# Integrating Multi-Disciplinary Offline and Online Engineering in Industrial Cyber-Physical Systems through DevOps

Iori Mizutani
iori.mizutani@unisg.ch
University of St.Gallen
St.Gallen, Switzerland

Ganesh Ramanathan
ganesh.ramanathan@siemens.com
Siemens AG
Zug, Switzerland

Simon Mayer
simon.mayer@unisg.ch
University of St.Gallen
St.Gallen, Switzerland

## ABSTRACT

Industry 4.0 is transforming industrial automation systems into increasingly complex cyber-physical systems (CPS). In particular, there is a need to integrate systems across horizontal and vertical layers across industrial and disciplinary domains. The development of industrial CPS requires not only high-level configuration and control at the execution system level, but also offline and online engineering (OOE) work, including mechanical and electrical engineering of the devices, their installation, networking, and documentation. Using DevOps, a collection of modern software lifecycle management tactics, we propose to build an integrated platform to manage the OOE tasks. Continuous Integration/Continuous Delivery (CI/CD) pipelines and Infrastructure as Code, which are key aspects of DevOps, coordinate the building, testing, and deployment of software services. To bring the corresponding benefits of pure software systems to industrial CPS, we integrate engineering tasks using distributed version control and the W3C Web of Things. As a proof of concept, we implemented the concept for a mock system that includes the OOE tasks in a process automation scenario. The DevOps platform provides a structured transfer and exchange of engineering knowledge among stakeholders involved in industrial CPS.

## CCS CONCEPTS

• **Software and its engineering** → *Software development process management*; • **Information systems** → Information integration; *Web applications*; • **Human-centered computing** → **Collaborative content creation**.

## KEYWORDS

DevOps, Cyber-Physical Systems, Industrial Automation, Web of Things, CI/CD

**ACM Reference Format:**
Iori Mizutani, Ganesh Ramanathan, and Simon Mayer. 2021. Integrating Multi-Disciplinary Offline and Online Engineering in Industrial Cyber-Physical Systems through DevOps. In *11th International Conference on the Internet of Things (IoT '21), November 8–12, 2021, St.Gallen, Switzerland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3494322.3494328

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*IoT '21, November 8–12, 2021, St.Gallen, Switzerland*
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8566-4/21/11…$15.00
https://doi.org/10.1145/3494322.3494328

## 1 INTRODUCTION

The engineering of industrial automation systems employs a distinct service-oriented architecture [37] to manage modular, interoperable, and reusable components in the control of complex cyber-physical systems (CPS) by separating concerns. *Offline and online engineering* (OOE) [14] accounts for a substantial fraction of the workflow in the development of industrial CPS. In particular, manual tasks such as construction design, installation, wiring, and verification of kinematic behavior require extensive human intervention (Figure 1) dealing with semantically overlapping engineering models [20]. The recent organizational efforts in a common industrial framework, such as the *OPC Unified Architecture* (OPC UA) standards, are being made to guarantee the interoperability between their operational units.However, the obstacles to integrating data from different engineering disciplines are the isolated and heterogeneous vocabularies developed in individual engineering disciplines and the use of engineering tools with diverging data models and representation formats [43]. For these reasons, OOE tasks remain highly empirical and time-consuming for engineers, as they manually identify, collect, and verify all the necessary information.

