Deployment Specification Challenges in the Context of Large Scale Systems

Miguel Jiménez
University of Victoria
Victoria, British Columbia, Canada
miguel@uvic.ca

Norha M. Villegas
Universidad Icesi
Cali, Valle del Cauca, Colombia
nvillega@icesi.edu.co

Gabriel Tamura
Universidad Icesi
Cali, Valle del Cauca, Colombia
gtamura@icesi.edu.co

Hausi A. Müller
University of Victoria
Victoria, British Columbia, Canada
hausi@uvic.ca

ABSTRACT

Traditionally, the focus of software deployment has been mainly on the infrastructure to realise deployment and configuration (D&C) of complex and distributed systems, with an increasing interest in deployment of internet of things and cyber-physical systems. Advances in job scheduling, storage orchestration, containerized applications, along with agile practices such as continuous integration and microservices architecture, have improved the state of the practice. However, little effort has been devoted to the need for D&C specifications to support the various levels of detail and abstraction present in large-scale systems. The understanding of the software components hierarchy has shifted from the comprehension of design artefacts, usually specified with static diagrams, to the understanding of runtime concepts. The DevOps movement has dramatically influenced how and when deployment is realised, but little has been done from the software perspective in terms of documentation and linkage between design and runtime artefacts in the sense of software specification as such. This paper presents an overview of the state of the art of deployment requirements for large-scale, distributed and complex software and its automation and characterises a set of deployment specification challenges intended as starting points for advancing the field of software deployment.

CCS CONCEPTS

• **Computer systems organization** → *Distributed architectures*;
• **Software and its engineering** → *Software post-development issues*;

KEYWORDS

DevOps, Continuous Software Deployment, Deployment Specification, Continuous Integration, Continuous Configuration, Runtime Artefacts, Models at Runtime

ACM Reference format:
Miguel Jiménez, Norha M. Villegas, Gabriel Tamura, and Hausi A. Müller. 2017. Deployment Specification Challenges in the Context of Large Scale Systems. In *Proceedings of 27th Annual International Conference on Computer Science and Software Engineering , Markham, Ontario, Canada, November 2017 (CASCON'17),* 7 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Today's development environments offer great tool support to assist engineers in executing, configuring and monitoring software components. Furthermore, advances in infrastructure management facilitate server provisioning and application deployment and redeployment. However, there is little support for mapping each component to the corresponding artefacts at runtime. Tool support on the software side has not advanced at the same pace as infrastructure management has. Deployment specifications are mostly used as static documentation that poorly reflects the actual information required to deploy a system. Moreover, deployment specification tools lack support for different levels of domain expertise and collaboration among the various stakeholders. The proliferation of cloud, container and microservices infrastructure has amplified the need for deployment specification.

Architecture and deployment specifications are vital in understanding the different parts composing a system [17]. Proper tool and notation support will have the effect of promoting specification reuse; and more importantly, automating and tracking continuous deployment and configuration. This technology will provide engineers with the means to understand complex systems, such as large-scale smart cyber-physical systems (CPS), from different perspectives. Furthermore, it will provide systems with a foundation to build mechanisms to reason about their needs and exploit other systems. To do this, engineers require a deployment specification that is complete in detail, but also tools to navigate and visualise the system. They require a specification that supports specifying cross-cutting concerns, such as quality and security requirements [12]. Moreover, specification technology must offer support for explicitly linking design and runtime concepts. In this paper, we expand on the aforementioned needs, and state a set of challenges in deployment specification with the aim of advancing the field of software deployment.

The remainder of this paper is organised as follows: Section 2 presents a motivational example along with a high-level description of the requirements for deploying large-scale systems. Section 3 states our vision regarding deployment specification technology. Section 4 presents an overview of the state of the art in the field of software deployment. Section 5 lists a set of deployment specification challenges aligned with our vision. And Section 6 concludes this paper.

2 MOTIVATIONAL EXAMPLE

We present a motivational example highlighting relevant deployment concerns in the context of a modern large-scale system, without considering the details of the underlying application architectures or technologies.

