Environmental IoT: Programming Cyber-physical Clouds with High-level System Specifications

Roberto Rodrigues Filho, Barry Porter and Gordon Blair
School of Computing & Communications
Lancaster University, Lancaster, UK
{r.rodriguesfilho, b.f.porter, g.blair}@lancaster.ac.uk

***Abstract*—The Environmental IoT is a project where we investigate the potential of an integrated distributed system consisting of an Internet of Things (IoT) and a Cloud Computing infrastructure. The resulting complex distributed system will be used to support deep understanding of the natural environment inter-dependencies and the management of the natural environment through appropriate interventions. In this paper, we discuss our approach to program this resulting complex distributed system with high-level system specifications in the environmental science context. The high-level specification encapsulates environmental science concepts and conveys the system's overall goals. The approach consists of three refinement steps that translate the high-level specification into the accordingly behaviour on the resulting distributed system. This process captures the abstract requirements of scientists and supports runtime adaptation.**

I. Introduction

Modern distributed systems are increasingly heterogeneous in their composition. Our *Environmental IoT* project is one example of this. It combines cutting edge Internet of Things (IoT) and Cloud Computing technology to provide real-time data streams to support climate science. The provision of real-time data is intended to provide a deep understanding of the environmental inter-dependencies and thus enable holistic management strategies.

Combining the IoT and Cloud Computing represents a highly complex distributed system. Programming the IoT, for example, requires very specific knowledge of energy management, routing protocols, signal processing, and data fusion and aggregation techniques. Programming for the cloud, meanwhile, requires an understanding of multi-tier architectures and virtual machine design and management.

Putting them both together results in a very complex infrastructure that is hard to program as a single integrated system. Additionally, considering the Environmental IoT in particular, we need a way for environmental scientists to describe the overall aims of the system in terms of both functional and non-functional (e.g. data quality) requirements. Finally, the highly volatile nature of IoT deployments and the variable multi-tenant nature of cloud systems, results in a pervasive need for runtime adaptation in the resulting infrastructure.

The aim of this paper is to describe a way to program the resulting complex distributed system composed by an IoT and Cloud Computing infrastructure, in a way that can capture the abstract requirements of scientists, and exhibits inherent support for pervasive adaptation. We also argue that the IoT and the cloud should be treated as equal partners in a distributed system, both able to support computation in a symbiotic manner.

The remainder of this paper is organised as follows: Section II presents the state of art of using IoT and Cloud Computing technologies from the interoperability, programmability and high-level system specification perspectives. Section III shows the potential of combining IoT technologies and Cloud Computing infrastructure, Section IV brings a discussion on Environmental IoT describing its goals and motivation, and Section V describes our approach. Finally, Section VI presents the main challenges and open issues, and Section VII reports the final considerations of this work.

II. Related Work

The combination of IoT and Cloud Computing technologies is a "hot topic" in the distributed systems research community, as shown in [1]. These initial works, despite their clear importance and contribution in highlighting the potential of combining both infrastructures, only discuss low-level programming aspects and the use of IoT to only push data towards a Cloud Computing platform.

These works such as [2], [3] and [4], discuss the interoperability of both infrastructures by connecting the sensor network using network overlay protocols to provide interoperability between the two platforms. They also mention the use of application-level protocols like HTTP or a publish/subscribe approach to push data collected from the IoT infrastructure to the cloud.

Third-party Platforms as a Service (PaaS) such as *ThingSpeak.com*, *Nimbits.org*, *Xively.com* and *ioBridge.com* are also used in that sense, i.e. to receive IoT data, store that data, and make it available for applications interested in consuming it.

This initial view of having IoT only collecting and sending data to a Cloud Computing infrastructure seems very limited. Especially from a programmability perspective, since each platform would have to be programmed individually considering fine-grained details specific of each, which increases the challenges of programming the combined distributed system as a whole.

The provision of a high-level system specification has been the focus of several work. Works such as [5] [6] [7] use model-driven engineering and domain-specific languages to abstract the complexity of underlying systems to make the

programmability of such systems easier. We pinpoint that [5] does not provide the level of abstraction required since their representation is semantically closer to the technical solution than to the application domain, while code generation in [6] and [7] makes it difficult to provide runtime adaptivity.

III. Internet of Things and Cloud Computing

In this section, we identify the general advantages of unifying IoT and Cloud Computing, discussing the relative benefits and the complementarity of the two technologies.

First the IoT offers localized computation that is close to the point of data collection, enabling fast identification of interesting events and low-latency decisions on actuation activities. On the other hand IoT technology is often unreliable and has significant resource constraints, including energy, memory capacity, processing power, network bandwidth, etc.

