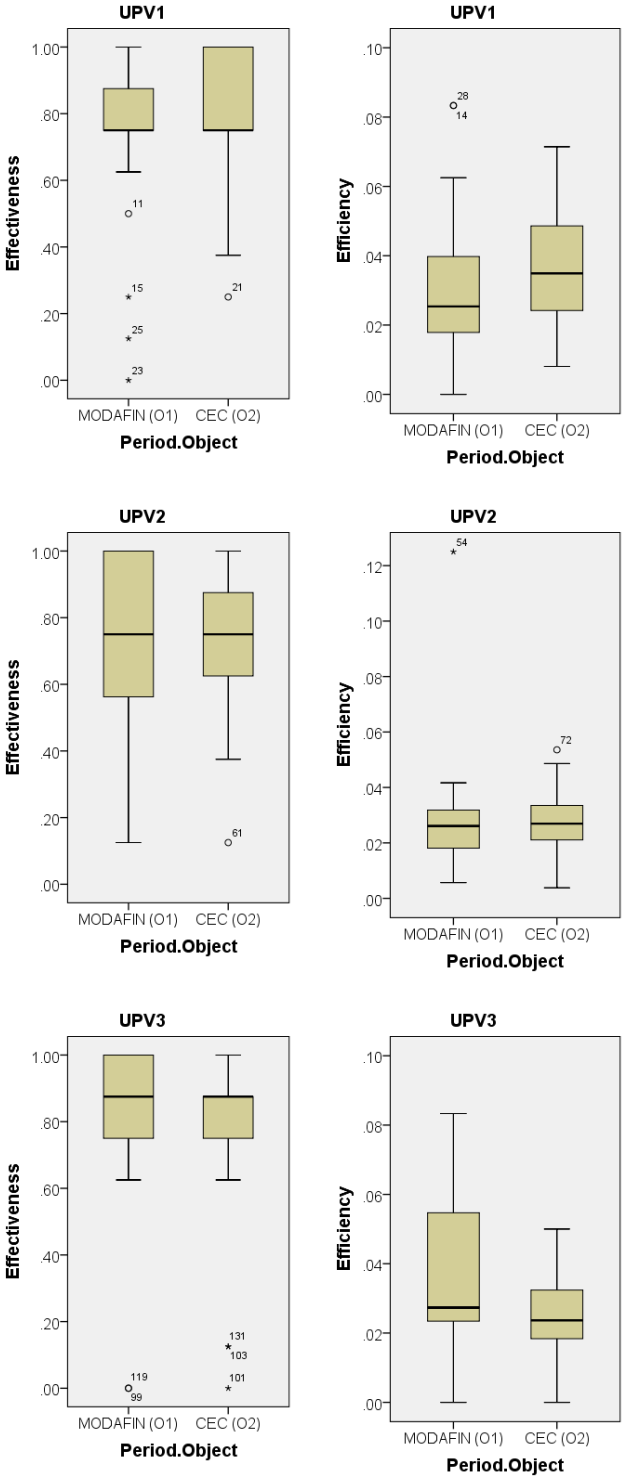


FIGURE 7. Boxplots for period effectiveness and period efficiency.

Fig. 8 shows the boxplots for the perception-based variables of all the experiments in the family. The median for each tool is shown as the horizontal line in the boxplot.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

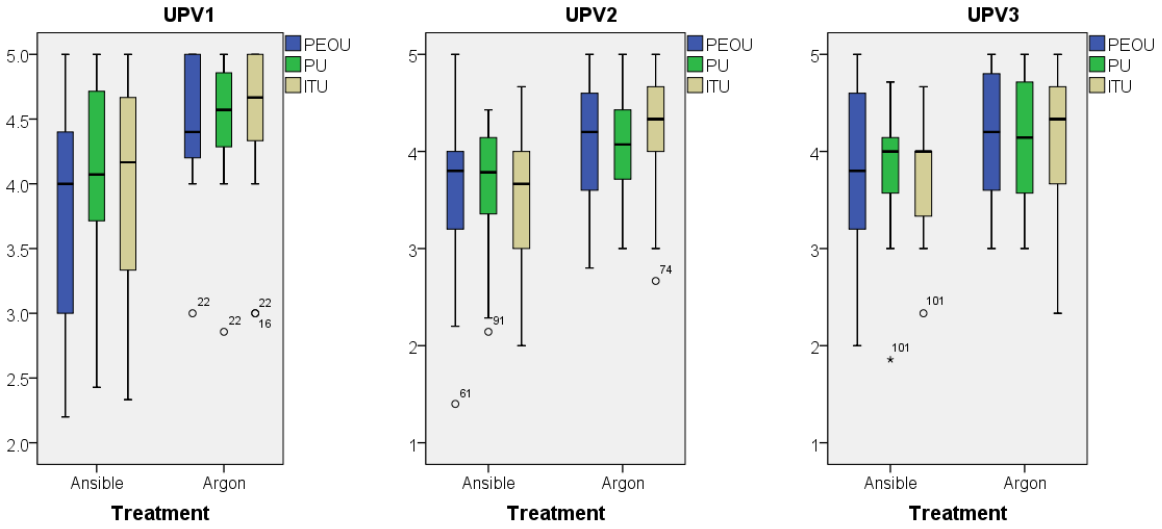


FIGURE 8. Boxplots for perception-based variables of all experiments.

The boxplots (see Fig. 8) show the participants' judgment after using each treatment (i.e., Ansible and Argon). Each tool was evaluated in terms of its perceived ease to use (PEOU) and perceived usefulness (PU) as regards supporting the participants in the definition of the cloud infrastructure. Moreover, the participants expressed their intention to use (ITU) these tools in the future. Because we used a 5-point Likert scale to measure these variables, the Likert neutral value was established at 3 points.

Fig. 8 indicates that both Argon and Ansible have a median value above the neutral value of the measurement scale, and these tools, therefore, achieved good results in terms of the participants' perceptions. With regard to the UPV1 sample, the results indicate that the participants had better perceptions of Argon (PEOU = 4.4, PU = 4.5, ITU = 4.6) than Ansible (PEOU = 4.0, PU = 4.0, ITU = 4.1) in terms of defining the cloud infrastructure. The participants similarly obtained better central tendency results as regards the perception-based variables when using Argon in the UPV2 and UPV3 samples, as shown in Fig. 8. Overall, these results suggest that the participants perceived Argon to be easier to use and more useful than Ansible when specifying the infrastructure resources, and they also expressed a greater intention to use this tool in the future.

5.2. Hypotheses testing

We used the Linear Mixed Model (LMM) to test the formulated hypotheses. According to Vega *et al.* [54], the LMM includes the following terms: technique (treatment), period (confounded with the experimental object) and sequence (confounded with carryover and period*technique interaction) as fixed factors, and subject as a random factor nested within the sequence. Note that it is necessary to verify whether the LMM residuals follow a normal distribution. In the case of absence of normality, we used the logarithm strategy or the two-step approach to transform continuous variables into normal variables [56]. Since the sample size was smaller than 50, we applied the Shapiro-Wilk test to verify whether the residuals follow a normal distribution. Table 7 shows the normality results for the performance-based and perception-based variables. The residuals express that the LMM is valid, as they meet the condition of normality, that is, the *sig.* is greater than 0.05. Additionally, the general shape of the normal distribution is analyzed in terms of skewness and kurtosis. On the one hand, skewness is a measure of the lack of symmetry in the distribution curve. A normal distribution has a skewness equal to 0, a positive value indicates that the distribution contains a larger proportion of data towards its right, whereas a negative value indicates that the distribution contains a larger proportion of data towards its left end. On the other hand, kurtosis is a measure of the degree of peakedness of the distribution curve. A normal distribution has a kurtosis equal to 0, a positive value indicates that the data is clustered around the center, whereas a negative value indicates that the data is spread out.

TABLE 7. SUMMARY OF SHAPIRO-WILK NORMALITY TESTS FOR RESIDUALS.