Figure 1 depicts a CPS for smart transportation in the context of a smart city. The various components illustrated are inspired by Sipresk [18], a platform for performing analytics on urban transportation data. On the one hand, the rungs in the illustration represent a multi-tier software architecture composed of layers ranging from the physical infrastructure to the visualisation of data analytics. On the other hand, the columns represent cross-cutting dimensions of the system, in which different actors perform different tasks and have diverse interests and expectations.

The smart transportation system comprises several interacting applications that may be managed by several teams, implemented in different technologies, and governed by different policies. There are, however, executive stakeholders interested in understanding the solution as a whole, such as transportation managers, traffic engineers, and decision makers. There are also technical stakeholders interested in knowing the necessary details and quality requirements for each subsystem. Moreover, these stakeholders may belong to different teams and, therefore, they profoundly depend on requirements specification to accomplish their responsibilities. Although the applications composing the system may be developed independently, possibly in different contexts, they share physical and logical resources. Quality expectations of stakeholders about these resources impose functional and non-functional requirements that constrain, in particular, the deployment and configuration (D&C) of the system.

The diagram depicted in Figure 1 represents a holistic abstraction of the system. It is intended to highlight the various subsystems composing the solution. Such abstraction, however, tells little if anything to an operations engineer about the deployment of the subsystems. One single diagram is insufficient to offer high-level views of the system, the applications, the underlying infrastructure, and, at the same time, detailed technical views and deployment topologies. Those views constitute relevant artefacts for the various system actors to communicate and understand concerns they have about the system, related to design, such as the UML's use case and component diagrams; development, such as the UML's activity and sequence diagrams; testing, such as context diagrams and flow charts; and D&C, such as the UML's deployment diagram. However, these diagrams become out of date very quickly with any change in the configuration.

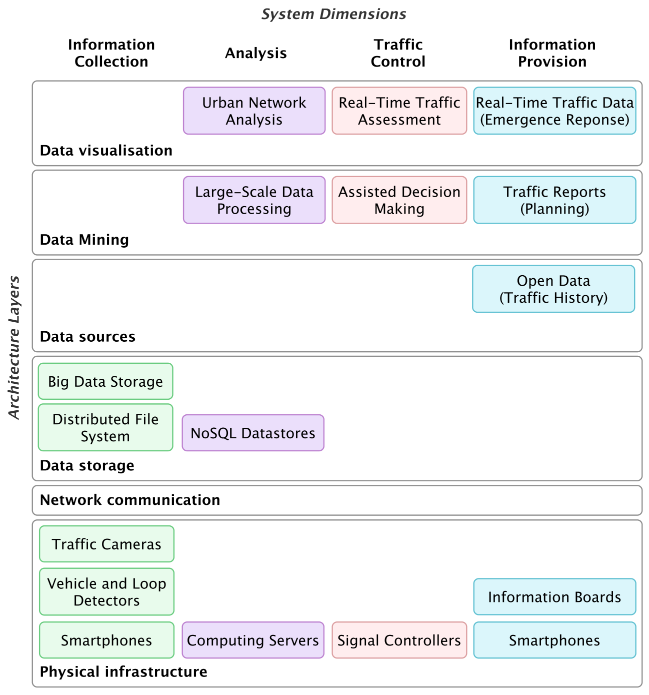

Figure 1: Layers and Dimensions of a Smart Transportation System

The deployment process of the smart transportation system involves many cross-cutting concerns such as connectivity, performance, and security requirements, as well as resource availability, policies and best practices [12]. Different domain experts and technical staff are responsible to address these concerns, which makes deployment specification a complicated process. Moreover, the various applications composing the system may be independently deployed in different environments, such as development, staging, quality testing, and production, each having specific requirements.

Deployment automation is essential to reduce personnel requirements and improve collaboration and knowledge reuse. It also reduces deployment complexity, guarantees configuration integrity and consistency, and allows separation of concerns [12]. Specification plays a key role in automating deployment; it supports the aforementioned desired properties, and enables incremental collaboration between actors across the multiple dimensions encompassing the smart transportation system.