Secondly the Cloud Computing paradigm offers the illusion of infinite resources with high reliability. This supports, for example, massive scale and long term storage of data, complex and computationally intensive models of climate predictions. However, Cloud Computing is distant from the data collection area, resulting in much more higher latency on actuation decisions.

Symbiotically unifying IoT and Cloud Computing results in a combined distributed system capable of handling large amounts of real-time data collected from physical environments and making that data available to interested applications, enabling novel application domains like environmental science.

In the next section we briefly describe the Environmental IoT project, which serves as a case study for our work.

IV. Environmental IoT

The Environmental IoT project represents a first attempt to instrument and manage a catchment in all its facets, across different geographical locations and at all its scales, for the benefit of the key stakeholders associated with that catchment - farmers and associated agricultural businesses, the water industry, tourists and tourism related business, and society more generally. This has the potential to completely transform these associated businesses, enabling critical areas such as integrated land and water management, coastal zone protection and precision agriculture.

We focus on one specific geographic region around Conwy, typical of many rural areas supporting important industries including agriculture, forestry, tourism and fishing but facing huge challenges brought about by climate change and conflicting demands on land/water resources.

One of the main challenges in the Conwy and many other catchments / landscapes is the potential conflict arising from the needs of different industries, e.g. agriculture, water, tourism and urban development and the need for decision support tools to future-proof against climate change. In the past, 'silo management' has often resulted in a development by one industry negatively impacting on another (e.g. intensification of agriculture reducing water quality).

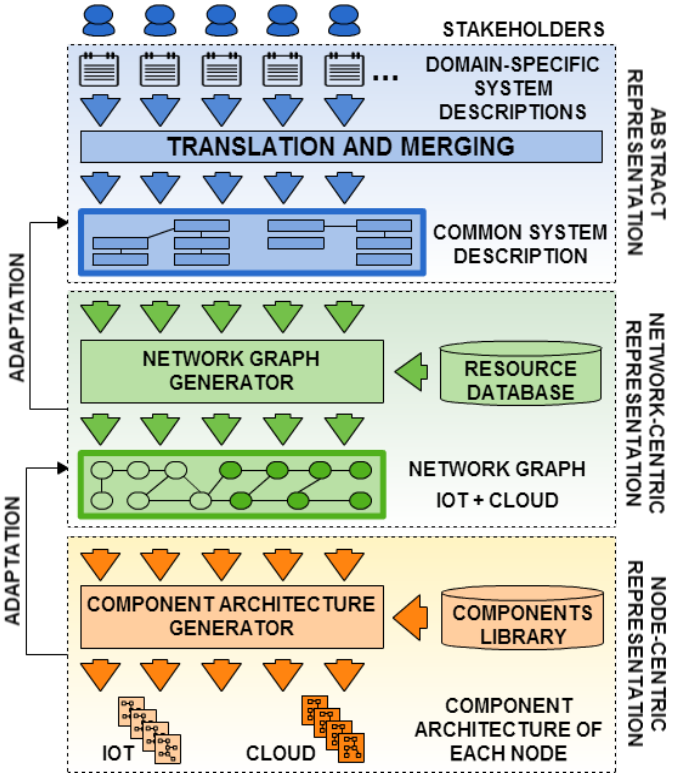

Fig. 1. An Overview of our Approach.

The Environmental IoT project provides a major opportunity to bring together data and assets from across different domains (soil, water, plant ecology, animals) and organisations (e.g. public, regulatory, industry) to enable integrated problem solving across all of the industries in the area.

The overall aim of this project is to develop an Environmental Internet of Things, supported by an associated cloud infrastructure with the view of enabling a paradigm shift in Environmental Science and associated environmental management. The practical motivation for the research is to develop a set of principles, techniques and tools that directly support our goal of a paradigm shift in this area.

V. Our Approach

Our approach aims to enable the description of the combined distributed system's goals through the use of a domain-specific notation capable of expressing environmental science concepts. The process of breaking these concepts down into functional components that can be deployed in the resulting distributed system is a complex process, for which we propose three distinct stages of increasingly refined representation, as illustrated in Fig. 1. In this section we describe each stage, with regards to its inputs, outputs and its operations.

A. The Abstract Representation

The Abstract Representation is the first stage. It depicts the process of merging and translating Domain-specific System Descriptions (DSD) into a Common System Description

(CSD). This consists of breaking down very domain-specific concepts into a common high-level collection of system goals about which further stages of our toolchain can reason.