		Effec_r	Effic_r	PEOU_r	PU_r	ITU_r
UPV1	Sig.	**0.999**	**0.211**	**0.999**	**0.825**	**0.532**
	Skewness	0.000	0.047	0.005	-0.003	0.060
	Kurtosis	-0.445	-0.213	-0.454	-0.447	-0.344
UPV2	Sig.	**0.170**	**1.000**	**0.299**	**0.084**	**0.465**
	Skewness	-0.439	0.000	-0.461	-0.001	-0.001
	Kurtosis	0.383	-0.431	0.658	-0.431	-0.430
UPV3	Sig.	**0.982**	**0.106**	**0.259**	**0.161**	**0.580**
	Skewness	-0.022	0.508	-0.518	-0.722	-0.380
	Kurtosis	-0.499	1.523	-0.253	0.371	0.374

In addition, we generated the normal probability plot of residuals for each dependent variable. Figure 9 shows the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

normal probability plot of residuals for Effectiveness. The other dependent variables have a similar normal probability plot for their residuals.

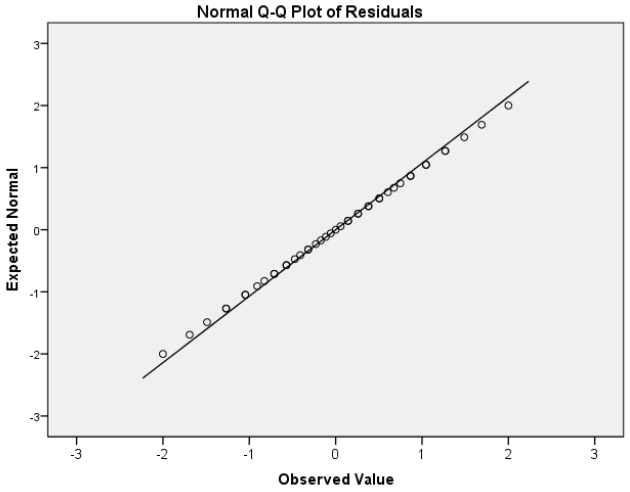


FIGURE 9. Normal probability plot of residuals (Effectiveness)

According to the test results of fixed effects shown in Table 8, the effectiveness of the Argon tool is significantly different from that of the Ansible tool for all the experiments (UPV1 = 0.002, UPV2 = 0.001, and UPV3 = 0.018, which are less than 0.05). Table 6 shows that Ansible is less effective than Argon when considering the mean effectiveness values of the experiments. In the case of the UPV1 sample, Ansible has a mean effectiveness of 0.648, while Argon has a mean effectiveness of 0.858. Similar outcomes can be found in the UPV2 and UPV3 replications. Additionally, estimated marginal means (EMM) for effectiveness were calculated, and their results are similar to Table 6, for instance: Ansible has EMM values of UPV1 = 0.648, UPV2 = 0.618, and UPV3 = 0.648, whereas Argon has EMM values of UPV1 = 0.858, UPV2 = 0.844, and UPV3 = 0.866. As a result, the null hypotheses $H1_0$ can be rejected for all the experiments, meaning that a significant statistical difference exists between the two tools in terms of the correctness of the infrastructure resources defined, which is in favor of Argon.

TABLE 8. SUMMARY OF THE TYPE III TEST OF FIXED EFFECTS FOR EFFECTIVENESS.

	UPV1	UPV2	UPV3
Source	**Sig.**	**Sig.**	**Sig.**
Period/Object	0.496	0.095	0.903
Technique	**0.002**	**0.001**	**0.018**
Sequence	0.742	0.970	0.758

With regard to efficiency, the test results obtained for fixed effects and shown in Table 9 corroborate that the efficiency of the Argon tool is significantly different from that of the Ansible tool in all the experiments (UPV1 = 0.000, UPV2 = 0.007, and UPV3 = 0.000). Table 6 shows the mean efficiency values of the experiments in which Ansible was found to be less efficient than Argon. For example, Ansible has a mean efficiency of 0.025, while Argon has a mean efficiency of 0.042 in UPV1, and similar results were attained in the UPV2 and UPV3 experiments. Likewise, estimated marginal means (EMM) for efficiency were calculated, and their results are similar to Table 6, for example: Ansible has EMM values of UPV1 = 0.025, UPV2 = 0.023, and UPV3=0.019, whereas Argon has EMM values of UPV1 = 0.042, UPV2 = 0.039, and UPV3= 0.040. It is consequently possible to reject the null hypotheses $H2_0$ in all the experiments, meaning that a significant statistical difference exists between the two provisioning tools, in favor of Argon, in terms of the number of requirements achieved for unity of time.

Note that the other two fixed factors (i.e., period and sequence) are not significant for effectiveness and efficiency. With regard to the fact that the period was not found to be significant, this means that the experimental objects (MODAFIN and CEC) did not affect the response variables (i.e., effectiveness and efficiency). Similarly, and as expected, the sequences were not found to be significant, which indicates that there is no carryover effect between either the treatments or the period*treatment interactions. Overall, the results indicate that the difference in the observed effectiveness and efficiency is owing to the provisioning tool employed.

TABLE 9. SUMMARY OF THE TYPE III TEST OF FIXED EFFECTS FOR EFFICIENCY.

	UPV1	UPV2	UPV3
Source	**Sig.**	**Sig.**	**Sig.**
Period/Object	0.051	0.149	0.110
Technique	**<0.001**	**0.007**	**<0.001**
Sequence	0.718	0.068	0.073

Before applying the LMM to the perception-based variables, we used Cronbach's alpha test to examine the reliability of each questionnaire. The test results for all the questionnaires were UPV1 = 0.962, UPV2 = 0.950, and UPV3 = 0.938, which are higher than the threshold level (0.70) [57]. Table 10 shows that the items used to measure PEOU, PU and ITU obtained a Cronbach's alpha coefficient of 0.928, 0.912 and 0.880 for the baseline experiment (UPV1), which are also higher than the threshold level. The replications show similar results, suggesting that the survey instrument can be considered reliable.

TABLE 10. SUMMARY OF CRONBACH'S ALPHA TESTS.

	PEOU		PU		ITU	
	No	**α**	**No**	**α**	**No**	**α**
UPV1	5	0.938	7	0.912	3	0.880
UPV2	5	0.896	7	0.904	3	0.780
UPV3	5	0.894	7	0.886	3	0.732

With regard to the participants' perceptions of ease of use, the test results of fixed effects shown in Table 11 confirm that the difference in perceptions of ease of use between the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

two tools is significantly different for all the experiments (UPV1 = 0.002, UPV2= 0.037 and UPV3 = 0.0034). In this scenario, Table 6 shows the mean PEOU values of those experiments in which the participants perceived that Argon was easier to use than Ansible. For example, in the first experiment (UPV1), Ansible has a mean PEOU value of 3.80, whereas Argon has a mean PEOU value of 4.49, and the results obtained for the replications (UPV2 and UPV3) were also similar. In the same way, estimated marginal means (EMM) for PEOU were calculated, and their results are similar to Table 6, for instance: Ansible has EMM values of UPV1 = 3.380, UPV2 = 3.3667, and UPV3=3.772, whereas Argon has EMM values of UPV1 = 4.491, UPV2 = 4.039, and UPV3= 4.158. The null hypotheses $H3_0$ can, therefore, be rejected for all the experiments, signifying that the participants perceived Argon to be easier to use than Ansible. The analysis of the answers to the open questions in the post-experiment questionnaire revealed that the participants found Argon was easy to use. For example, participant ID 14 said "*Modeling the infrastructure characteristics is easier than programming*" while participant ID 9 said *"Although I was not familiar with cloud computing, I was able to define an infrastructure in a short time".*

TABLE 11. SUMMARY OF THE TYPE III TEST OF FIXED EFFECTS FOR PEOU.

