Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 104 (2021) 1506–1511

www.elsevier.com/locate/procedia

54th CIRP Conference on Manufacturing Systems

A Fractal Control System Architecture for Next Generation Factories

Maximilian Raphael Visotschnig[a,*], Jürgen Henke[a], Dominik Lucke[a,b]

[a]*Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Nobelstrasse 12, 70569 Stuttgart, Germany*
[b]*Hochschule Reutlingen, ESB Business School, Alteburgstraße 150, 72762 Reutlingen, Germany*

* Corresponding author. Tel.: +49-178-1425224. *E-mail address:* maximilian@visotschnig.net

Abstract

This paper presents the concept of the system architecture of a flexible cyber-physical factory control system. The system allows the automation of process structures using cyber-physical fractal nodes. These nodes have a functional and independent form and can be clustered to larger structures. This makes it possible to equip the factory with a flexible, freely scalable, modular system. The description of this system architecture and the associated rules and conditions is outlined in the concept.

Keywords: Cyber-Physical Systems; System Architecture; Control System

1. Introduction

Factories today have to act in highly turbulent markets, often demanding more and more personalized products and shorter life cycles, resulting in rising number of product variants and complexity to manage. As a response on this market developments, technical processes are optimized by introducing new machines and equipment or by upgrading legacy machines. The ladder gains more and more of importance as often a replacement is not possible due to economic or regulatory reasons in the case a product and the technical process are certified. Therefore, companies have to adapt permanently more and more rapidly to changing situations in order to stay competitive and need to develop skills of flexibility and transformability. Automation, information and communication technologies can enable and improve these skills. [1] With the introduction of programmable logic controls (PLC), computerized numerical control (CNC) machines and flexible manufacturing cells during the last decades already huge improvements in productivity and flexibility have been achieved. During the last 20 years developments such as OPC UA or IEC TS 62832-1 (Industrial-process measurement, control and automation - Digital factory framework) or asset administration shell (AAS) [2] simplify the development, adaptation, and maintenance of these control systems. But still these tasks are time consuming and cause high efforts, as these have to be done often manually within each separate programming environments or even physically. On the higher optimization level, big data and artificial intelligence approaches enable further improvements. However, these are predominantly implemented in separate systems as additional functions. The current developments are going towards so-called edge architectures, processing already the acquired information locally to reduce the amount of information sent to a server located in the cloud. Here, such edge devices use IP-based communication protocols and web services as technical standards, but still the software components on these have to be configured and deployed manually. Therefore, the next evolutionary step in the optimization of the processes is to support and simplify the development, adaption, and

10.1016/j.procir.2021.11.254

maintenance by further assistance up to a self-configuration and optimization of the machines and equipment. Therefore, new approaches and architectures in hardware and software are required providing the bedrock of next generation of machine and technical process control. This paper presents a concept, addressing the challenges and allows a variable, scalable factory control system. As a conclusion of the paper, a possible system architecture is used to show how the concept can be implemented.

2. Related work

The basic functions required for a process control system result from the interaction between the automated process and the automation device. On the one hand, this involves setting process parameters and, on the other, measuring or monitoring the process. These functions are made possible by actuators and sensors. Control is another function in which the measured and observed information is processed to determine new manipulated variables. [3]

Cyber-Physical Systems (CPS) are objects that connect the physical and digital worlds. They are able to interact and communicate with both worlds. There are different forms of cyber-physical systems. Sensors and actuators are used for physical interactions, network devices for connection to the digital world. Cyber-physical Systems can have many forms, starting from embedded systems up to complete buildings or plants. It should be added that cyber-physical systems can be aggregated. For example, a machine can be composed of different cyber-physical systems. [4]

To process data from cyber-physical systems and the Internet of Things, cloud computing capabilities are used. [5] Cloud computing is composed of dynamically provisioned IT resources and virtualization of software. [6]

The hardware required to operate the CPS is now typically implemented as a virtual system, i.e. container virtualization. Here, no complete virtual machine is created, but a runtime environment separate from the host operating system. For this reason, containers use the same kernel as the host, but otherwise run their applications separately from the actual operating system. Containers are operating on an image that runs an operating system and exclusively the software necessary for their function. They can be quickly implemented and used. Furthermore, they offer good portability as long as the restrictions imposed by the kernels are respected. Examples of container virtualization include Docker and CRI-O. [7]

The operation of such containers in a larger and networked context can be realized by orchestration and scheduling. For this, there are solutions such as Kubernetes or Docker Swarm, which manage a cluster of devices and containers running on them. There is already research done regarding the usage of container architectures in distributed control applications [8] and the orchestration of containerized microservices for the IIoT [9].

