Synchronized Sub-Second Arbitrary Changes to Decoupled Components for Ultimate Resilience in Cross-Platform Geo-Distributed Smart Factories

Mirco Soderi
Data Science Institute
University of Galway
Galway, Ireland
mirco.soderi@universityofgalway.ie

John Gerard Breslin
Data Science Institute
University of Galway
Galway, Ireland
john.breslin@universityofgalway.ie

***Abstract*—Modern manufacturing systems characterize for the multiple dimensions of their complexity. They are numerically complex, as they consist of several components. They are logically complex, as multiple and variegated links exist among the different components. They are technologically complex, as a mix of different hardware and software technologies and architectures is typically found. They are geographically complex, as they often extend across multiple physical locations and sometimes involve multiple organizations. However, resilience to predictable and unpredictable events through timely, efficient, and effective reconfiguration of the whole manufacturing ecosystem remains a key objective, being it a key enabler of industry competitiveness. In this work, an innovative approach based on API request collections, containerization technologies, and past research about remotely reconfigurable distributed systems, is proposed for achieving ultimate resilience in modern industry.**

***Index Terms*—Resilience, Reconfigurable Manufacturing, Software-as-a-Service, Distributed System, Cross-Platform, Geo-Distributed, API Collection, Containerization Technology**

I. Introduction

Software-as-a-Service (SaaS) has been extensively discussed in the literature [1] [2] [3]. In SaaS, the software provider makes the application available over the Internet, and the customer connects to that through thin clients. The model is suitable for slowly-changing applications with loose time constraints and low-to-moderate data volumes exchanged between clients and server. Indeed, in real-time data-intensive applications, the overhead imposed by the data transfer over the Internet is not tolerable.

In reconfigurable manufacturing, multiple and possibly geo-distributed equipment and software components interact and evolve over the time to promptly adapt to the changing context, with the context being typically represented in the form of huge and variegated data streams to be processed at real-time or nearly-real-time. Because of that, traditional SaaS cannot be applied to reconfigurable manufacturing.

This publication has emanated from research supported in part by a grant from Science Foundation Ireland under Grant Number SFI/16/RC/3918 (Confirm), and also by a grant from SFI under Grant Number SFI 12/RC/2289_P2 (Insight).

However, the idea of relying on software repositories where applications can be changed producing immediate effects on all their replicas across heterogeneous and possibly geo-distributed equipment would find application in modern industry, and especially in reconfigurable manufacturing.

In this work, an approach is proposed for the implementation of such idea, which is based on: (i) containerization technologies; (ii) past research on remote building, configuration and operation of distributed systems for data engineering, analytics, visualization, and human interaction; (iii) remotely reconfigurable software modules that run API collections.

A. Background

Soderi, Kamath, Morgan, and Breslin [4], have proposed remotely reconfigurable containerized Node-RED applications, named *Service Nodes*, for ubiquitous system integration as a Service in smart factories. Also, they have proposed a containerized reconfigurable Scala + Spark server application for parallel computing and Cloud analytics, named *AIS Node* [5], which integrates into the system presented in [4] thanks to a Node-RED module, named *ai*, which can be loaded to any Service Node from a shared repository of Node-RED modules (the Transformation Factory), through an API request made to the node itself, so turning the node into a client application for the AIS Node. Soderi, Kamath, and Breslin [6] have developed and made natively available in the AIS Nodes a set of modules targeted to enable the building of API-driven infinite cyber-screens for custom real-time display of Big Data streams, relying on the same building blocks and technologies presented in [5]. A demo [7] has been given, which also includes the remote building and configuration of Web interfaces. More recently, Soderi and Breslin [8] have proposed the *Crazy Nodes*, which expose APIs for performing arbitrary and immediately effective modifications on the node implementation. A containerized Node-RED application named *Network Factory*, distributed as a Docker Image, can be used for creating, organizing, and controlling the different kinds of nodes. Considering that containers can be run from Docker Images through API requests made to

the Docker Engine, the mentioned work cumulatively enables the (re)creation, (re)configuration and operation from scratch, and from remote, via API requests, of potentially any software application, on any hardware device where a Docker engine is in operation.

B. Paper structure

In Section I the context, motivation and enabling research for this work are described. In Section II, API request collections are introduced. In Section III a design for the remotely reconfigurable software module targeted to run API request collections is proposed. In Section IV the suitability of the outlined building blocks for addressing the stated problem is discussed. In Section V, conclusions are drawn.