	UPV1	UPV2	UPV3
Source	**Sig.**	**Sig.**	**Sig.**
Period/Object	0.511	0.317	0.330
Technique	**0.002**	**0.037**	**0.034**
Sequence	0.824	0.723	0.467

With regard to Perceived Usefulness (PU), the test results for fixed effects presented in Table 12 indicate that a statistically significant difference exists between the two tools in terms of the usefulness perceived by the participants when applying the tool (UPV1 = 0.007, UPV2= 0.019 and UPV3 = 0.045). Table 6 confirms that this difference is in favor of Argon, as this tool has a mean PU value of 4.53, whereas Ansible has a mean PU value of 4.08 in the baseline experiment (UPV1), and similar results were obtained in the replications (UPV2 and UPV3). Additionally, estimated marginal means (EMM) for PU were calculated, and their results are similar to Table 6, for example: Ansible has EMM values of UPV1 = 4.084, UPV2 = 3.623, and UPV3=3.827, whereas Argon has EMM values of UPV1 = 4.526, UPV2 = 3.992, and UPV3= 4.155. As a result, the null hypotheses $H4_0$ can be rejected for all the experiments.

TABLE 12. SUMMARY OF THE TYPE III TEST OF FIXED EFFECTS FOR PU.

	UPV1	UPV2	UPV3
Source	**Sig.**	**Sig.**	**Sig.**
Period/Object	1.000	0.179	0.565
Technique	**0.007**	**0.019**	**0.045**
Sequence	0.764	0.630	0.768

With regard to Intention to Use (ITU), the test results for fixed effects depicted in Table 13 reveal that there is a difference in intention to use between the two tools for all the experiments (UPV1 = 0.032, UPV2= 0.001 and UPV3 = 0.045). Table 6 show that this difference is in favor of Argon (e.g., in the first experiment (UPV1), Argon has a mean ITU value of 4.53, whereas Ansible has a mean ITU value of 4.06). Moreover, similar results were obtained in the replications (UPV2 and UPV3). Similarly, estimated marginal means (EMM) for ITU were calculated, and their results are similar to Table 6, for instance: Ansible has EMM values of UPV1 = 4.061, UPV2 = 3.246, and UPV3 = 3.764, whereas Argon has EMM values of UPV1 = 4.530, UPV2 = 3.513, and UPV3 = 4.144. As a result, the null hypotheses $H5_0$ can be rejected for all the experiments, meaning that the participants perceived Argon to be more likely to be used than Ansible in the context of this family of experiments. This may be because, since they are novice software engineers, they perceive that the tool provides them with mechanisms with which to carry out the definition of infrastructure resources. This was confirmed by the participants in their response to the open questions in the post-experiment questionnaire: "*I would use this tool to define cloud infrastructure resources since it is intuitive, and the models allowed me to identify the architecture and components of the infrastructure quickly", "I would like to use Argon because it is easy to understand in order to model the cloud infrastructure'.*

TABLE 13. SUMMARY OF THE TYPE III TEST OF FIXED EFFECTS FOR ITU.

	UPV1	UPV2	UPV3
Source	**Sig.**	**Sig.**	**Sig.**
Period/Object	0.942	0.740	0.565
Technique	**0.032**	**0.001**	**0.045**
Sequence	0.659	0.468	0.786

Again, the fixed factors such as period and sequence are not significant for PEOU, PU, and ITU. The fact that the periods are not significant means that the experimental objects (MODAFIN and CEC) did not affect the participants' perceptions in terms of ease of use, usefulness and intention to use one of the treatments. Furthermore, the sequences are not significant, which means that there is no carryover effect between the treatments and there are no period*treatment interactions. Overall, the results indicate that the participants' observed perceptions are owing to the provisioning tool employed.

6. FAMILY DATA ANALYSIS

In this section, we present a meta-analysis that aggregates the empirical findings obtained in the individual experiments. We then answer the research questions stated for the family of experiments as a whole by considering the results obtained in each experiment and the meta-analysis.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

6.1. Meta-analysis

Since the statistical significance (*p*-value) is not sufficient to be able to claim that the difference was caused by treatments, the effect size is used to measure the magnitude of that difference. Because we used a crossover design, the effect of the treatments had to measure whether the period, the sequence or any blocking variable had any bearing, and whether there was any carryover [54]. In this context, the empirical evidence shows that the experiments have no statistical significance as regards the period, sequence and carryover.

Moreover, the effect size should be calculated only when the main factor of the experiments is the only statistically significant variable [54]. The experiments in our family consequently fulfill the requirements stated above because only the treatments have statistical significance ($p<0.05$) and it is, therefore, necessary to calculate the effect size to discover their statistical power.

In this study, we are comparing two groups rather than evaluating the strength of association between two variables. The index used to compare two groups is, therefore, Cohen's d. Table 14 presents a summary of the Cohen's d measures for the dependent variables of the experiments. The first column shows the dataset label corresponding to each experiment. The second column details the dependent variables employed to measure the effectiveness and efficiency of each treatment, along with the ease of use, usefulness and intention to use perceived by the participants. The third column presents the *p*-values in order to demonstrate compliance with the requirement to calculate the effect size. The fourth column shows the value of Cohen's d, while the fifth column provides an interpretation of the effect size. Cohen provides guidelines that make it possible to compare different effect sizes, suggesting the use of small, medium, and large effects translated into d values of 0.2, 0.5, and 0.8, respectively [58]. The sixth column shows the standard error of sampling distributions. According to Ellis [59], all the mean values of a sample are called the sample distribution, while the standard error is the standard deviation of a sampling distribution. In this scenario, the standard error is necessary to determine the confidence intervals. Finally, the last column presents the confidence intervals, which are used to combine the location and precision of the effect size level. For instance, the effectiveness of the UPV1 sample has a Cohen's d value of 0.89, and after using the corresponding confidence interval (0.14), we obtain that 0.89 ± 0.14. As a result, we obtain a range of 0.75 and 1.03, which corresponds to a medium effect and a large effect, respectively.

A meta-analysis is the most suitable option when evaluating and generalizing the results of a family of experiments. According to Ellis [59], a meta-analysis is a statistical analysis of the statistical analyses, which describes a set of procedures with which to systematically review the research examining a particular effect and combining the results of independent studies to estimate the size of the effect on the population. In this context, we used a meta-analysis to combine the Cohen's d values and analyze them to estimate the size of the effect on the target population, that is, novice software engineers. The result of the meta-analysis was a weighted mean effect size, which reflects the target population effect size more accurately than any of the individual experiments [59].

The meta-analysis was performed using the RStudio tool, version 1.1.463, and the Metaviz package, version 0.3.0. Fig. 10 summarizes the results of the meta-analysis using a forest plot. The first column shows the dataset label of each experiment, along with the effect size label of each dependent variable. The second column displays the mean and standard deviation values of the treatment with Argon. Likewise, the third column shows the mean and standard deviation of the treatments with Ansible. The fourth column presents the forest plot that summarizes the results of the meta-analysis.

TABLE 14. SUMMARY OF COHEN'S D FOR DEPENDENT VARIABLES OF EXPERIMENTS.