3 OUR VISION

We envision a deployment technology that supports the reuse of deployment specifications and automation, as well as collaboration among the various actors involved in specification creation and evolution. This technology allows connecting human and machine readable representations of the software components and computing infrastructure. Deployment specification tools must maintain the correspondence between all these representations, including changes that occur at design and execution time.

Specification tools must support the modelling of software deployment at different levels of abstraction. This is key for understanding the deployment of a system from different perspectives, and for promoting reusability. These tools must orchestrate the execution of distributed causally-connected specifications developed within the context of each component. For instance, in the case of back-end technology, a system deployment could rely on existing mechanisms for configuring and activating each component in the same way an object instantiates another in object-oriented programming. Such technology would reduce the time invested in deployment design, as it allows reusing specifications and enables stakeholders to focus on a smaller set of concerns.

4 OVERVIEW OF THE STATE OF THE ART

This section discusses relevant prior works in the area of software deployment. We show that research has been mainly focused on the infrastructure to realise D&C of software systems. The primary focus in existing works is on D&C infrastructure for grid computing and distributed real-time embedded (DRE) systems [25], with an increasing interest in IoT and CPS.

4.1 Deployment for Large-Scale Infrastructures

DeployWare [14] addresses the deployment of distributed and heterogeneous software systems on large scale infrastructures. It is a Fractal [5] based framework for the orchestration of deployment tasks and management of software dependencies. It provides a domain specific modelling language (DSML) to describe deployments based on a meta-model, a virtual machine that interprets these descriptions, and a graphical console to manage and monitor deployed systems. The metamodel as well as the DSML are focused on the installation, configuration, activation, and deactivation of software artefacts. Its deployment specifications are heavily related to the artefacts and their corresponding computing nodes. Although it does consider different technical roles such as, system administrator and technology expert, its deployment specifications are not suitable to be analysed by executive staff. We find them limited in the context of large-scale systems, as described in Section 2. DeployWare lacks support for application reconfiguration.

GoDIET [6] is a tool for the configuration, launch, and management of DIET [7] applications. It expects as input an XML file describing available computing and storage resources, along with a desired distribution of DIET agents on the resources. From such description, GoDIET generates all necessary configuration files, and launches agents and servers in appropriate hierarchical dependency-aware order. GoDIET limits the deployment specification to the input XML file. There are two main concerns in such file: storage and computing devices, and the hierarchy of deployment agents. GoDIET lacks support for application reconfiguration.

Toure et al. [27] present a decentralised and hierarchical deployment approach using the TUNe [4] autonomic management system. Their solution takes into account the topology and characteristics of the computing grid via a deployment schema. The schema is a UML profile that describes low-level deployment properties such as whether or not to use a log service, or a flush interval. These properties are then mapped to a group of nodes from a grid schema that describes the computing infrastructure. The main contribution of this work resides in its hierarchical approach to deployment; every task is divided into less complex subtasks, and managed by an independent TUNe instance. This is of particular relevance when deployments involve thousands of computing nodes (e.g., IoT or COS applications). In such scenarios, centralised approaches do not scale well because the master node tends to run out of resources.

The OMG D&C specification [21] defines standard formats for metadata, and interfaces to facilitate the D&C of component-based applications. It standardises many aspects of the component application development life cycle, including component configuration, assembly, packaging, and package configuration and deployment. The standard also defines a deployment process that starts after the software is packaged and published, and establishes stages ranging from the software installation to its launch. We find this D&C standard is nicely aligned with our vision, in the sense that there is a clear separation of responsibilities per actor across the development and deployment stages. It also defines independent models to describe the application assembly and the target environment. However, this standard is intended for component models, leaving out the myriad of technologies that actually interact in large-scale systems.

The Deployment and Configuration Engine (DAnCE) [9] is a QoS-enabled middleware compliant with the OMG D&C specification. It simplifies the configuration, deployment, and management of application services in the context of large-scale distributed real-time and embedded (DRE) systems. The Locality-Enhanced Deployment and Configuration Engine (LE-DAnCE) [22] extends DAnCE, overcoming the lack of appropriate optimizations specific to the architecture from the OMG D&C standard. According to [22], the architecture yields various overhead sources that cause performance bottlenecks, thus negatively affecting deployment latency.