II. API request collections

An API request collection is a sorted sequence of API requests. Each API request is described by its (i) HTTP method, (ii) URL, (iii) body, (iv) headers, (v) preconditions; (vi) verification logic. Requests can be organized in folders. In all of the mentioned parts, variables can be used. Variables can be set by the consumer when starting the collection execution, or they can be automatically set during the execution as a part of the verification logic. For example, during the verification of the response received for a given request, a variable can be set to the value of the response and used for the next request.

For running an API collection, it is necessary to rely on a software tool that interprets the textual serialization of the API collection, uses the user input for actualizing variables, and then for each request verifies the preconditions, then sends the request if it is the case, then executes the verification. As requests can be organized in folders, the tool should also accept as optional input the name of the folder to be executed. Postman and Newman are well-known tools for API collection development, testing, serialization, and execution.

III. Reconfigurable API collection execution

The reconfigurable application responsible for the execution of the API request collection should include three modules: (i) I/O configuration; (ii) execution configuration; (iii) business logic. It should act as a wrapper for the API collection execution engine. The *I/O configuration module* should expose APIs for setting the data source and destinations. From the data source, values come that are to be used for setting the variables used in the collection. Multiple destinations are advisable for differentiating the standard from error outputs. The *execution configuration module* should expose APIs to be used for setting parameters that are specific to the API collection execution engine, including the URL where the collection to be executed locates. The *business logic module*, for each input coming from the data source: (i) retrieves the most up-to-date version of the collection, relying on mechanisms such as the ETag for avoiding unnecessary network traffic and reducing the time of response; (ii) runs the execution engine with the configured engine-specific parameters, setting execution variables according to the input received from the data source.

IV. Discussion

If the application for reconfigurable API collection execution is distributed as a Docker Image or as a Docker Volume backup meant to be mapped to the /data path of a Node-RED container, a container for such application can be created and run through requests made to a Network Factory or directly to a Docker Engine. That means that one or more remote API collections can be run on any equipment where a Docker Engine is in operation. According to what described in Section III, if the remote collection is updated, all devices will download and use the new version when the next input will come from their data sources, without the need of any third party or usage tracking mechanism. The remote collection can include requests made to the local host, for example for building and configuring a software application from scratch on each of the devices where the collection is configured for usage, according to what described in Subsection I-A. The collection can also include calls to external services, in which case the implementation of those services can be changed at any moment with immediate effect even without generating a new version of the collection. Thanks to containerization technologies, the approach is applicable to the most internally variegated manufacturing systems. Being based on API requests, the approach is applicable across multiple physical locations relying on VPNs, or the Internet.

V. Conclusions

In this work, an approach has been proposed for introducing a variant of Software-as-a-Service in reconfigurable manufacturing, so enabling software changeability also in real-time data-intensive internally variegated and often geographically sparse systems such as those in use in modern industry.

References

[1] Dubey et al., "Delivering software as a service," The McKinsey Quarterly, vol. 6, no. 2007, p. 2007, 2007.

[2] Tsai et al., "Software-as-a-service (saas): perspectives and challenges," Science China Information Sciences, vol. 57, pp. 1–15, 2014.

[3] Ma, "The business model of software-as-a-service," in Ieee international conference on services computing (scc 2007). IEEE, 2007, pp. 701–702.

[4] Soderi et al., "Ubiquitous system integration as a service in smart factories," in 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS). IEEE, 2021, pp. 261–267.

[5] Soderi et al., "Advanced analytics as a service in smart factories," in 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI). IEEE, 2022, pp. 000 425–000 430.

[6] Soderi et al., "Toward an api-driven infinite cyber-screen for custom real-time display of big data streams," in 2022 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, 2022, pp. 153–155.

[7] Soderi et al., "A demo of a software platform for ubiquitous big data engineering, visualization, and analytics, via reconfigurable micro-services, in smart factories," in 2022 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, 2022, pp. 1–3.

[8] Soderi et al., "Crazy nodes: towards ultimate flexibility in ubiquitous big data stream engineering, visualisation, and analytics, in smart factories," in Leveraging Applications of Formal Methods, Verification and Validation. Practice: 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22–30, 2022, Proceedings, Part IV. Springer, 2022, pp. 235–240.