Data Set	Variable	p-value	Cohen's d	Cohen's d Interpretation	Standard Error	Confidence Interval
UPV1	Effectiveness	0.002	0.89	Large	0.0712	0.14
	Efficiency	<0.001	0.90	Large	0.0057	0.01
	PEOU	0.002	0.98	Large	0.2133	0.42
	PU	0.007	0.75	Medium	0.1786	0.35
	ITU	0.032	0.65	Medium	0.2163	0.42
UPV2	Effectiveness	0.001	0.74	Medium	0.0667	0.13
	Efficiency	0.007	0.72	Medium	0.0048	0.01
	PEOU	0.037	0.65	Medium	0.2044	0.40
	PU	0.019	0.80	Large	0.1687	0.33
	ITU	0.001	1.10	Large	0.1816	0.36
UPV3	Effectiveness	0.018	0.81	Large	0.0837	0.16
	Efficiency	<0.001	1.40	Very Large	0.0049	0.01
	PEOU	0.034	0.47	Small	0.2365	0.46
	PU	0.045	0.54	Medium	0.1934	0.38
	ITU	0.045	0.63	Medium	0.1879	0.37

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

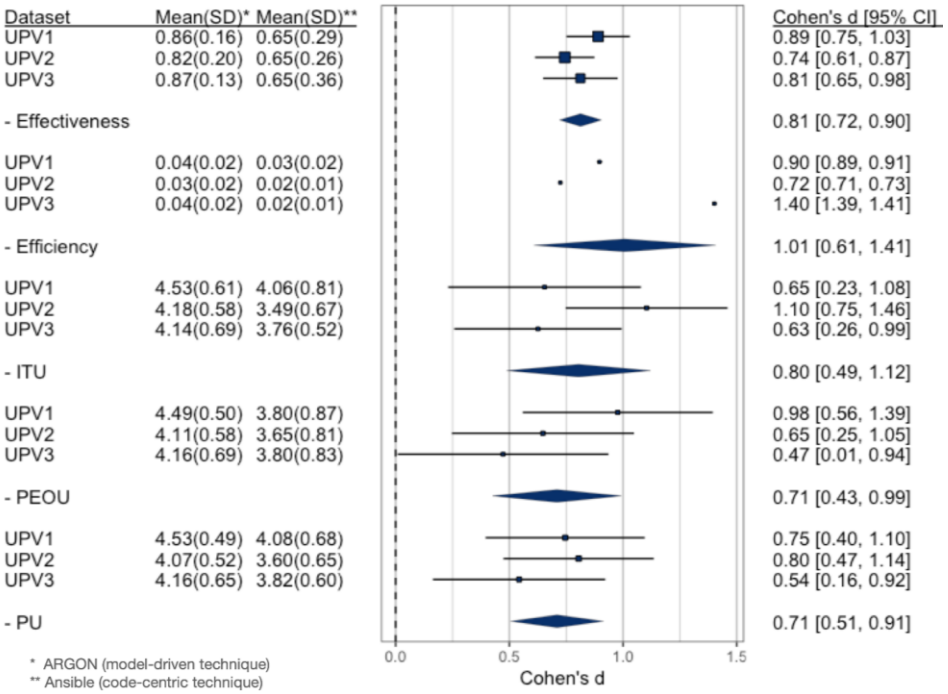


FIGURE 10. Forest plot for the meta-analysis of the family of experiments

The horizontal axis represents the scale for effect sizes being shown, while the vertical line (0.0) depicts the line of null effect. Each horizontal line in the forest plot represents the 95% confidence intervals for each dependent variable being analyzed. Each black box is proportional to the size of the study, and its position about the horizontal axis indicates the Cohen's d value. Each diamond in the forest plot represents the point estimate and confidence intervals when the results of each dependent variable are combined and averaged. In this meta-analysis, no study crosses the null effect line, signifying that all the effect sizes have statistically significant results. Finally, the fifth column indicates the Cohen's d value of each dependent variable, along with the 95% confidence interval in brackets.

6.2. Answering the research questions

Having conducted the experiments, we performed a global analysis of the results in order both to determine whether the main goal had been achieved and answer the research questions. In this study, we have gained empirical evidence on how a model-driven approach may support novice software engineers in specifying the infrastructure resources as opposed to a code-centric technique that is widely used in industrial environments. This empirical evidence is a contribution to the empirical findings on defining the cloud infrastructure resources, and it provides factual data about which tool is more suitable for the definition of infrastructure resources under certain conditions.

Table 15 presents a summary of the results of each hypothesis evaluated in the family of experiments. The results of all the experiments were, in terms of effectiveness, statistically significant ($p < 0.05$), and it is thus possible to claim that Argon is more effective than Ansible as regards defining the cloud infrastructure resources.

TABLE 15. SUMMARY OF HYPOTHESIS RESULTS IN THE FAMILY OF EXPERIMENTS.

Data Set	Experiment	No. Subjects	Type of Subjects	Hypotheses Confirmed	Hypotheses not Confirmed
UPV1	1st Experiment	22	Master's students	H1, H2, H3, H4, and H5.	-
UPV2	2nd Experiment. Strict replication of the 1st experiment (different subjects)	24	Undergraduate students	H1, H2, H3, H4, and H5.	-
UPV3	3rd Experiment. Strict replication of the 1st experiment (different subjects)	21	Undergraduate students	H1, H2, H3, H4, and H5.	-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

In this context, and considering the descriptive statistics shown in Section 5.1, the boxplots for treatment effectiveness suggest that Argon obtained better results because it obtained 75% for effectiveness, while Ansible obtained only 50%. With regard to the boxplots for the sequence effectiveness of the UPV1 and UPV3 samples, the S1 sequence (Ansible-Argon) and the S2 sequence (Argon-Ansible) scored 75% for effectiveness above the 25th percentile. Finally, the boxplots for the period effectiveness of the UPV1 and UPV3 samples show that MODAFIN (O1) and CEC (O2) have 75% of effectiveness above the 25th percentile and, in the case of the UPV2 sample, MODAFIN (O1) and CEC (O2) have 50% of effectiveness above the 50th percentile.

To conclude, Argon proved to be a more effective means to specify the cloud infrastructure resources. Moreover, there is no evidence that one period is more effective than another, and there is no evidence that one sequence is more effective than another. In the case of the UPV2 sample, the possible reason why S2 was more effective than S1 was the knowledge that the participants had obtained on model-driven techniques from the Model-Driven Software Development course.

With regard to efficiency, the results show that Argon is more efficient than Ansible as regards defining the cloud infrastructure. In order to perform the statistical analysis, it was necessary to consider the time needed to carry out the experimental tasks. In this scenario, Argon proved to be more efficient than Ansible, but it was not possible to clearly compare the sequences and periods of the experiments. This is why the crossover design does not focuses its attention on efficiency in order to carry out statistical analyses and draw conclusions [54].

With regard to the Perceived Ease of Use (PEOU), Perceived Usefulness (PU), and Intention to Use (ITU) variables, the results are statistically significant ($p < 0.05$) and it is possible to claim that Argon is easier to use and more useful as regards specifying the cloud infrastructure. Moreover, the participants indicated their intention to use Argon in the future. In general terms, the median values of the PEOU, PU, and ITU variables of both Argon and Ansible are above the Likert neutral value established at 3 points. The participants consequently perceived that both tools are easy to use and useful, and they also expressed their likely intention to use these tools. However, because the average time expended by the participants when using Ansible was 32 minutes, whereas the average time spent using Argon was 21 minutes, this might be the factor that decides which treatment was perceived as better by the participants.

In the case of the research questions motivating our family of experiments, we shall answer them by using the empirical findings to support the claims.

RQ1. Which IaC tool is more effective when defining the cloud infrastructure?

We found empirical evidence to claim that Argon is more effective than Ansible as regards specifying the infrastructure resources (i.e., the correctness of the definition of the cloud infrastructure resources was superior with Argon), and there is no proof to allow us to state that periods or sequences affected the effectiveness of the treatments. All the experiments are, therefore, statistically significant in terms of effectiveness, and we can, therefore reject the null hypotheses $H1_0$. The meta-analysis also confirmed that the Cohen's d coefficient for the effectiveness variable has a practical significance, with a medium effect size for the UPV2 dataset and a large effect size for the UPV1 and UPV3 datasets. In general, our results show that although Ansible was designed to make IaC environments accessible to anyone with a basic knowledge of modern coding techniques and structures, the model-driven approach followed by Argon helped the participants to specify more correct cloud infrastructure definitions.

RQ2. Which IaC tool is more efficient when defining the cloud infrastructure?

We found empirical evidence to support the fact that Argon is more efficient than Ansible as regards specifying the infrastructure resources. All the experiments are statistically significant in terms of efficiency, and this allows us to reject the null hypotheses $H2_0$. The meta-analysis also confirmed that the Cohen's d coefficient for the efficiency variable has a practical significance, with a medium, large, and very large effect size for the UPV2, UPV1 and UPV3 datasets, respectively. Since the efficiency is the ratio between effectiveness and time, and Argon was found to be more effective than Argon as regards specifying the infrastructure resources, this might have affected the efficiency of Argon.