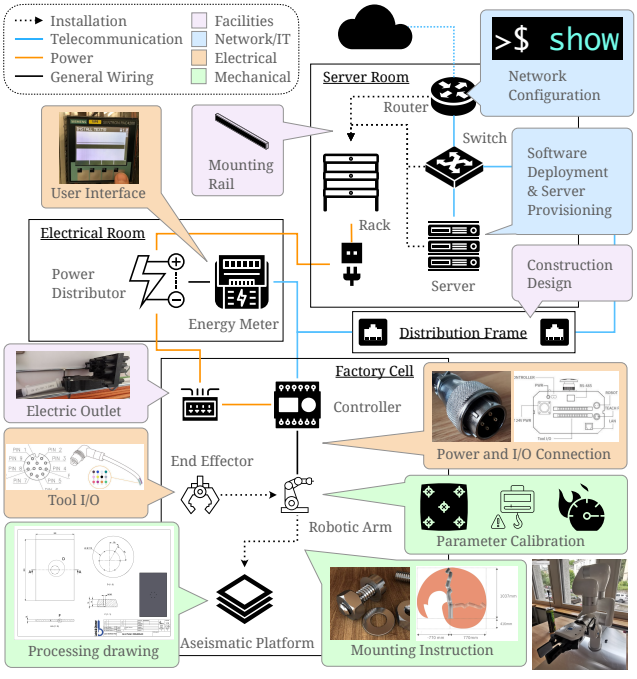


**Figure 1: Cross-disciplinary offline and online engineering.**

Various stakeholders are involved in the design and operation of industrial CPS as seen in Figure 3. As envisioned in the *Charter for Work and Learning in Industry 4.0* [1], safe, fair and self-determined work for humans should be focused on the three central pillars of *people*, *organisation*, and *technology*, given the ongoing digital transformation. Despite the technological innovation in the context of Industry 4.0, the evident importance of human aspects integration in operations processes has been underinvestigated in research to date [38]. This lack of human factors aspects in research induces pitfalls that lead to innovation without due attention to Human-in-the-Loop (HITL) operations. Optimization of HITL operations, especially regarding OOE, is inevitable to evolve an industrial automation system into truely interconnected CPS. A key challenge for realizing integrated coordination and management of engineering tasks is thus collaboration between humans with different roles and disciplines [8, 38].

Meanwhile, the concept of DevOps was created to counter the similar problems of HITL management in software development. We therefore propose to adopt DevOps for multi-disciplinary human roles integration in industrial CPS to streamline the OOE process. We leverage existing automation tools across the industrial automation and DevOps domains by describing the industrial CPS infrastructure in machine-readable formats. This establishes similar pipelines for different organizational and technical roles of humans and their heterogeneous tasks they must undertake. In this paper, we demonstrate our concept by building an DevOps platform that coordinates OOE tasks at the field level and ensures consistency over the task lifecycle. Concretely, we apply the proposed method to conceptualize, install, configure, and commission a mock manufacturing line. This example allows us to evaluate the OOE process integration from a practitioner's perspective.

In the rest of the paper, we first introduce the core concepts in our approach with related work in Section 2. In Section 3, we elaborate the proposed OOE integration through DevOps. In Section 4, we present key ingredients of the concept specific to the disciplinary domains. In Section 5, we explain our implementation in the mock system and verify the concept. In Section 6, we discuss the constraints and the challenges. Finally, Section 7 concludes the paper and provides an outlook.

## 2 BACKGROUND AND RELATED WORK

Industrial automation systems are becoming highly complex due to the need of cross-domain integration, such as building automation, automatic identification and data capture, and process automation as shown in Figure 2. Coordination of OOE tasks is particularly important in managing such complex infrastructure of industrial CPS and the workflow of the development process. In this section, we present the three core concepts in our work and their related work.

### 2.1 Offline and Online Engineering

Historically, the term "*offline engineering*" has been used to refer to engineering work carried out by engineers whereas there are no actual devices [47] (i.e., conceptualization and design), and "*online engineering*" for engineering work to bring devices *online* by installing and commissioning them [40]. As the Open Process Automation Forum (OPAF) has formulated in the O-PAS standards the requirements to streamline field device commissioning processes, OOE requires the most extensive human efforts during the development.For example, when determining the physical location of devices and their wiring (power, data, and control cables), the surrounding conditions of the device (e.g., obstacles and the relative position to other devices) as well as its functional requirements (e.g., kinematic behaviour) must be considered to avoid interference with other equipment and people around.