Cloud computing very much assumes that information is collected at the end device and processed in the cloud. More advanced or related approaches are edge and fog computing. These increasingly shift the processing of data to the end device. This ensures that information are already processed at the point of origin and, if necessary, only relevant information is transferred further to the cloud. [10].

Current research regarding the integration of machine or plant controls into cloud-based environments focuses on the existing real-time requirements. [11]

Transfer of basic mathematical models to production structures three decades ago led to the now well accepted concepts to structure production. The abstraction of the mathematical model defined in [12] for assembly processes, i.e. the search for the smallest production unit capable of self-organization led to genuine acceptance of the team concept in production planning [13].

In [14] is shown, that CPS can interconnect and build networks autonomously and decentralized – in other words, entirely in the spirit of these self-similar production fractals – and optimize themselves independently.

The self-description of resources is kept simple in even advanced concepts, as in [15]. As noted there it needs to be generalized and should be based on a standardized format, e.g. use the asset administration shell. The necessity for a complete self-description and an according directory service to allow for adaption to the manufacturing situation dynamic updates to the configuration shows the need for a new architecture. This architecture must allow the dynamic definition of functions and data streams at any time and still be tamper-proof during operation. For this purpose, the device status has to be continuously monitored. All presented research concentrates either on data aspects or on the deployment of such systems. A general architecture for a distributed self-organizing scalable process – or better – plant control system is missing.

3. Concept

The key idea is to introduce the model of a so-called node which has in simplest case one defined function and can be combined with other nodes to a complex system. We name this fractal node system (FRANS). A single FRANS node can be seen as a black box with standardized communication

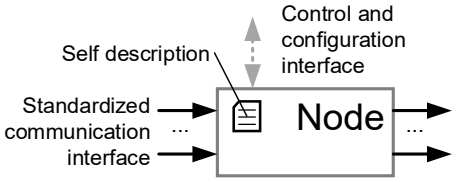

Fig. 1. Basic model of a FRANS node

channels for input and output data streams, as well an interface for control and configuration (Fig. 1). Furthermore, a FRANS node has a self-description comprising a unique-id, name, description, category, status messages such as active, inactive, idle, error and the assigned execution device such as a microcontroller, an industrial PC or a cloud environment. Also, the self-description enables an automatic configuration and orchestration. FRANS nodes are following the software-defined principle where economic and are executed on a virtualized hardware and use containerized

software virtualization. In order to facilitate the implementation of a node-based system, some nodes types and IO streams are predefined as standard. The FRANS nodes can be combined arbitrarily via their input and output channels. In Table 1 the basic types of FRANS nodes are shown, based on the functions of an automation system.

Table 1. Basic FRANS node categories

	Meaning
Data Acquisition (Measurement)	These nodes are dedicated to acquiring data from various sources. Data sources can be e.g., physical sensors but also digital data sources such as internet data sources or a data from an OPC UA server.
Action	Function of these nodes is to execute a specified action based on the incoming data. An action can reach from a physical one e.g., to switch a motor on or off, to digital ones e.g., to send an email.
Pre-Processing and Analysis	This type of nodes cluster all functions for data pre-processing and analysis of acquired data. Examples for functions are basic sensor data processing, feature detection or anomaly detection also with machine learning algorithms, but also the data format transformation.
Optimization	This type of nodes cluster all functions for control and optimization. Examples for functions are simple rule-based decision or complex simulation models.
Data Management	This type of nodes cluster all functions for data storage and management. Examples for functions are data bases or file server.
Visualization	These nodes are used to visualize data.
Control and Surveillance	These nodes are responsible for administration and control of the assigned nodes. This includes also a self-optimization of this system according to objectives such a quality-of-service level.