Copil et al. [8] present SYBL, a composable and extensible language for controlling elasticity in cloud applications. SYBL's directives allow controlling elasticity by defining relations between elastic properties categorised into three dimensions, namely cost, quality and resource. Although SYBL does not provide deployment mechanisms, its simple yet powerful syntax allows representing first-class elasticity conditions that would be otherwise hidden in programming code. For instance, `Cons: CONSTRAINT cpuUsage < 80%` creates a constraint at the application level to maintain the CPU utilisation under 80%.

Eilam et al. [12] present a model-driven approach that allows reusing application deployment through an annotation mechanism to configure different deployment properties. Application developers describe the structure of the application and its requirements as a logical application topology. This topology represents the application components, and their connectivity, hosting, and configuration requirements. From there, application deployers and business users annotate the logical topology with additional requirements such as performance and security. Although their prototype only produces network configurations, we find it very aligned with our vision. Their approach enables domain experts to define model transformations and constraints in their area of expertise, taking complexity away from the end application deployer. Eilam et al. [11, 13] extend this work to bridge the gap between existing imperative scripts and declarative model-based planning and validation. Their work enforces a strict separation between resource structure representation and the operational model.

4.2 Deployment for Cyber-Physical Systems

In CPS, D&C of the various application components must happen in a timely manner. In mission-critical CPS, D&C capabilities are also expected to be reliable and failure tolerant, where failure recovery is timely and predictable, and minimises the negative impact on quality of service. Moreover, mission-critical CPS are exposed to frequent redeployment and reconfiguration, including incremental D&C of its constituents, which to some extent must be supported predictably [26]. That is, mission-critical CPS require support for uncertainty management within their deployment mechanisms.

In Extensible CPS, both the applications and the computing resources in which these systems are deployed are highly dynamic. In [25], Pradhan extends LE-DAnCE to achieve a resilient D&C infrastructure to deploy CPS applications and dynamically reconfigure them once they are deployed. Pradhan's infrastructure implements distributed group membership, a mechanism to detect when nodes leave or join a cluster of available resources for deployment. This approach also implements a mechanism for state preservation in case of failure, by distributing node-specific deployment plans. However, the state is stored in memory, which is vulnerable to process failures. As described in [25], application deployment is specified using a global deployment plan, which describes the various application components, their implementation, and the way to connect them. This approach does not consider all the different requirements and restrictions associated with the D&C of CPS, and is rather focused on the component topology specific to the Lightweight CORBA Component Model. Thus, it does not support infrastructure provisioning and configuration. In light of our vision, Pradhan's contribution is focused on deployment execution, rather than its design and evolution.

CPS and IoT applications integrate an ever-increasing number of heterogeneous devices to be aware of their environment, and to manipulate it. These devices often provide constrained execution environments with limited capabilities. DIANE [28] is a framework for the generation of deployment topologies for IoT cloud applications, and respective provisioning of the available physical infrastructure. DIANE is based on MADCAT [17], a methodology for architecture and deployment of cloud application topologies. It uses a declarative language to describe *technical units* (i.e., application components), *deployment units* (i.e., infrastructure requirements), and *deployment instances* (i.e., concrete deployments on actual machines of the infrastructure). Although the deployment units and instances are constraint-based, the technical units require a procedural workflow specification per artefact. As DIANE is based on MADCAT, the main concerns approached are related to deployment realisation, rather than the requirements we established in Sections 2 and 3.

4.3 Discussion

Large-scale deployment is traditionally approached by addressing the size and complexity of the computing infrastructure [6, 14, 27]. In this scenario, complexity revolves around issues that arise during deployment execution, such as synchronisation and proper execution order of workflow tasks, failure detection, and state preservation. These issues become more manageable in the SOA world (e.g., the microservices architecture), in which each service may be deployed independently on its own infrastructure resources, and scalability is mainly managed at runtime.

As presented in Section 2, modern large-scale systems pose additional deployment requirements. These requirements are rather related to how systems are engineered and managed, instead of the infrastructure supporting their execution. We believe that deployment specification plays an important role in fulfilling the aforementioned emerging requirements.