The results of this study are encouraging and suggest that applying a model-driven approach during the definition of small/ medium cloud infrastructures is useful. Nevertheless, more evidence on the effect of using large specifications is required to determine long-term cost implications. In addition, more experimentation is needed to study the effect of model-driven vs. code-based tools on the long-term maintainability of cloud infrastructure definitions in a DevOps process.

RQ3. Which IaC tool is perceived to be easier to use?

We found empirical evidence to affirm that the participants perceived Argon to be easier to use than Ansible when defining the cloud infrastructure. This result may be owing to the fact that the participants perceived that learning and using a model-driven approach like Argon was effortless. All the experiments are, therefore, statistically significant in terms of PEOU, and we can reject the null hypotheses $H3_0$. The meta-analysis also confirmed that the Cohen's d coefficient for the PEOU variable has a practical significance, with a medium effect size for the UPV2 dataset and a large effect size for the UPV1 and UPV3 datasets. Our results generally suggest that the use of models to define the cloud infrastructure improved the participants' perceived ease of use of the tool. In fact, the participants were able to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

correctly define the infrastructure resources with Argon without any extensive training on modeling.

RQ4. Which IaC tool is perceived to be more useful?

We found empirical evidence to claim that the participants perceived Argon to be more useful than Ansible when specifying the cloud infrastructure. This result might be owing to the fact that the participants perceived that Argon was effective as regards achieving its objective, which was to define the infrastructure resources. All the experiments are, therefore, statistically significant in terms of PU, and we can reject the null hypotheses $H4_0$. The meta-analysis also confirmed that the Cohen's d coefficient for the PU variable has a practical significance, with a medium effect size for the UPV1 and UPV3 datasets and a large effect size for the UPV2 dataset. Our results generally suggest that Argon mitigates the complexity of using and managing the scripting language, and the participants consequently perceived that Argon is useful to define the cloud infrastructure resources.

RQ5. Which IaC tool is most intended to be used?

We found empirical evidence to claim that the participants intend to use Argon in the future. This result may be owing to the fact that the participants accomplished the experimental tasks in a short time, and they felt comfortable when using Argon to specify the cloud infrastructure. As a result, all the experiments are statistically significant in terms of ITU, signifying that we can reject the null hypotheses $H5_0$. Moreover, the meta-analysis confirmed that the Cohen's d coefficient for the ITU variable has a practical significance, with a medium effect size for the UPV1 and UPV3 datasets and a large effect size for the UPV2 dataset.

In summary, the results support our hypothesis that Argon would better define the cloud infrastructure in a specific context, in which the participants specify the infrastructure resources to be deployed in a particular cloud provider. According to the previously discussed results, we can conclude that Argon can be considered as a promising approach with which to specify cloud infrastructures and generate scripts for different provisioning tools using model-driven techniques.

Feedback on the difficulties experienced by the participants when using the provisioning tools or suggestions on how to improve the tools were also obtained. This feedback was provided in their responses to the open questions in the questionnaire. In the case of Ansible, the participants reported problems as regards understanding error messages because what was stated in an error message did not always reflect what was wrong. It would appear that, despite its widespread use, Ansible has unknown and unidentified errors. Note that we delivered an already configured Ansible tool in order to avoid installation or configuration problems, but the participants found that some messages were ambiguous and misleading. This may be owing to the participants' low level of knowledge as regards using the debug logging feature of Ansible.

With regard to Argon, some participants reported difficulties when modeling the infrastructure resources. The main issues are related to difficulties in distinguishing the meanings of some modeling elements (i.e., health check), establishing relationships among the elements and defining the element properties. One possible reason for this may be the fact that it was the first time that the participants had used cloud provisioning tools to define resources for a cloud infrastructure. As further work, we plan to replicate the experiment with participants with experience in cloud provisioning. We also plan to perform an empirical study in order to evaluate the graphical notation of the language according to the Physics of Notation [60].

The results are, on the whole, promising, as we obtained empirical evidence regarding the effectiveness of two tools with which to support Infrastructure as Code concerning the definition of cloud infrastructure resources. Conducting a family of experiments rather than a single experiment allowed us to strengthen the results obtained, as the same hypotheses were tested in different settings.

6.3. Limitation of the study

With respect to the proposed evaluation method, two limitations should be acknowledged and addressed in relation to the family of experiments. The first limitation concerns the investigation of other factors that may affect the comparison of IaC tools (i.e., Argon and Ansible) in the context of the definition of cloud infrastructure resources. For instance, a new factor would be the industrial experience to define cloud infrastructure resources. In order to use the industrial experience as a factor, it is necessary to define one group of students and another of professionals with industrial experience to evaluate if the experience affects the finding of this study.

The second limitation concerns the measurement scales for measuring the perception-based variables. We used the TAM method propose by Moody [51]. However, TAM focuses specifically on Information Systems design methods. We transferred the TAM items to the context of IaC tools and, in particular, in the context of defining the cloud infrastructure resources. The weakness involved in this is that TAM has been developed specifically for the context of technology acceptance, and the items may not be totally transferable to a different domain. Even though our family of experiments presented good results, an in-depth analysis of this issue should be carried out in further experimentation.

7. THREATS TO VALIDITY

In this section, we follow the recommendations of Wohlin *et al.* [45] to discuss the issues that might have threatened the validity of our family of experiments.

7.1. Internal Validity

Threats to internal validity are influences that can affect the independent variable with respect to causality [45]. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

includes learning effect, fatigue effects, participant experience, information exchange among participants, understandability of the documents, and instrumentation validity. The threats to the internal validity have been mitigated by the design of the experiment. In particular, we take into account the factors that intervene in the crossover design, such as period, sequence, carryover and subjects.

The period is confounded with the experimental object, and it consists of requirements that the participants have to use to specify the cloud infrastructure resources. Since we used two experimental objects, the results show that there is no empirical evidence that one of the periods improves the experimental results more than another.

The sequence specifies the order in which the treatments will be applied. In this case, there are two sequences: S1 (Argon-Ansible) and S2 (Ansible-Argon). In general, there is not sufficient empirical evidence to be able to claim that one of the sequences is better than another, although, in the UPV2 sample, S2 would appear to be better than S1. The result obtained when using Argon could have motivated the participants to improve the outcome when using Ansible. However, further replications are required to confirm or contradict these results.

The carryover occurs when a treatment is administered before the effect of another previously administered treatment has completely receded [54]. Additionally, the interaction between treatment and period is intrinsically confounded with carryover, and with the sequence effect, and it is consequently impossible to distinguish which of the three is occurring [54]. However, period and sequence have no statistical significance in any of the experiments, and there is consequently no carryover. This signifies that the participants were not affected by the carryover, and that the learning effect was mitigated by ensuring that each participant worked with the two tools on two different experimental objects, using a within-subjects experimental design.

With regard to the participants' experience, the random heterogeneity of subjects is always present when experimenting with students, and we are also conscious that they had no previous knowledge of either cloud computing or the cloud provisioning tools being compared. Furthermore, if the knowledge of the students involved in the experiment could be assumed to be comparable to that of junior industry professionals, the working pressure and the overall environment in industry is different. The experiment should, therefore, be replicated with participants with experience in cloud provisioning tools. Nevertheless, the experience attained in this first study will allow us to refine the material and tasks with the objective of performing a replication in an industrial setting.

With respect to the survey questionnaire, we had special careful in the procedural cautionary procedures such as the anonymity of respondents and the confidentiality of the questionnaire in order to reduce evaluation apprehension. On the one hand, participants were not evaluated to carry out the experimental task and hence they used an identification code rather than their names. On the other hand, due to questionnaires do not have the participants name, as well as participants, were not graded, they did not feel stressed and hence we mitigate the evaluation apprehension.

We mitigated the fatigue effects by carrying out the experiment in a time slot of 2 hours per session. Moreover, we were able to prevent information exchange by using different experimental objects in the two runs and monitoring the participants during the experiments. Finally, we assessed the understandability of the materials by conducting a pilot study in order to discover mistakes and correct them.