Several FRANS nodes are assigned to a so-called control and surveillance node (CS-node), which configures and monitors the execution of assigned FRANS nodes. Also, the CS-nodes are responsible to reach operational goals such as quality of service level, within their surveillance horizon. Therefore, CS-nodes comprise functions for system reconfiguration or maintenance operations. An example is an automatic reconfiguration of the system due to a failure of a sensor. It detects that one of the data acquisition nodes is down. As solution it can check if there are other data acquisition nodes delivering the equivalent data are available. To achieve this, it uses the self-description of the FRANS nodes which are stored in a repository. With this information it can evaluate a possible alternative FRANS node and to recover the functionality within its surveillance horizon. Furthermore, the combination and nesting of FRANS nodes allows to create FRANS nodes with more than one function, which can be aggregated to new FRANS nodes. In this case the aggregated nodes are assigned to more than one category. Linked nodes can be aggregated into a more complex node, which builds a black box around these child nodes. It allows that only the relevant inputs and outputs have to be set for this node.

Fig. 2 shows the example of configurable fan control and temperature monitoring in a machine. It can be used e.g., for the upgrade of a machine with an optimized ventilation system using existing temperature sensors. The function of the system is to control the speed of the fan, based on the measured temperatures by an external existing sensor and the target temperature getting from a database. Moreover, the average measured temperatures are sent to a data management node, which feeds a visualization node. The FRANS nodes directly used for the data acquisition and processing, analyzing the temperature values, and optimizing it to a target fan speed e.g., in the simplest case via a characteristic curve. This aggregated node has an own CS-node and in this case this CS-node contains the self-description of the aggregated node, so that it can be easily

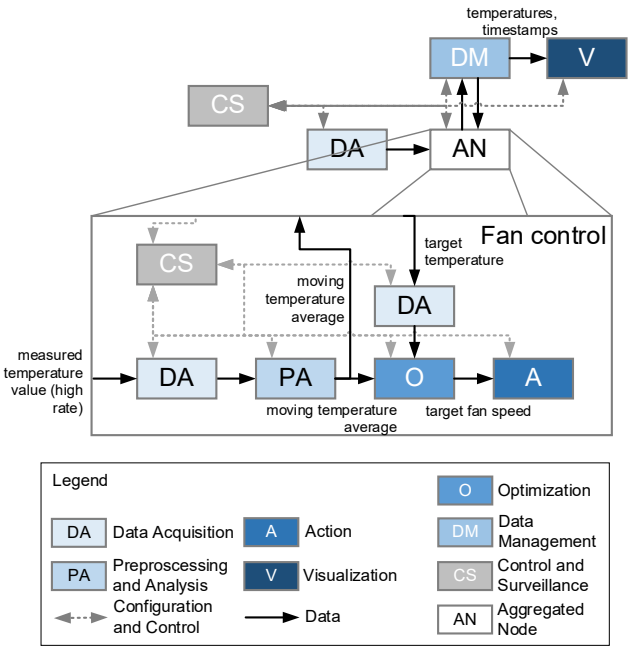

Fig. 2. Example temperature-controlled fan

integrated. So, a system of FRANS nodes can range from a smart object such a sub system of a machine, up to a highly distributed system on factory or even value adding network level (Fig. 3).

In principle, one FRANS system can comprise an unlimited number of FRANS nodes. However, in reality the number of FRANS nodes in one FRANS system is limited mainly by the performance of the underlying communications system and computing resources hosting and executing the FRANS nodes. It results in high latency times in a system with a high number of FRANS nodes. To address this problem, the quality-of-service (QoS) containing the latency is measured by the CS-nodes. Then