IEC 62337 defines that mechanical and electrical checks precede the commissioning of field devices in an industrial automation system.However, depending on the components to be (re-)installed, (re-)placed, or (re-)configured in the workflow, the development process may cause changes that propagate to other system components of any disciplinary domain ("ripple effects"). As a concrete example, when the reorganization of an assembly line involves moving a robot arm from one place to another, this requires (*i*.) manual tasks that are executed to dismount/remount the robot physically and arrange its wiring as well as documenting the setup, (*ii*.) verifying the required functionality of the robot at the new installation, (*iii*.) establishing the communication channel between the robot and its controller, (*iv*.) modification and testing of the motion control logic of the robot to adapt it to its new task in the new environment, and (*v*.) modification of the automation flow to re-integrate the robot in the execution system. This OOE process of the robot requires coordination between (*i*.-*v*.) with constraints stating under what conditions they should be executed. The reconfiguration of the motion control logic will require to know the way the robot has been mounted, and vice-versa; the remounting task needs to know about the constraints of the motion controller; and if modifications are made during any of these processes, all other affected operations need to be re-verified.

An interdependency between system components materializes during an online engineering process to interface the system components physically or logically. Hence, the dependencies often span across different disciplinary domains and do not necessarily follow a linear pattern; rather, they induce the necessity of cyclic coordination of the corresponding tasks to fulfill the (final) requirements. Furthermore, the heterogeneity in the system architecture

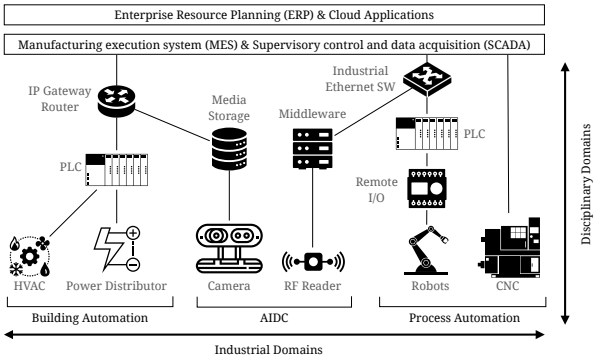


**Figure 2: Horizontal and vertical integration in industrial CPS.**

appends another dimension of complexity to the interdependencies. For example in the deployment of software components, while a loosely-coupled and hybrid interface structure allows the dynamic deployment of the logic and configurations, a tightly-coupled and on-device interface structure requires a time-consuming process to reconfigure the control system and the mechanical subsystem [34].

The identification and resolution of these interdependencies necessitate collaboration among the engineers; however, there is no structured and systematic way of achieving this. As of today, resolutions of these dependencies are done through discussions via phone calls, exchanging (e)mails, chat messages, issue tracker systems, or as part of in-person meetings [35, 42, 46]. Hence, the management of task executions and dependency resolution is an inevitable element during the development process of industrial CPS. To this end, we aim to establish a mechanism that allows for structured knowledge exchange of the OOE process in order to identify and seek solutions to dependencies between system components.

## 2.2 Multi-Disciplinary Workflow Management

In general, development processes of industrial CPS are carried out by an organization of people with different roles (e.g., owners, domain experts, and technicians) and responsibilities (e.g., facilities, electrical, mechanical, network/IT, and software engineering). Such heterogeneity both in the organizational and disciplinary domains requires different domain experts and technicians to collaborate with each other [8]. These stakeholders participate in different development phases as shown in Figure 3.