7.2. Conclusion Validity

Threats to conclusion validity concern data collection, the reliability of the measurement and the validity of the statistical tests. In order to achieve reliability as regards data collection, we provided to the participants a virtual machine configured with all the tools required to execute the experimental task. Note that the virtual machines for all the participants were set precisely with the same tools and workspace. We additionally used the AB/BA crossover design to mitigate the issues regarding a small sample size and to increase the sensitivity of each experiment. With regard to statistical power, we followed the guideline proposed by Vega *et al.* [54], which states that the effect size of the treatments should be measured only if the period, the sequence or any blocking variables have no bearing and there is no carryover. All the experiments in this study fulfill these requirements and we, therefore, obtained a meta-analysis with both statistically significant and practically significant results, as evidenced by the effect size scores. Moreover, in order to decrease the data collection threat, we applied the same data-extraction procedures to each experiment and ensured that each dependent variable was calculated consistently.

7.3. Construct Validity

Construct validity concerns generalizing the result of the experiment, taking into account the experiment design and its ability to reflect the constructs to be studied, along with issues related to the behavior of the participants and the experimenters [45].

The measures used to obtain the qualitative and quantitative variables might influence the construct validity. We mitigated this threat by using measures that are commonly applied in other empirical software engineering studies. On the one hand, since we used performance-based variables such as effectiveness and efficiency to evaluate the definition of the cloud infrastructure resources, we used a set of eight requirements to evaluate the two treatments. In this scenario, a requirement is correctly defined if the infrastructure described satisfies everything requested in that requirement, independently of the technique (IaC tool) used.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

On the other hand, we used a questionnaire to measure the perception-based variables in terms of perceived ease of use, perceived usefulness and intention to use of each treatment. These constructs are widely used to measure participants' perceptions and are based on the Technology Acceptance Model [61]. The questionnaire was, therefore, defined using standard forms and scales. The reliability of the questionnaire was assessed using Cronbach's alpha test. Table 10 shows that the Cronbach's α coefficients for all the experiments and variables in the family were higher than the threshold level (0.70). Although TAM was suitable in the context of our family of experiments because we evaluated the user's perceptions about the use of IaC tools (i.e., Argon and Ansible), there exist other models that would be used in different contexts. On the one hand, the Matching Person and Technology (MPT) model is useful to assess and recommend the successful use of assistive technologies for people with disabilities and hence MPT would be used to evaluate how IaC tools support people with disabilities in tasks of definition and provisions of cloud resources. On the other hand, the Hedonic-Motivation System Adoption Model (HMSAM) is suitable to explain the adoption of purely intrinsic or hedonic systems, such as games, music, learning for pleasure. Thus, HMSAM would be useful to evaluate IaC tools in the contexts of serious games.

Finally, in order to avoid evaluation apprehension, the participants were not graded on the results they obtained. Nevertheless, they gained an extra point for participating in the experiment. Furthermore, the participants were not aware of the experimental hypotheses so as to prevent bias in the treatments performed.

7.4. External Validity

Threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice [45]. The first concern was to select groups of participants who are representative of the target population, that is, software developers and operation staff. The participants chosen were students who had no previous knowledge of cloud computing or IaC tools. However, students are the next generation of software professionals and are, therefore, relatively close to the target population [47]. Additionally, in the baseline experiment, 14 Master's students reported that they had professional experience in software development, and hence the term "student" does not preclude the possibility of having industrial experience. In the case of the family of experiments, the goal is to analyze the definition of cloud infrastructure resources from the viewpoint of novice software engineers in the context of Computer Science Master's and Bachelor's degree students. Consequently, we believe that it is possible to generalize the experimental findings between Computer Science Master's and Bachelor's degree students or related fields.

In addition, the tasks to be performed did not require high levels of industrial experience and we are, therefore, of the opinion that this experiment could be considered appropriate, as suggested in previous studies [48]. Working with students also has some advantages, such as the fact that the students' prior knowledge is fairly homogeneous. Nevertheless, we dealt with ethical issues properly. There was no information about the students in the raw data that could allow a particular student to be identified, and it was not possible to link their names to their responses when the experimenters were analyzing the data.

The size and complexity of experimental objects is a threat that might affect external validity. We mitigated the selection of the systems to be provisioned by considering two systems with a similar size and complexity. The participants' lack of familiarity with the problem domain of a system might affect the understandability of the experimental objects, thus biasing the results by adding an extra cognitive effort. It was for this reason that we decided to use well-known domains (MODAFIN and CEC) as part of the experiment.

The size and complexity of the tasks may also affect the external validity. In this study, we used an experimental task with a moderate complexity because the experiment required the participants to complete the assigned task in a limited time slot (2-hours sessions). Nevertheless, we consider that the task performed by the participants (specify infrastructure resources using a load balancer that distributes workloads among virtual machines) has a moderate complexity and can consequently be considered as a representative task in a practical setting. Furthermore, we selected the Ansible tool as a control treatment because it is a widely used IaC tool in industry [41].

8. CONCLUSIONS

The theoretical contribution of this work is a TAM-based model for evaluating IaC tools. The theoretical model explains the relevant dimensions of quality for IaC tools, along with a practical instrument with which to measure these quality dimensions. Basically, it allows us to predict the possible acceptance of a IaC tool based on the effort of applying the tool, the quality of the cloud infrastructure definition produced, and the user perceptions with regard to the quality of the tool. This theoretical model can be reused by researchers or practitioners to evaluate other IaC tools.

The practical contributions are the application of this model for evaluating two specific IaC tools (Ansible and Argon) and the empirical evidence regarding which tool is more effective in supporting human users when defining the cloud infrastructure. Specifically, we conducted a family of experiments to gain empirical evidence regarding how a recently proposed model-driven infrastructure provisioning approach, supported by the Argon tool, may help novice software engineers when specifying cloud infrastructure resources in comparison to a widely used scripting tool (Ansible).

This empirical evidence is a contribution to the body of knowledge on model-driven engineering and the IaC

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

approach, since it provides factual data concerning which approach (model-driven or code-centric) is more suitable under certain conditions as regards supporting the IaC in terms of defining the cloud infrastructure resources. In particular, we found evidence supporting the claim that a model-driven approach (Argon) is more effective than a code-centric technique (Ansible) when specifying cloud infrastructure resources. Moreover, the time required to model the infrastructure resources and generate the scripts with Argon was less than that required to write the script with Ansible. Note that fixed factors such as the period, sequence and carryover were not statistically significant and, as expected, these factors consequently had no influence on the experimental results. As a result, our proposal with which to model the cloud infrastructure was more effective and efficient as regards defining cloud infrastructures.

We also found evidence to support the claim that the participants perceived Argon to be easier to use and more useful than Ansible when specifying the infrastructure resources. The participants also expressed their intention to use Argon in the future. Argon can consequently be considered as a promising infrastructure modeling tool for cloud provisioning, which may improve the time and effort required to define infrastructure resources. However, note that the effort required to create infrastructure models may decrease after the intensive adoption of Argon by an organization. Indeed, we plan to carry out an empirical study to assess the effort involved when modeling the infrastructure resources with Argon.

Our findings have several practical implications. We believe that Argon is industry-relevant. Practitioners consider IaC to be a fundamental pillar on which to implement DevOps practices, which helps them to rapidly deliver software and services to end-users. Argon proved to be a useful tool when modeling cloud infrastructure resources and generating scripts for different DevOps tools (e.g., Ansible). This may reduce the time and effort required to write scripts for different tools, along with reducing the appearance of defects in IaC scripts. The results obtained are, therefore, of interest to all those companies that plan to adopt IaC tools to define infrastructure resources. The expertise that a company should have in order to adopt Argon includes a basic knowledge of modeling and expertise in cloud provisioning.