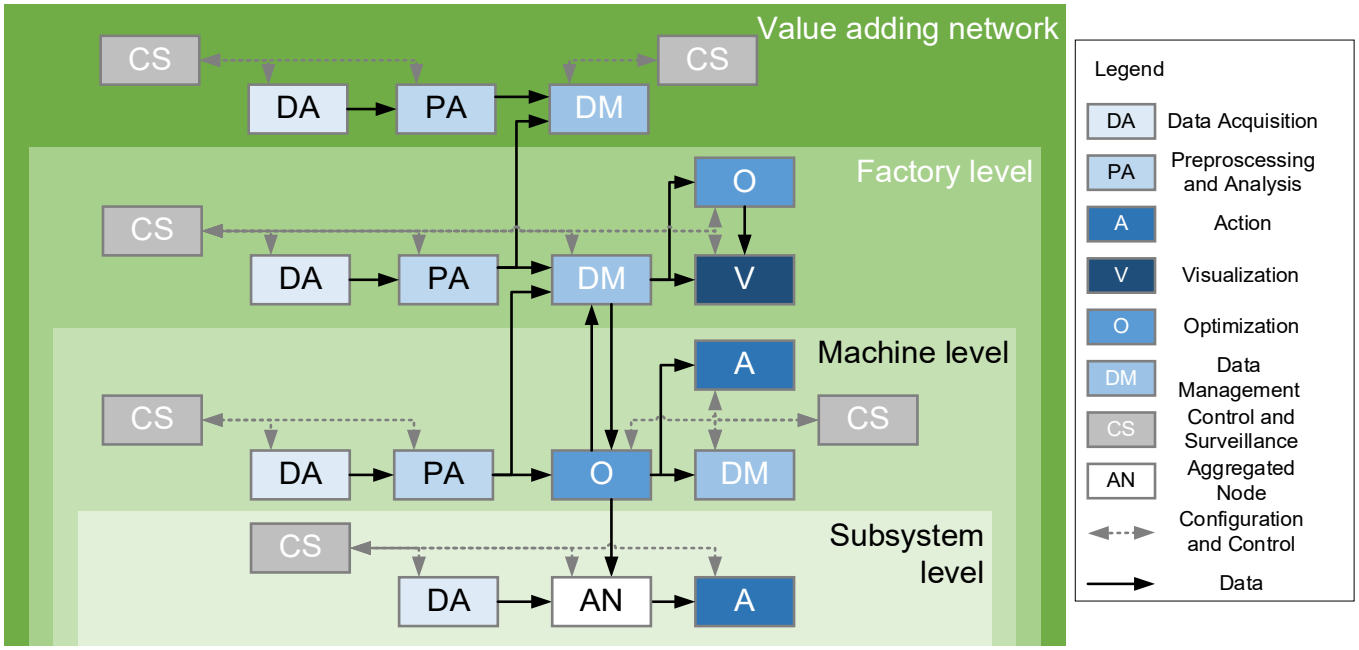

Fig. 3. FRANS Architecture

several strategies can be applied. Firstly, one solution is to restrict the number of FRANS nodes monitored by a CS-node, depending on the communication network and computing resources used. Secondly, as the CS-nodes are designed with built-in self-optimization capabilities, an automatic reconfiguration based on a QoS level objective is enabled. Examples for automatic reconfiguration strategies can be the relocation and redeployment on a higher performant resource or the automatic division of the system and assignment of parts to another or new CS-node.

Also, the FRANS nodes allow an accelerated and simplified manual system development by reusing of existing FRANS nodes. The single FRANS nodes can be developed in parallel, and each validated in white and black box tests in advance. This enables to start with a part of a single machine sub system, such as a temperature control, that is validated continuously, also against the conventional system. Then the FRANS system can be extended step-by-step up to the whole plant in a controlled way.

4. Architecture

For the design of FRANS nodes, the basic assumptions presented in the concept are implemented. For this, the focus is strongly placed on using a self-contained and transportable software solution for the realization of the functions. For this purpose, container virtualization like Docker is used. The Open Container Initiative (OCI) standard provides a standard that can be used for developing FRANS nodes as container applications.