**Table 1: Responsibility in engineering disciplines**

| Discipline | Responsibility |
|---|---|
| Facilities | Design, construction, and maintenance of the facility (building) consisting of Building Automation components and floor plans. |
| Electrical | Installation of electrical system in the facility conforming the safety regulations and functional requirements. |
| Mechanical | Installation, configuration, calibration, and validation of mechanical field equipment such as robotic arms, CNC machines, belt conveyors, etc. |
| Network & IT (Software) | Installation of network appliances for the industrial fieldbus and/or TCP/IP network for the interlink in the shopfloor, and the management of the network and the IT systems (e.g., MES, SCADA, ERP, etc.). |

Although engineers from different disciplines are responsible for their own parts of the project (Table 1), it is important to recognize that the boundaries of responsibility (and the limits of authority) are blurred because of the complex intertwining and interdependencies of components, tasks, and processes in the system. As a result, the engineers are required to be multi-disciplinary, but the lack of domain knowledge may lead to a poorly documented system [39]. Such a system lacks transparency in the provenance of the components [36] and makes it difficult to identify the root cause of incidents on the shopfloor [26]. While the HITL task optimization in the post-development phases of industrial CPS are being studied by researchers (e.g., automation-assisted inspection process [2]), the operations management for OOE process are underinvestigated.

## 2.3 DevOps

Workflow management in industry is usually coordinated using an (on-premise) issue tracking system provided by the information system that oversees the project (i.e., Enterprise Resource Planning; ERP). In software development projects, issue reporting is an important artifact to enable social interaction among developers [6] and to build a common knowledge-base for problem resolution [10], as there are a plethora of proprietary and open-source software (OSS) issue tracker products implementing different principles and features [19]. The important thing here is that an issue is closely tied to a particular version of the source code, and likewise, an OOE task is (only) required at a specific state of the shopfloor. For this reason, distributed version control (DVC), such as Git and Mercurial, is widely adopted by OSS projects leveraging the cohesive and isolated development with *branches* and *merges* [5], which can be further integrated with the issue management. In this context, various code hosting platforms (e.g., GitHub) are facilitating pull-based development [23] with workflow support tools including code review systems and integrated issue tracking systems.

Furthermore, the concept of a *Continuous Integration/Continuous Delivery* (CI/CD) pipeline has been widely recognized to promote collaborations across (multiple) software developers and service operators (i.e., "Dev" and "Ops") [22] at the building, testing, deployment, and monitoring stages in the lifecycle of modern software services. A CI/CD pipeline continuously executes integration of distributed development processes using DVC as well as automated testing and verification by CI tools [15] and deliver the (software) artifact to its operational environment [16] (e.g., cloud infrastructure). Here, another concept called *Infrastructure as code* (IaC) has emerged to automate iterative provisioning, deployment, and management of computing infrastructure across physical, virtual, and cloud environments [3], which are the integral part for continuous delivery of software applications.

*NetDevOps* is a newly evolving concept that adopts DevOps principles to network engineering and operations to foster the agility and innovation of network configuration and management.It often leverages Software-Defined Network (SDN) to provide conventionally fixed network functions in a virtualized environment, i.e., Network Function Virtualization (NFV) technologies on generic telecommunications hardware [31]. Our CI/CD pipeline can be further extended to include NetDevOps for commissioning the network infrastructure.

Our work is inspired by the pull-based development model and the CI/CD pipeline concept with IaC to manage OOE tasks. Distinguishing our proposal from related work [9, 28, 48] as well as the two large research projects *COSMOS*[1] and *ADEPTNESS*[2] that also investigate application of DevOps to industrial cyber-physical systems, our research scope focuses on the engineers who perform OOE tasks and the coordination of their executions.