Novel technologies such as containerised applications and modern job orchestration are fading away direct management of computing resources (i.e., computing resources lose their identity). This is of particular relevance for our vision, because there is a clear separation between infrastructure management and application deployment. However, this poses new challenges in terms of the mapping between the software artefacts and the runtime concepts understood by such technology.

5 RESEARCH CHALLENGES

In Sections 2 and 3 we defined the requirements for the deployment of large-scale, distributed and complex software, and its automation. We now characterise four deployment specification challenges (i.e., CH1-4) aligned with our vision, as a starting point for advancing the field of software deployment. We believe there is a clear need for better specification notations, however, deployment specification and evolution tools are the real enablers for their adoption.

5.1 CH1: Completeness and Expressiveness of Deployment Specification

Stakeholders expect documentation that is compliant with the application under development in different levels of detail and abstraction [17]. This requires notations with the proper power of expression for specifying the various concerns encompassing D&C, and tools to navigate and visualise these specifications.

To the best of our knowledge, deployment is specified using either UML, informal diagrams, code (i.e., textual models), or a combination of them. However, we believe it is necessary to perform a systematic literature review to determine and characterise the elements involved in D&C and the notations available to specify them.

The UML deployment diagram is a well-known notation for describing the architecture of a software system. It provides graphical elements to represent a static perspective of the various elements involved in the deployment life cycle. It has been refined in several versions of the UML specification, yet it remains to be one of the least adopted diagrams within the Model-Driven Engineering (MDE) community [16], and among UML users [24]. According to a survey conducted in 2007 with 80 professional developers [20], Nugroho and Chaudron found that most UML models have a low degree of completeness; that is, they specify only a small part of the required system elements. In the case of the UML deployment diagram, some elements from the software, hardware and network perspectives remain missing, including requirements for infrastructure provisioning (e.g., storage, memory, and processing capacity of the computing nodes), network configuration requirements (e.g., network security zones and access rules) as well as elasticity requirements [8, 17]. The software part, however, is incomplete for a different reason. The level of abstraction needs to be at a higher level than it actually allows when working with large scale systems. Currently, the diagram does not scale up well, enforcing the creation of a bloated specification or many specifications containing duplicate elements. This is mainly because deployment specification tools are intended for producing static documentation, instead of managing the software as a dynamic entity where the configuration changes over time. The benefit of using UML in this context is then limited to communicating architectural decisions, affecting little of the actual deployment.

Informal diagrams are non-standard graphical notations used to communicate insights regarding an application deployment. As these diagrams are simple in syntax and semantics, they are popular in environments with lean documentation requirements. However, they are not machine readable. D&C for large-scale systems requires a specification that enables automation, and therefore, specification tools should support the integration of existing notations. As mentioned in Section 3, deployment tools must allow connecting human and machine readable representations of the elements involved in D&C.

5.2 CH2: Specification of Transversal Concerns

Two important concerns associated with software deployment are infrastructure provisioning and application deployment. New trends in software engineering bring both concerns together in scenarios of continuous integration and deployment. D&C is a responsibility shared by various teams, including design, development, operations, and security. Therefore, the notation used should allow specifying transverse information along with the necessary details for these teams to design high-level guidelines (e.g., architectural patterns and security policies) and orchestrate provisioning and deployment tasks collaboratively. In addition, these specifications have a documentation role and are especially beneficial for new members of the team and people who do not participate in their creation [15].

Different technical levels of stakeholder proficiency must be considered in the development of D&C specifications. Specification tools must provide high-level views of the application architecture and its deployment to executive stakeholders but also offer detailed technical views for specialised staff [17]. Architecture Description Languages (ADL) are essential in realising this, given the relation between architecture and deployment. However, ADLs do not support multiple views and are often awkward to use in the workplace because of lack of good tools [23]. One notation may not be enough to express different levels of abstraction adequately. It is vital that these tools and notations allow the connection of the specifications (i.e., establishing dependencies among them), as well as the runtime models[1] that represent them.