From a research perspective, we are aware that this family of experiments has provided preliminary results on the effectiveness of Argon as an infrastructure modeling tool. Although the findings are promising, these results need to be interpreted with caution, since they are valid only within the context established in this family of experiments. In particular, the empirical evidence obtained from this study should be considered valid in the context of undergraduate/Master's degree students (considered as novice software engineers) who are defining cloud provisioning resources of relatively simple systems from well-known domains. It is, therefore, necessary to verify whether the same results hold if more complex experimental objects are used or practitioners experienced in cloud infrastructure provisioning are involved in the experimentation. Nonetheless, this study is valuable as a first family of experiments with which to evaluate the effectiveness of the tools used to support the IaC approach. The experimental design and materials from this family of experiments will also be useful for other researchers interested in comparing the effectiveness of IaC tools.

Other implications are related to education in the fields of model-driven engineering and cloud computing. Educators confront the need to choose which of several tools is the most suitable when teaching the foundations of model-driven techniques or cloud provisioning. On the one hand, Argon leverages the model-driven techniques and consequently abstracts the cloud capabilities into an infrastructure model and provides infrastructure automation through model-to-model and model-to-text transformations. On the other hand, Argon proposes a domain-specific language with which to model the cloud infrastructure resources, which is an alternative for a better understanding of infrastructure resources. In particular, the results could guide educators to focus on the specific aspects of a given IaC tool so as to better support students in overcoming difficulties related to specifying cloud infrastructure resources.

As future work, we plan to replicate this experiment using practitioners experienced in cloud infrastructure provisioning and more complex systems. This will allow us to gather further empirical evidence about the effectiveness of Argon in industrial settings. We also plan to carry out other experiments focusing on the maintenance of existing infrastructure resources using Argon. This will allow us to provide empirical evidence on the usefulness of Argon as regards supporting the evolution of cloud infrastructures. Finally, we plan to compare Argon with other code-centric IaC tools such as Terraform and AWS CloudFormation.

APPENDIX A. Questionnaires of the IaC tools.

The questionnaire items to assess the IaC tools were taken and adapted from the TAM method propose by Moody [51], and hence are organized based on its perception-based variables such as Perceived Ease of Use (PEOU), Perceived Usefulness (PU), and Intention to Use (ITU). The questionnaire items were formulated by using a 5-point Likert scale and adopting the opposing-statement question format. Various items within the same construct group were randomized to prevent systemic response bias.

A.1. Questionnaire items for the Argon tool

PEOU1: I found the procedure for using the Argon tool complex and difficult to follow.
PEOU2: Overall, I found the Argon tool difficult to use.
PEOU3: I found the Argon tool easy to learn.
PEOU4: I found it difficult to define the cloud infrastructure with the Argon tool.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966597, IEEE Access

PEOU5: I found the use of the Argon tool clear and easy to understand.
PU1: I believe that the Argon tool would reduce the effort required to define the cloud infrastructure.
PU2: Overall, I found the Argon tool useful.
PU3: The cloud infrastructure defined using the Argon tool would be more difficult to understand.
PU4: Overall, I think the Argon tool does not provide an effective solution to define the cloud infrastructure.
PU5: Overall, I think the Argon tool makes an improvement to the cloud infrastructure definition process.
PU6: The Argon tool would make it easier for practitioners to define the cloud infrastructure.
PU7: Using the Argon tool would make it easier to communicate the cloud infrastructure definition to other practitioners.
ITU1: I would recommend the Argon tool to define the cloud infrastructure.
ITU2: If I am working at a company in the future, I would like to use the Argon tool to define the cloud infrastructure.
ITU3: It would be easy for me to become skillful in using the Argon tool to define the cloud infrastructure.

A.2. Questionnaire items for the Ansible tool

PEOU1: I found the procedure for using the Ansible tool complex and difficult to follow.
PEOU2: Overall, I found the Ansible tool difficult to use.
PEOU3: I found the Ansible tool easy to learn.
PEOU4: I found it difficult to define the cloud infrastructure with the Ansible tool.
PEOU5: I found the use of the Ansible tool clear and easy to understand.
PU1: I believe that the Ansible tool would reduce the effort required to define the cloud infrastructure.
PU2: Overall, I found the Ansible tool useful.
PU3: The cloud infrastructure defined using the Ansible tool would be more difficult to understand.
PU4: Overall, I think the Ansible tool does not provide an effective solution to define the cloud infrastructure.
PU5: Overall, I think the Ansible tool makes an improvement to the cloud infrastructure definition process.
PU6: The Ansible tool would make it easier for practitioners to define the cloud infrastructure.
PU7: Using the Ansible tool would make it easier to communicate the cloud infrastructure definition to other practitioners.
ITU1: I would recommend the Ansible tool to define the cloud infrastructure.
ITU2: If I am working at a company in the future, I would like to use the Ansible tool to define the cloud infrastructure.
ITU3: It would be easy for me to become skillful in using the Ansible tool to define the cloud infrastructure.

REFERENCES

[1] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, First Edition. Addison-Wesley Professional, 2010.
[2] "DevOps - Gartner IT Glossary." [Online]. Available: https://www.gartner.com/it-glossary/devops/. [Accessed: 04-Jul-2019].
[3] A. Mann, A. Brown, M. Stahnke, and N. Kersten, "2018 State of DevOps Report," 2018.
[4] K. Morris, *Infrastructure As Code: Managing Servers in the Cloud*, First Edit. O'Reilly Media, 2016.
[5] Y. Brikman, *Terraform: Up and Running*, First Edit. O'Reilly Media, 2017.
[6] R. Buyya, J. Broberg, and A. Gościński, *Cloud computing : principles and paradigms*. Wiley, 2011.
[7] J. Sandobalín, E. Insfrán, and S. Abrahão, "An Infrastructure Modelling Tool for Cloud Provisioning," in *Proceedings - IEEE 14th International Conference on Services Computing, SCC*, 2017, pp. 354–361.
[8] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Inf. Softw. Technol.*, pp. 65–77, 2019.
[9] J. C. Carver, N. Juristo, M. Teresa, and B. Sira, "Replications of software engineering experiments," pp. 267–276, 2014.
[10] D. T. Campbell and J. C. Stanley, *Experimental and Quasi-Experimental Design for Research*. 1963.
[11] M. DeHaan, "Ansible," *Sponsored by Red Hat*, 2012. [Online]. Available: https://www.ansible.com/. [Accessed: 25-Jan-2019].
[12] A. Santos, O. S. Gomez, and N. Juristo, "Analyzing Families of Experiments in SE: a Systematic Mapping Study," *IEEE Trans. Softw. Eng.*, no. July, pp. 1–18, 2018.
[13] O. S. Gómez, N. Juristo, and S. Vegas, "Understanding replication of experiments in software engineering: A classification," *Inf. Softw. Technol.*, 2014.
[14] V. R. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments," *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 456–473, 1999.
[15] Amazon Web Services, "CloudFormation," 2011. [Online]. Available: https://aws.amazon.com/cloudformation/. [Accessed: 29-Aug-2019].
[16] Amazon Web Services, "OpsWorks," 2011. [Online]. Available: https://aws.amazon.com/opsworks/. [Accessed: 29-Aug-2019].
[17] HashiCorp, "Terraform," 2017. [Online]. Available: https://www.terraform.io/. [Accessed: 29-Aug-2019].
[18] Chef, "Chef," 2009. [Online]. Available: https://www.chef.io/. [Accessed: 25-Sep-2019].
[19] Puppet Labs, "Puppet," 2005. [Online]. Available: https://puppet.com/. [Accessed: 25-Sep-2019].
[20] N. Ferry and A. Rossini, "CloudMF: Model-Driven Management of Multi-Cloud Applications," *ACM Trans. Internet Technol.*, vol. 18, no. 2, pp. 16–24, 2018.
[21] V. Casola *et al.*, "MUSA deployer: Deployment of multi-cloud applications," in *Proceedings - IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, 2017, pp. 107–112.

[22] W. Chen *et al.*, "MORE: A model-driven operation service for cloud-based IT systems," in *2016 IEEE International Conference on Services Computing (SCC)*, 2016, pp. 633–640.

[23] A. Bernal, M. E. Cambronero, V. Valero, A. Nunez, and P. C. Canizares, "A Framework for Modeling Cloud Infrastructures and User Interactions," *IEEE Access*, vol. 7, pp. 43269–43285, 2019.