[1]See https://cordis.europa.eu/project/id/957254
[2]See https://cordis.europa.eu/project/id/871319

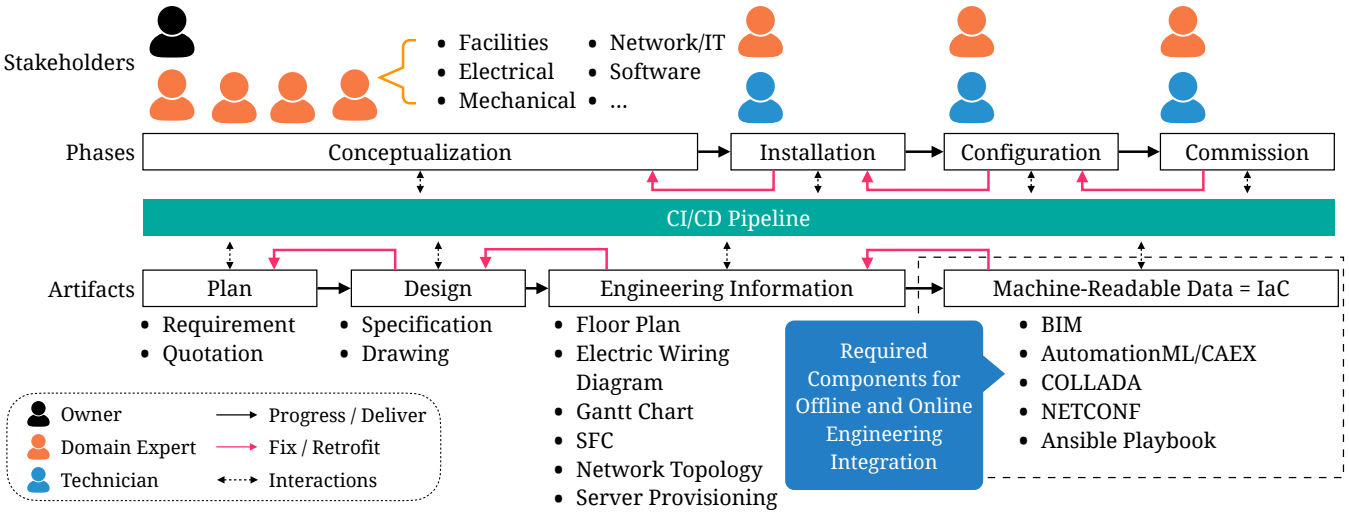


**Figure 3: Stakeholders, phases, and artifacts in industrial CPS development workflow coordinated through a CI/CD pipeline.**

## 3 CI/CD PIPELINE FOR ENGINEERING TASKS

In this section, we present the core proposal of our research and elaborate on the methodology of applying the CI/CD pipeline concept to the development workflow management as shown in Figure 3.

### 3.1 Task Description

One of the major challenges to apply CI/CD pipelines to industrial CPS is that most OOE tasks can only be fulfilled by domain-specific HITL methodologies. The integration of the tasks cannot be fully automated because the required tasks conceptually differ in different disciplines. To mediate between the tasks and CI/CD pipelines, we propose to describe the tasks using Thing Description (TD [30]) from the W3C Web of Things (WoT) Architecture [33]. WoT and TD have been developed to counter the fragmentation of protocols and interfaces of both virtual and physical entities (i.e., *Things*) in CPS.

A TD provides a machine-readable description about capabilities of a Thing with *property*, *action*, and *event* of the thing, and is not required to be hosted by the corresponding physical components themselves. This allows to use TDs to not only interface with tasks that can be systematically observed with existing automation tools, but also to integrate conceptual tasks that are (today) executed by humans – TD-described tasks as proxy for human agents (e.g., an expert electrician) and their actions. This feature of TDs would even allow creation of sub-tasks, updating the description of the task, and reassignment of the task to other human experts through the interface defined in the TD.

For example, a task to *verify a power cable is connected between plugs A and B* can be described as the TD shown in Listing 1. TD-described tasks can be serialized into the JSON-LD [32] format to benefit from the `@context` to semantically annotate the task and its required `actions` with the engineering tools by domain-specific knowledge models or vocabularies, such as ECLASS for manufacturing tools and planning data, SAREF [11] for smart appliances, SOSA/SSN [25] for sensors and actuators, and so forth. The `actions` (e.g., documenting the manual work conducted, uploading the video of the task, creating additional engineering data files, etc.) are consumed by WoT client applications to support the engineers to fulfill the tasks.