5.3 CH3: Linking Design and Runtime Deployment Concepts

Traditionally, the mapping between design and runtime concepts for deployment has been rather direct. In Java EE, for example, Web application ARchives (WAR) and Java ARchives (JAR) are the manifestation of at least one software component. This relationship is directly represented in the UML deployment diagram with a `manifest` relationship connecting the component(s) and the artefact. As a result, the workflow procedures contain the necessary code to build the component(s) into a compressed file. Modern deployment tools introduce new concepts that require manual mapping to take advantage of deployment automation mechanisms. As an example, Kubernetes,[2] a container orchestration tool for automated container deployment, scaling, and management, requires as input descriptors in terms of *Pods*, *Services*, *Namespaces*, *Jobs*, *ReplicaSets*, *DaemonSets*, among others; similar cases are Mesos Marathon,[3] Singularity,[4] and Apache Aurora.[5] The latter requires a configuration file written in a python-based Domain Specific Language, which increases the level of difficulty in a scenario of generative configuration.

Systematic approaches to maintain the correspondence between diagrams and code are rarely used in practice (e.g., UML models [19]). However, software and its deployment specification evolve over time. As this mapping between software and runtime deployment concepts becomes more complex, staff skills required to realise software deployment are higher.

Linking D&C specifications and runtime models facilitates D&C requirements realisation and documentation maintenance. Furthermore, establishing causal connections among these models provides systems with support for change propagation across the different dimensions encompassing them. There is a need for libraries and frameworks to connect specifications with their runtime representation.

5.4 CH4: Adaptivity and Configuration Management at Runtime

The dynamic nature of the cloud computing paradigm enables architectural agility that allows applications to evolve along with application requirements dynamically [17]. Cloud native software requires mechanisms for seamlessly integrating new components and updating the ones already running as part of its normal operation. Furthermore, the flexibility of the cloud enables today's systems to update infrastructure resources at execution time, such as Software Defined Networks [2] and Software Defined Infrastructures [1]. Regardless of the characteristics of these mechanisms, deployment specifications should remain updated with respect to the actual system deployment as it adapts. Runtime representations of these specifications, as explained in the previous challenges, facilitate maintaining the correspondence between the various system views and its architecture and deployment. The challenge here is to depict the D&C evolution, not only by performing isolated architectural changes in the infrastructure but especially tracking and visualising them in the specification effectively.

Deployment specification tools must support decision making by allowing practitioners to assess different configurations and

[1]Also found in literature as models@run.time [3]

[2]https://kubernetes.io
[3]https://mesosphere.github.io/marathon
[4]http://getsingularity.com
[5]http://aurora.apache.org

evaluate the track of architectural changes. In the same sense, systems themselves may take advantage of these techniques via the associated D&C runtime models.

According to the quality concerns, application architects may need to exploit the cloud elasticity to scale up and down infrastructure resources [17]. D&C notations must support the explicit incorporation of elasticity requirements. However, elasticity is a multi-dimensional view and, therefore, a complex problem [10]. It requires evaluating three different factors namely resource, cost, and quality; and most importantly, their relation. Therefore, the integration of such requirements should be made through specialised models, such as SYBL [8].

6 CONCLUSIONS

The D&C of complex and distributed software requires technological capabilities that pose specific challenges in its specification. We strongly believe that the current research focus on the computing infrastructure to realise the deployment of such systems is not enough for coping efficiently with these challenges, especially when the distribution is as wide as in CPS, such as smart cities and cloud environments.

The challenges we characterize in this paper are: (1) the need for complete deployment notations to allow stakeholders specifying and visualising large-scale deployments from different perspectives and levels of abstraction; (2) the need for deployment notations to support cross-cutting concerns; (3) the need for notation and tool support for linking design and runtime deployment concepts; and (4) tool support for the evolution of deployment specifications and configuration management at runtime.

By identifying this initial set of challenges, we aim to motivate researchers to investigate deployment specification in the context of large-scale systems. Overcoming these challenges would imply not only significant savings in engineers and computing time, but also a more seamless process of re-deployment in continuous integration; along with the possibility of self-learning when and how to adapt portions of a CPS based on context-based performance indicators.