The DSD is the input of this stage. We envision that the DSD can be represented by *any* domain-specific notation. In fact, the notation used to provide the description should match the background and interests of the particular stakeholder. For example, a stakeholder with a background in mathematics, who is interested in providing a mathematical model to establish relations among environmental data, might want to represent the system's goals in terms of mathematical equations. Meanwhile, a stakeholder with a soil science background might want to represent the system's goals in terms of soil science concepts using water permeation graphs.

The pool of available DSD will be extensible to support the creation of new notations to suit a wide range of domain specialisms (e.g. soil science, botany, hydrology, animal behaviour). In detail, when a new DSD is created by a domain specialist, that specialist will also define a mapping between the DSD and the CSD. Other domain specialists can then use this DSD without being concerned about the mapping details.

The CSD is the output of the first stage. It is a high-level representation of the combined goals that the stakeholders want the system to perform.

The CSD is used to drive the remaining stages of refinement (i.e. Network-centric and Node-centric representations) and is a format with a sufficiently rich vocabulary to capture both the abstract computational workflow of the system and the relevant scientific parameters from a particular DSD (e.g. data collection mechanics, aggregation and correlation functions). The precise form of the CSD is an open research question that we are currently investigating.

B. Network-centric Representation

The second refinement stage is the Network-centric Representation. As input this stage receives the CSD, represented as a workflow, as illustrated in Fig. 1. As output, a Network Graph is produced which represents the resources of the resulting distributed system and the connections among them. This output is generated with the help of a resource database which maintains an up-to-date list of resources such as specific IoT nodes or cloud data-centre hosts.

We envision this refinement stage being performed in two main steps. Firstly, for each element of the CSD workflow our request is made to the resource database to verify if there is a suitable resource to match that element and allocate it if so. The resource database will attempt to return a nearest match to each request: either an exact match will be returned, for example temperature sensors in the exact area that is requested; or a best-effort match, for example temperature sensors are not available in that area but the temperature can be inferred through other kinds of sensors in or near the area. In the latter case, the stakeholders will be notified of the imperfect match. Finally, if no resources can match the element then the stakeholders are notified with an error. At the end of this step we will have a set of resources reserved for use, including sources of data on the IoT and storage facilities on the cloud.

Secondly, the Network Graph Generator will consider the allocated resources R from the above step and will then allocate further (generic) resources which will serve as connections between the elements of R. These connections reflect the connections on the CSD workflow. The fine details of this step are themselves a complex research challenge which is a subject of our future work.

C. Node-centric Representation

The third and last refinement stage is Node-centric Representation. This representation consists of the fine-grained component-based architecture that can be deployed in each individual IoT and Cloud Computing resource present in the Network Graph. In detail, the component configuration of each resource comprises one generated 'main' component and a collection of pre-built library components.

Each main component describes a specialized, application-specific, behaviour of resources, ultimately derived from the high-level specification of the system. An example of a main component will be "collect temperature of a given region at a given frequency" for the IoT, "create a database to store temperature data" for the Cloud. These main components have a set of required interfaces which link them to other components that implement fine-grained behaviour, for example routing protocols, sensor drivers, aggregation functions or databases. Those components may, in turn, have their own required interfaces, and so on.

The separation of specialized (generated) behaviour from generic (library) behaviour gives us the flexibility to reuse specialized behaviours in different contexts by connecting them to alternative library components.

The component-based models that we intend to use here are Dana [8] for the Cloud Computing and LorienOS [9] for resource-constrained IoT devices. Both of these models provide advanced runtime introspection and adaptation capabilities. The main motivation behind the use of component models in this context is that it enables the encapsulation of well defined units of functionality to which we can attach appropriate semantic annotations. This annotated units are used for automating the generation of software composition to meet high-level goals.

D. Deployment and Adaptation

The component-based architecture resulting from the last refinement stage will be deployed in the IoT and Cloud Computing resources. The deployment process is achieved by sending the components one-by-one to the selected resource which will have a component-based model responsible for receiving each component and connecting it to its dependencies. The first deployed version will be functional, because the *Network Graph Generator* will always generate the best-guess configuration based on the resources available.

However, due to the fact that the resources are volatile (e.g. sensor node R runs out of battery) a generated functional configuration may be sub-optimal or may become non-functional.

In order to ensure that the system will perform in its "best" configuration, runtime adaptation is required.

We envision the system adaptation to be driven from the bottom up, i.e. from the resources to the high-level specification. That is because the resources can reason in real-time about its own infrastructure. Through successive iterations, optimal-local configurations may be achieved based on local adaptation decisions. This decentralized approach makes adaptation fast and scalable in contrast to having a centralized entity dictating how and where adaptation is performed.