### 3.2 Lifecycle of Tasks

Lifecycle of the TD-described tasks in the CI/CD pipeline has the following six steps as depected in Figure 4. Note that we use GitHub as the Git hosting platform, ECLASS [17] as the semantic vocabulary, and AutomationML (AML) [14] as the engineering data format to explain the steps (i.e., the concept still applies to alternative choices: GitLab and UNSPSC for managing YANG [7], for instance). The CI tool on GitHub can be implemented as `Actions`[3] and Docker.

*3.2.1* ***Extraction*** *of tasks in the Git repository if the corresponding TD does not exist or is outdated in the main branch.* Tasks will be extracted from an AutomationML file on the pull event to the main branch of the repository. The task extraction process utilizes the topological hierarchy and the interlink between objects in AML, for example.

*3.2.2* ***Creation*** *of the extracted task in the previous steps.* When a new task extraction is detected, the CI tool automatically creates a new branch in the Git repository and commits the TD describing the task in the branch. For engineers to interact with the pipeline – especially, but not only for monitoring tasks in CI/CD – the new branch represents the task need to be fulfilled by engineers by instantiating the TD with an IRI (e.g., on GitHub, `https://raw.githubusercontent.com/...`).

*3.2.3* ***Assignment*** *of the newly created task/branch to the designated person with the required criteria of the task.* Using the ECLASS category linked in the TD, the CI tool identifies the right person to assign the task. The assignment is completed by creating an *Issue* on GitHub and assign users (i.e., the domain experts or technicians with the corresponding role on GitHub) to the task.