[24] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, and G. Reggio, "Relevance, benefits, and problems of software modelling and model driven techniques - A survey in the Italian industry," *J. Syst. Softw.*, vol. 86, no. 8, pp. 2110–2126, 2013.

[25] G. Mussbacher *et al.*, "The Relevance of Model-Driven Engineering Thirty Years from Now," pp. 183–200, 2014.

[26] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE Softw.*, vol. 31, no. 3, pp. 79–85, 2014.

[27] L. T. W. Agner, I. W. Soares, P. C. Stadzisz, and J. M. Simão, "A Brazilian survey on UML and model-driven practices for embedded software development," *J. Syst. Softw.*, vol. 86, no. 4, pp. 997–1005, 2013.

[28] F. Ricca, M. Torchiano, M. Leotta, A. Tiso, G. Guerrini, and G. Reggio, "On the impact of state-based model-driven development on maintainability: a family of experiments using UniMod," *Empir. Softw. Eng.*, vol. 23, no. 3, pp. 1743–1790, 2018.

[29] A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. 2003.

[30] Y. Martínez, C. Cachero, and S. Meliá, "MDD vs. traditional software development: A practitioner's subjective perspective," *Inf. Softw. Technol.*, vol. 55, no. 2, pp. 189–200, 2013.

[31] Y. Martínez, C. Cachero, and S. Meliá, "Empirical study on the maintainability of Web applications: Model-driven Engineering vs Code-centric," *Empir. Softw. Eng.*, vol. 19, no. 6, pp. 1887–1920, 2014.

[32] O. Parra, S. España, J. I. Panach, and O. Pastor, "An empirical comparative evaluation of gestUI to include gesture-based interaction in user interfaces," *Sci. Comput. Program.*, vol. 172, pp. 232–263, 2019.

[33] K. Ikeshita, F. Ishikawa, and S. Honiden, "Test Suite Reduction in Idempotence Testing of Infrastructure as Code," 2017, vol. 10375, pp. 98–115.

[34] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, "Testing Idempotence for Infrastructure as Code," *14th Int. Middlew. Conf. Proc.*, pp. 368–388, 2013.

[35] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, "Automated testing of chef automation scripts," pp. 1–2, 2013.

[36] Y. Jiang and B. Adams, "Co-evolution of infrastructure and source code: An empirical study," *IEEE Int. Work. Conf. Min. Softw. Repos.*, vol. 2015-Augus, pp. 45–55, 2015.

[37] T. Sharma, M. Fragkoulis, and D. Spinellis, "Does your configuration code smell?," *Proc. - 13th Work. Conf. Min. Softw. Repos. MSR 2016*, pp. 189–200, 2016.

[38] A. Weiss, A. Guha, and Y. Brun, "Tortoise: Interactive system configuration repair," *32nd IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 625–636, 2017.

[39] C. Parnin *et al.*, "The Top 10 Adages in Continuous Deployment," *IEEE Softw.*, vol. 34, no. 3, pp. 86–95, 2017.

[40] L. Hochstein and R. Moser, *Ansible: Up and Running*, 2nd Edition. O'Reilly Media, 2017.

[41] J. Geerling, *Ansible for DevOps: Server and configuration management for humans*. Leanpub, 2015.

[42] E. Di Nitto, P. Matthews, D. Petcu, and A. Solberg, *Model-Driven Development and Operation of Multi-Cloud Applications*. Cham: Springer International Publishing, 2017.

[43] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. 2008.

[44] M. Ciolkowski, F. Shull, and S. Biffl, "A family of experiments to investigate the influence of context on the effect of inspection techniques," *Proc. Sixth Int. Conf. Empir. Assess. Softw. Eng. (EASE), Keele, UK.*, pp. 48–60, 2002.

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Berlin Heidelberg, 2012.

[46] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," in *Encyclopaedia of Software Engineering*, Wiley, 1994.

[47] B. A. Kitchenham *et al.*, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, 2002.

[48] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects - a comparative study of students and professionals in lead-time impact assessment," *Empir. Softw. Eng.*, vol. 5, no. 3, pp. 201–214, 2000.

[49] N. Juristo and O. S. Gómez, "Replication of Software Engineering Experiments," Springer, Berlin, Heidelberg, 2012, pp. 60–88.

[50] J. C. Carver, "Towards Reporting Guidelines for Experimental Replications : A Proposal," *1st Int. Work. Replication Empir. Softw. Eng. Res. RESER*, pp. 2–5, 2010.

[51] D. L. Moody, "The method evaluation model: a theoretical model for validating information systems design methods," in *Proc. of ECIS '03*, 2003, pp. 1327–1336.

[52] ISO/IEC, "International Standard ISO/IEC 9126-4," *Inf. Technol. -- Softw. Prod. Eval. -- Qual. Charact. Guidel. their use*, 1991.

[53] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models," *Manage. Sci.*, vol. 35, no. 8, pp. 982–1003, 1989.

[54] S. Vegas, C. Apa, and N. Juristo, "Crossover Designs in Software Engineering Experiments: Benefits and Perils," *IEEE Trans. Softw. Eng.*, vol. 42, no. 2, pp. 120–135, 2016.

[55] R. O. Kuehl, *Design of experiments : statistical principles of research design and analysis*, 2nd ed. Pacific Grove CA: Duxbury/Thomson Learning, 2000.

[56] G. F. Templeton, "A two-step approach for transforming continuous variables to normal: Implications and recommendations for IS research," *Commun. Assoc. Inf. Syst.*, 2011.

[57] K. Maxwell, *Applied Statistics for Software Managers*. Prentice

Hall, 2002.

[58] A. Davey and J. Savla, *Statistical power analysis with missing data: A structural equation modeling approach*. 2009.

[59] P. D. Ellis, *The Essential Guide to Effect Sizes*. Cambridge University Press, 2012.

[60] D. Moody, "The physics of notations: Toward a scientific basis for constructing visual notations in software engineering," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 756–779, 2009.

[61] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Q.*, vol. 13, no. 3, pp. 319–340, 1989.

JULIO SANDOBALÍN received an MSc. in Software Engineering from the Universitat Politècnica de València, Spain, in 2014. He is currently studying to attain a PhD in the Software Engineering and Information Systems Research Group of the same university. His research activities and interests include a wide range of subjects related to model-driven engineering, cloud computing, DevOps and empirical software engineering. His research is currently focused on the continuous delivery of cloud resources based on model-driven engineering and DevOps.

EMILIO INSFRAN is an Associate Professor at the Department of Information Systems and Computation (DISC) of the Universitat Politècnica de València (UPV), Spain. He received a MSc degree in Computer Science from the Cantabria University (Spain, 1994) and a PhD degree from the UPV (Spain, 2003). He worked as a visiting researcher at the UCLouvain (Belgium, 2017), Software Engineering Institute (SEI-CMU) (USA, 2012), and also performed research stays at the University of Twente (the Netherlands), Brigham Young University, Utah (USA), and University of Porto Alegre (Brazil). His research interests are cloud service architectures, DevOps, model-driven development, requirements engineering, and software quality. He has published more than 140 journal and conference papers and he has been working in a number of national and international research projects and in several technology transfer projects with companies.

SILVIA ABRAHÃO is an Associate Professor at the Universitat Politècnica de València (UPV), Spain. She received a PhD in Computer Science from the UPV in 2004. She was a visiting professor at the Carnegie Mellon Software Engineering Institute (2010 and 2012), the UCLouvain (2007 and 2017), and Ghent University (2004). She has (co)authored over 130 peer-reviewed publications. She is currently leading the Spanish Network of Excellence on Software Quality and Sustainability. She is a member of the editorial board of the Software and System Modeling (SoSyM) journal and an Associate Editor of IEEE Software, where she is responsible for Software Quality. Her main research interests include continuous development and integration of cloud services, quality assurance in model-driven engineering, the empirical assessment of software modeling approaches, and integration of usability/UX into software development.