VI. Challenges

In order to create a working solution for the approach described, some challenges and open issues were identified. Below is a short list of those we judge to be the most important ones:

- The *CSD representation* (see Sec. V-A) itself is a challenge. This challenge consists of defining the workflow elements that will constitute the CSD representation in order to make it: i) powerful enough to reflect the concepts described by a wide range of DSD; and ii) well structured to enable automatic checking by the Network Graph Generator to verify if the available resources can realize the system's goals.

In the Network-centric Representation (see Sec. V-B), fine details involving the Resource Database, Network Graph Generator and the Network Graph representation presents some challenges.

- The *Resource Database* is envisioned as an entity capable of reasoning over a list of resources in order to decide whether the system goals can be achieved or not. The challenge consists of finding the right techniques and algorithms that can be used to infer a match to the CSD workflow elements, in a reasonable time frame.
- The *Network Graph Generator* (see Fig. 1) consists of allocating generic resources (any resource available on the resulting distributed system) to connect selected resources (the ones necessary to realizing a system's goal) that are located separately and can not connect directly to each other. The challenges are: i) how to select the generic resources, considering the criteria and techniques to select the best set of generic resources to provide connection, and ii) how to decide the best network configuration based on a list of selected resources.

Those challenges and open issues are going to be investigated in the context of the Environmental IoT project in the near future. We encourage and invite the community to join us in investigating the listed, and related, challenges.

VII. Final Considerations

The IoT and Cloud Computing have individually great potential in a variety of application domains such as smart cities, health-care and so on. Together these technologies complement each other, increasing their potential and presenting novel opportunities to investigate emerging application domains like environmental science.

We have described an approach using three refinement stages approach to translate a high-level representation into a component-based architecture that can be deployed to a combined distributed system formed by an IoT and Cloud Computing. This approach captures scientific abstract requirements and translates them into a component-based architecture capable of providing control over the resulting distributed system. This process decreases the complexity of programming the resulting system and supports pervasive adaptation.

However, there are some challenges and open issues that need to be addressed in order to create a working solution for the presented approach. We consider: the abstract representation itself (the CSD representation Sec. V-A), the process of verifying if the resulting distributed platform is capable of realizing the system's goals (the Resource Database Fig. 1) and the generation of a network graph representation (see Sec. V-B) the most important ones.

Acknowledgements

Roberto Rodrigues Filho would like to thank his sponsor, Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil, for the scholarship grant Proc. BEX 13292/13-7.

This work has also been supported by the Engineering and Physical Sciences Research Council (EPSRC) under contract number EP/L023636/1.

References

[1] Cuzzocrea, A., Fortino, G., & Rana, O. (2013, May). Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-Of-The-Art and Future Research Directions. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (pp. 583-588). IEEE.

[2] Kurschl, W., & Beer, W. (2009, December). Combining cloud computing and wireless sensor networks. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services* (pp. 512-518). ACM.

[3] Alamri, A., Ansari, W. S., Hassan, M. M., Hossain, M. S., Alelaiwi, A., & Hossain, M. A. (2013). A survey on sensor-cloud: architecture, applications, and approaches. *International Journal of Distributed Sensor Networks*, 2013.

[4] Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., & Oliveira, A. (2011). *Smart cities and the future internet: Towards cooperation frameworks for open innovation* (pp. 431-446). Springer Berlin Heidelberg.

[5] Fouquet, F., Morin, B., Fleurey, F., Barais, O., Plouzeau, N., & Jezequel, J. M. (2012, June). A dynamic component model for cyber physical systems. In *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering* (pp. 135-144). ACM.

[6] Patel, P., Morin, B., & Chaudhary, S. (2014, May). A model-driven development framework for developing sense-compute-control applications. In *Proceedings of the 1st International Workshop on Modern Software Engineering Methods for Industrial Automation* (pp. 52-61). ACM.

[7] González García, C., Pelayo, G., Bustelo, B. C., Pascual Espada, J., & Cueva-Fernandez, G. (2014). Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios. *Computer Networks*, 64, 143-158.

[8] Porter, B. (2014, June). Runtime Modularity in Complex Structures: A Component Model for Fine Grained Runtime Adaptation. *Component-Based Software Engineering*.

[9] Porter, B., & Coulson, G. (2009, December). Lorien: a pure dynamic component-based operating system for wireless sensor networks. In *Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks* (pp. 7-12). ACM.