[3] See https://docs.github.com/en/actions

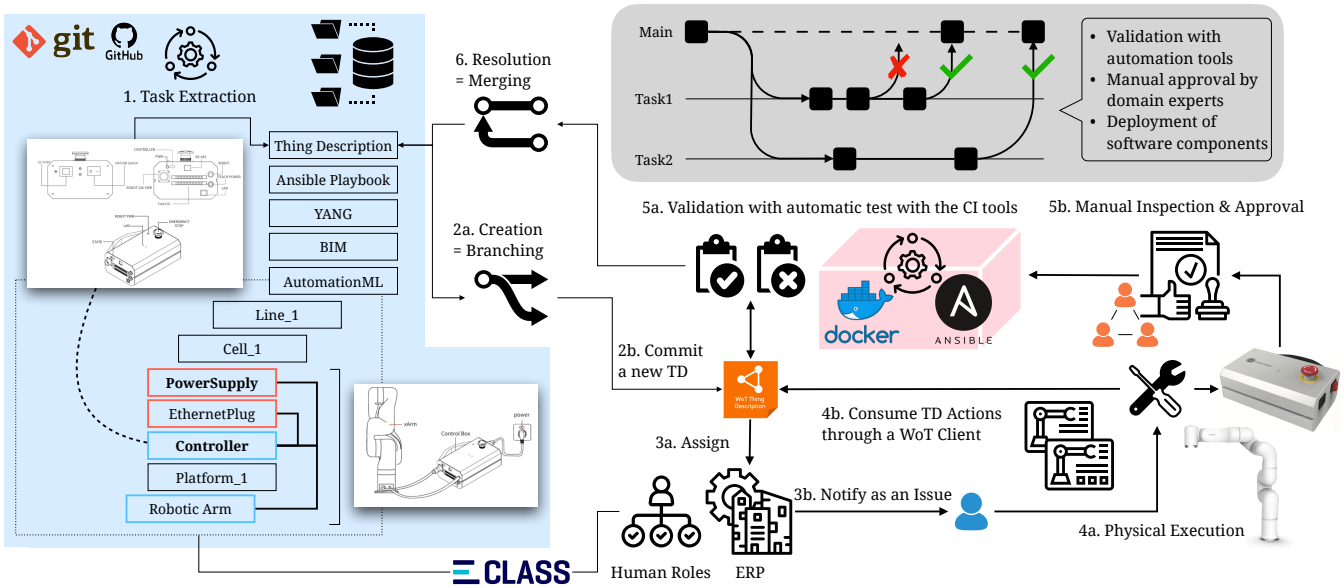


**Figure 4: Overview of an OOE task lifecycle as a TD in a CI/CD pipeline.**

*3.2.4* ***Execution*** *of the task.* This step is completed by the designated person manually fulfilling the (physical) task and consuming the `actions` of the TD through a WoT client application.

*3.2.5* ***Validation*** *of the task.* Completion of the `actions` triggers the CI tools to run automatic tests. On GitHub, `Checks`[4] can be used to monitor the TD properties after running the CI tool in `Workflow`as shown in Listing 2.

*3.2.6* ***Resolution*** *of the task.* At the end of the task lifecycle, the CI tools creates a Pull Request (PR) to merge the task branch to the main branch assigning another technician to review the change. This step can only be available if all the automated tests in the previous step are executed and pass the check conditions. With the manual approval of the PR, the task is resolved so that the main branch contains the TD-described task representing the corresponding task has been fulfilled on the shopfloor.

## 3.3 Virtual Engineering

Virtual Engineering (VE) is a concept specific to industrial CPS and advocates the iterative design and testing of automation systems on virtual builds [44]. Consider, if the motion controller demands a tolerance towards vibrations passed up from the base platform, the commissioning of the motion controller might be constrained by the robot's mounting condition. Such verification can be performed by simulation of the physical behaviour in Modelica or Simulink to test the capability of the component by *virtually commissioning* it before the deployment. For example, calibration of parameters in a field device's sensors and actuators validates the functional requirement of the device. With TDs, the CI/CD model can switch the pipeline between the actual components and their simulated "twins", which can also extend this VE method.

[4]See https://docs.github.com/en/rest/reference/checks

**Listing 1: A Thing Description example of the power cable wiring task extracted and created from an AutomationML file.**

```
"@context": [
  "https://www.w3.org/2019/wot/td/v1",
  {"ooe": "https://devops.autoidlabs.ch/ooe#"}
],
"@id": "https://raw.githubusercontent.com/...",
"title": "ConnectPLCPowerSupply",
"@type": "ooe:AMLInstallationTask",
"properties": {
  "status": { "@type": "ooe:TaskStatus", ...}},
"actions": {
  "document": {
    "@type": "ooe:InstallationDocument",
    ...,
  },
}
```

Semantic annotations of TDs facilitate cross-domain semantic interoperability throughout the CI/CD pipelines by supporting the diversity in the domain-specific knowledge with the external knowledge models. This linked data nature enables reasoning on the semantic models, which can also be utilized to validate the task workflows. In addition, since W3C WoT emphasizes the usage of hypermedia-based uniform interfaces [21], linking TDs to existing CI tools would work effectively – the CI/CD pipeline can be constructed through the TD interaction model based on compatible APIs. The TDs also enable the pipeline to communicate with industrial communication frameworks to access information on the shopfloor, for instance by leveraging the binary `opc.tcp` protocol binding or through RESTful extensions to OPC UA [24].

# 4 INFRASTRUCTURE AS CODE

In addition to the CI/CD pipeline, another key ingredient is the code to describe the industrial CPS infrastructure to bring the OOE to the DevOps ecosystem. As discussed in Section 2, OOE span different disciplinary domains with a diverse set of tools. The extraction step in the task lifecycle requires machine-readable data files containing the OOE information to manage their development under DVC. To this end, the most practical way is to use existing tools to configure the files. In the rest of this section, we analyze several data exchange formats and relevant technologies for the OOE information related to the disciplines presented in Table 1.