The FRANS nodes are implemented as container image which can be deployed as container instance on devices with a preinstalled runtime environment like Docker Engine, CRI-O, or container. This allows to replicate FRANS nodes for different issues by creating new container instances based on the same image. In addition, container images offer a high degree of flexibility for the development of new FRANS nodes. They can be easily and quickly modified or extended by using existing images as a basis. This is possible due to their layered build process. The information for the self-description and configuration of a FRANS node is handed over by environment variables and other parameters. These environment variables are for example used to define the IOs of a FRANS node. Also, information like mounting points for storage media or network configuration are passed over as part of the deployment. The FRANS DA-nodes will – for certain, typically digitally interfaced sensor types, like 1-WIRE™ and I2C – recognize the attached sensors on their own and report the current configuration to the CS-node via the implemented REST interface. In turn, the CS-node can change the specific configuration of the sensor node via the REST interface. All node types can be rolled out via the registry and revised if necessary - this includes, for example, newly implemented functions. Beside these container FRANS nodes, it is planned to implement also simple nodes like a data acquisition of a temperature sensor directly with embedded systems.

These devices are initialized with the same amount of self-description like a container. The software is delivered via over-the-air upgrade upgrades and will also be managed by the CS-nodes. The aggregation of different FRANS nodes is realized as part of the orchestration. In this case it isn't needed that the aggregated FRANS node must be built as new larger image. Instead, there are solutions which can combine different container to a larger application. Orchestration tools like Docker Compose, Docker Swarm or Kubernetes use configuration files to define which containers should be started and which parameters should be handed over for every container. In the current system architecture Kubernetes is intended to do this orchestration. In Kubernetes such aggregated nodes will be implemented as so-called pod. How the aggregated node is composed is

described in a pod manifest file. In Fig. 4, an example for the platform docker compose in YAML is shown. This file contains all relevant information to describe the node and to refer on which kind of device it should be deployed.

Kubernetes will check this file and schedule on which device the nodes will be deployed. Because Kubernetes services are already container based, they will be aggregated into the surveillance nodes, which are distributed over the system. The so-called pod configuration for Kubernetes will be automatically created also as part of these nodes and can be influenced by the system operator as well as by CS-nodes.

The orchestration of FRANS nodes is currently planned to be implemented by publish-subscribe-approach. An important advantage of this approach is that nodes can be freely swapped in this system. For example, a new sensor can send its measurements on a data channel while the actuator on the other side of the channel requires no reconfiguration.

Also, a FRANS node can be shut down or be restarted without directly affecting the other FRANS nodes. A basis for secure communication is the common semantics of the participants. As part of the FRANS nodes' self-description, it is also necessary that their incoming and outgoing information is uniquely determined. For this purpose, it is intended that nodes serialize outgoing data uniformly according to defined message schemes. A recipient FRANS node can then deserialize the data and validate it directly

```
frans-sensor-w1:
    image:
"${REGISTRY_HOST}frans/frans_registry/frans/frans.sensor.w1:
${FRANS_VERSION}"
    logging:
      driver: "json-file"
      options:
        max-size: "${LOGGING_MAX_SIZE}"
        max-file: "${LOGGING_MAX_FILES}"
    env_file:
      - env.list
    networks:
      default:
        aliases:
          - frans.sensor.w1
    ports:
      - "8112:8112"
      - "8113:8113"
    depends_on:
      - frans
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f
http://localhost:8112/sensor/health || exit 1"]
      interval: 1m30s
      timeout: 10s
      retries: 3
      start_period: 40s
```

Fig. 4. Part of a FRANS docker compose description file

against a stored schema. According to this approach Fig. 5 shows an example of a temperature message and its scheme in JSON syntax.

The scheme contains therefore mandatory and optional properties. The message includes its format, the value, and its measuring unit. Further meta or context data such a position can also be included. In this case the position of the sensor would be an option to better handle the data and to use it for further optimization of the system. The data channels are self-organizing. The FRANS nodes decide which information is required to realize a function and check whether required information is already provided as a data channel or whether such a channel must be created. FRANS nodes use for this purpose a registry service. This registry service gathers information of provided FRANS nodes and communication channels and provides them to other FRANS nodes.