ACKNOWLEDGMENTS

This work was funded in part by the National Sciences and Engineering Research Council (NSERC) of Canada, IBM Canada Ltd., the University of Victoria and Universidad Icesi. The authors would like to thank IBM's Centre for Advanced Studies, Canada for supporting this project.

REFERENCES

[1] Nasim Beigi-Mohammadi, Hamzeh Khazaei, Mark Shtern, Cornel Barna, and Marin Litoiu. 2016. On Efficiency and Scalability of Software-Defined Infrastructure for Adaptive Applications. In *Proceedings 13th IEEE International Conference on Autonomic Computing (ICAC 2016)*. 25–34. https://doi.org/10.1109/ICAC.2016.39

[2] Nasim Beigi-Mohammadi and Marin Litoiu. 2017. Engineering Self-Adaptive Applications on Cloud with Software Defined Networks. In *Proceedings IEEE International Conference on Cloud Engineering (IC2E 2017)*. 9–12. https://doi.org/10.1109/IC2E.2017.43

[3] Gordon Blair, Nelly Bencomo, and Robert B. France. 2009. Models@run.time. *Computer* 42, 10 (Oct 2009), 22–27. https://doi.org/10.1109/MC.2009.326

[4] Laurent Broto, Daniel Hagimont, Patricia Stolf, Noel Depalma, and Suzy Temate. 2008. Autonomic Management Policy Specification in Tune. In *Proceedings 23rd ACM Symposium on Applied Computing (SAC 2008)*. ACM, New York, NY, USA, 1658–1663. https://doi.org/10.1145/1363686.1364080

[5] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, and Jean-Bernard Stefani. 2006. The FRACTAL component model and its support in Java. *Software: Practice and Experience* 36, 11-12 (2006), 1257–1284. https://doi.org/10.1002/spe.767

[6] Eddy Caron, Pushpinder Kaur Chouhan, and Holly Dail. 2006. *GoDIET: a deployment tool for distributed middleware on Grid'5000.* Research Report RR-5886. INRIA. https://hal.inria.fr/inria-00071382

[7] Eddy Caron and Frédéric Desprez. 2006. Diet: A scalable toolbox to build network enabled servers on the grid. *The International Journal of High Performance Computing Applications* 20, 3 (2006), 335–352.

[8] Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, and Schahram Dustdar. 2013. SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications. In *Proceedings 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)*. 112–119. https://doi.org/10.1109/CCGrid.2013.42

[9] Gan Deng, Jaiganesh Balasubramanian, William Otte, Douglas C. Schmidt, and Aniruddha Gokhale. 2005. *DAnCE: A QoS-Enabled Component Deployment and Configuration Engine.* Springer Berlin Heidelberg, Berlin, Heidelberg, 67–82. https://doi.org/10.1007/11590712_6

[10] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong-Linh Truong. 2011. Principles of Elastic Processes. *IEEE Internet Computing* 15, 5 (Sept 2011), 66–71. https://doi.org/10.1109/MIC.2011.121

[11] Tamar Eilam, Michael Elder, Alexander V. Konstantinou, and Ed Snible. 2011. Pattern-based composite application deployment. In *Proceedings 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)*. 217–224. https://doi.org/10.1109/INM.2011.5990694

[12] Tamar Eilam, Michael H. Kalantar, Alexander V. Konstantinou, and Giovanni Pacifici. 2005. Reducing the complexity of application deployment in large data centers. In *Proceedings 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005)*. 221–234. https://doi.org/10.1109/INM.2005.1440790

[13] Kaoutar El Maghraoui, Alok Meghranjani, Tamar Eilam, Michael Kalantar, and Alexander V. Konstantinou. 2006. Model Driven Provisioning: Bridging the Gap Between Declarative Object Models and Procedural Provisioning Tools. In *Proceedings 7th International Middleware Conference (Middleware 2006)*. Springer-Verlag New York, Inc., New York, NY, USA, 404–423. http://dl.acm.org/citation.cfm?id=1515984.1516015

[14] Areski Flissi, Jérémy Dubus, Nicolas Dolet, and Philippe Merle. 2008. Deploying on the Grid with DeployWare. In *Proceedings 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008)*. 177–184. https://doi.org/10.1109/CCGRID.2008.59

[15] Truong Ho-Quang, Regina Hebig, Gregorio Robles, Michel R. V. Chaudron, and Miguel Angel Fernandez. 2017. Practices and Perceptions of UML Use in Open Source Projects. In *Proceedings 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP 2017)*. IEEE Press, Piscataway, NJ, USA, 203–212. https://doi.org/10.1109/ICSE-SEIP.2017.28

[16] John Hutchinson, Mark Rouncefield, and Jon Whittle. 2011. Model-driven Engineering Practices in Industry. In *Proceedings 33rd International Conference on Software Engineering (ICSE 2011)*. ACM, New York, NY, USA, 633–642. https://doi.org/10.1145/1985793.1985882

[17] Christian Inzinger, Stefan Nastic, Sanjin Sehic, Michael VÃűgler, Fei Li, and Schahram Dustdar. 2014. MADCAT: A Methodology for Architecture and Deployment of Cloud Application Topologies. In *Proceedings 8th International Symposium on Service Oriented System Engineering (SOSE 2014)*. 13–22. https://doi.org/10.1109/SOSE.2014.9

[18] Hamzeh Khazaei, Saeed Zareian, Rodrigo Veleda, and Marin Litoiu. 2016. *Sipresk: A Big Data Analytic Platform for Smart Transportation.* Springer International Publishing, Cham, 419–430. https://doi.org/10.1007/978-3-319-33681-7_35

[19] Ariadi Nugroho and Michel R.V. Chaudron. 2007. A Survey of the Practice of Design – Code Correspondence amongst Professional Software Engineers. In *Proceedings 1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. 467–469. https://doi.org/10.1109/ESEM.2007.69

[20] Ariadi Nugroho and Michel R.V. Chaudron. 2008. A Survey into the Rigor of UML Use and Its Perceived Impact on Quality and Productivity. In *Proceedings 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2008)*. ACM, New York, NY, USA, 90–99. https://doi.org/10.1145/1414004.1414020

[21] OMG. 2006. Deployment and Configuration of Component-based Distributed Applications Specification—Version 4.0. *Object Management Group* (2006).

[22] William R. Otte, Aniruddha Gokhale, and Douglas C. Schmidt. 2011. Predictable Deployment in Component-based Enterprise Distributed Real-time and Embedded Systems. In *Proceedings 14th International ACM Sigsoft Symposium on Component Based Software Engineering (CBSE 2011)*. ACM, New York, NY, USA, 21–30. https://doi.org/10.1145/2000229.2000233

[23] R. K. Pandey. 2010. Architectural Description Languages (ADLs) vs UML: A Review. *ACM SIGSOFT Software Engineering Notes* 35, 3 (May 2010), 1–5. https://doi.org/10.1145/1764810.1764828

[24] Marian Petre. 2013. UML in Practice. In *Proceedings 35th International Conference on Software Engineering (ICSE 2013)*. IEEE Press, Piscataway, NJ, USA, 722–731.

http://dl.acm.org/citation.cfm?id=2486788.2486883

[25] Subhav Pradhan. 2016. *Algorithms and Techniques for Managing Extensibility in Cyber-Physical Systems.* Ph.D. Dissertation. Vanderbilt University.

[26] Subhav Pradhan, Aniruddha Gokhale, William R. Otte, and Gabor Karsai. 2013. Real-time Fault Tolerant Deployment and Configuration Framework for Cyber Physical Systems. *SIGBED Rev.* 10, 2 (July 2013), 32–32. https://doi.org/10.1145/2518148.2518170

[27] Mahamadou Toure, Patricia Stolf, Daniel Hagimont, and Laurent Broto. 2010. Large Scale Deployment. In *Proceedings 6th International Conference on Autonomic and Autonomous Systems (ICAS 2010).* 78–83. https://doi.org/10.1109/ICAS.2010.20

[28] Michael Vögler, Johannes M. Schleicher, Christian Inzinger, and Schahram Dustdar. 2015. DIANE - Dynamic IoT Application Deployment. In *Proceedings IEEE International Conference on Mobile Services (MS 2015).* 298–305. https://doi.org/10.1109/MobServ.2015.49