5. Conclusions and outlook

```
{
 "type": "object",
 "properties":
      {
  "msgformat": {"type": "string", "enum": ["temperature"]},
  "value": {"type": "number"},
  "unit": {"type": "string", "enum": ["c", "k", "f"]},
  "position": {"type": "string"}
      },
 "required": ["msgformat", "value", "unit"]
}

{
 "msgformat": "temperature",
 "value": 24.3,
 "unit": "c",
 "position": "zone_a"
}
```

Fig. 5. Example of a temperature schema and message

In this paper an ICT system architecture is presented enabling to simplify the development, adaptation, and maintenance of process control systems in a broader sense within a factory. For this purpose, based on the key idea of fractal node the characteristics of a system are presented. Using the specific features of the FRANS nodes, the software-defined principle executed on a virtualized hardware and their self-description are the basis for flexible and resilient system. To achieve the ladder, the concept of control and surveillance nodes (CS-nodes) is introduced, which are responsible to reach operational goals such as quality of service level, within their surveillance horizon. Therefore, they include functions for the automatic system reconfiguration or maintenance operations of their assigned FRANS nodes. This concept also enables a self-optimization and certain resilience of the whole system against failure of single FRANS nodes and, if necessary, the re-segmentation of the FRANS network. Further research activities focus on the implementation architecture in industrial use cases to determine further aspects and optimize and evaluate the different elements. The described architecture also creates the opportunity to use aspects of machine learning to achieve the next level of resilience.

References

[1] Bauernhansl T. Die Vierte Industrielle Revolution: Der Weg in ein wertschaffendes Produktionsparadigma. In: Vogel-Heuser B, Bauernhansl T, ten Hompel M, editors. Handbuch Industrie 4.0 Bd.4: Allgemeine Grundlagen, 2nd ed. Berlin: Springer Vieweg; 2017. 1-31.

[2] Plattform Industrie 4.0. Details of the Asset Administration Shell Part 1: The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01).
[3] Lunze J. Automatisierungstechnik. 4th ed. Berlin, Boston: De Gruyter; 2016.
[4] Vogel-Heuser B. Herausforderungen und Anforderungen aus Sicht der IT und der Automatisierungstechnik. In: Vogel-Heuser B, Bauernhansl T, ten Hompel M, editors. Handbuch Industrie 4.0 Bd.4: Allgemeine Grundlagen, 2nd ed. Berlin: Springer Vieweg; 2017. 33-44.
[5] Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge Computing: Vision and Challenges. IEEE Internet Things J; 2016;3(5):637–46.
[6] Meinel C, Willems C, Roschke S, Schnjakin M. Virtualisierung und Cloud Computing: Konzepte, Technologiestudie, Marktübersicht. Potsdam: Univ.-Verl. Potsdam; 2011.
[7] Liebel O. Skalierbare Container-Infrastrukturen: Das Handbuch für Administratoren. 2nd ed. Bonn: Rheinwerk; 2019.
[8] Grüner S, Malakuti S, Schmitt J, Terzimehic T, Wenger M, Elfaham H. Alternatives for Flexible Deployment Architectures in Industrial Automation Systems. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA); 2018. p. 35–42.
[9] Rufino J, Alam M, Ferreira J, Rehman A, Tsang KF. Orchestration of containerized microservices for IIoT using Docker. 2017 IEEE International Conference on Industrial Technology (ICIT); 2017. p. 1532–1536.
[10] Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F, Niakanlahiji A, Kong J, Jue JP. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture; 2019;98:289–330.
[11] Tasci T, Melcher J, Verl A. A Container-based Architecture for Real-Time Control Applications. 2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC); 2018:1–9.
[12] Mandelbrot BB. The Fractal Geometry of Nature. New York: Henry Holt and Company; 1983.
[13] Warnecke H-J. Fractal factory - virtual production. Innovative Production 2000 Yearbook for Automation and Modernization. Berlin: Trade & Contact; 2000. p. 1–2.
[14] Bauernhansl T. From the fractal factory to the cyber-physical production system. Cyber-Physical Systems: Uplifting Europe‘s Innovation Capacity 29th-30th October 2013, Brussels, Belgium; 2013.
[15] Schneider M, Breunig DA, Götz B, Khalil AK. Cloud-managed Service Deployment for Manual Assembly Workstations. 25th IEEE International Conference on Emerging Technologies and Factory Automation September 08-11, 2020. Piscataway, NJ, USA; 2020. p. 53–60.