## 4.1 Mechanical and Electrical Engineering

AutomationML (AML, IEC 62714 [29]) is the most common data exchange format to describe offline engineering information in mechanical and electrical engineering domains such as topology, geometry, kinematics, behavior, and sequencing information for field devices. AML follows the object-oriented paradigm encapsulating different aspects of plant components as data objects. Based on Computer Aided Engineering Exchange (CAEX) meta model to describe the vendor-independent requirements, the capabilities, the interfaces, and the topology of components with semantic models [12]. Thus, AML files can be translated into knowledge graphs for verification and validation of the content [41]. With semantic integration of heterogeneous information models and tools, the VE method can automate the formal verification of the heterogeneous models of the programs that are executed by Programmable Logic Controllers (PLCs), e.g. with PLCopen XML stored in AML [44].

In addition, Asset Administration Shell (AAS [13], under development as IEC 63278-1) and its open-source implementations[5] are designed to provide access to information about plant components (e.g., AML, PDF documents, etc.) through a standardized API, integrating the field devices in the digital twin context [48] The AAS can also be used in our approach to interact with the system components information through as both TDs and AAS have the HTTP/1.1 interface.

## 4.2 Facilities and Network Engineering

In the building automation (BA) domain, construction, components, and their network are stored in *Building Information Model* (BIM) [4]. BIM data can be developed with CAD tools and enhanced by adding description of the BA installation using the domain ontologies, such as Brick and ifcOWL[6]. The two communication protocols BACnet and KNX[7] as well as their IP extensions (BACnet/IP and KNXnet/IP) are prominently used to interconnect the BA components throughout the entire facility.

Network engineers have traditionally relied on command-line interfaces to manage the configuration of network appliances, but this is error-prone and does not scale. This has led to the adoption of configuration management protocols (e.g., NETCONF [18]) and data modeling languages (e.g., YANG [7]) to describe the configuration and status of network appliances, interfaces, and the notification types so that the network configuration can be automated. Industrial networks are often multi-tiered, formed by a combination of TCP/IP, Industrial Ethernet (e.g., EtherCAT), and classical fieldbuses (e.g., MODBUS over RS-485). The communication bus needs to be established to enable the devices to join the local network while considering Quality of Service (QoS) requirements, including IEC/IEEE 60802 Industrial Automation profile for Time-Sensitive Networking (TSN) capability in the data link layer, and such network management is also possible through TDs [45]. At the time of writing, the OPC Field Level Communication (FLC) initiative[8] is working on the OPC Field eXchange (FX) standards to extend the OPC UA information model to the field level by describing functional entities and connection managers of devices and controllers with AML.

## 4.3 Software Engineering

For the Software Engineering aspect of IaC, existing DevOps toolchains can be leveraged in principle. A plethora of cloud infrastructure solutions for virtualization and containerization, for instance *Docker* and *Kubernetes* can be seamlessly integrated into our proposed CI/CD pipeline not only for implementing the CI tools but also deploying the artifacts in the system. Finally, the popular IaC orchestration technologies such as Ansible[9] can be the part of the CI tools to automate the configuration deployment and provisioning of the software components and their runtime environments.

# 5 PROOF OF CONCEPT

We applied the proposed CI/CD pipeline to a mock manufacturing plant in our research group's laboratory environment. The system involves the OOE of tasks for a processing cell consisting of a 7-DoF robotic arm and its controller, a power distributor, and their networking using GitHub, AML, YANG, and ECLASS as seen in Figure 1. The AML file was annotated with the ECLASS vocabularies to associate with the related disciplinary roles. The semantic representation of AML was generated by the CI tool implemented with AMLEngine2.1[10] to be executed on Windows self-hosted runners[11] for GitHub Action. The CI tool detects the AML file on *push* events on the Git repository, analyzes required tasks to integrate every AML component by traversing the topology of objects and interlinks using the *AutomationML Ontology*[12], and create the TD-described tasks. In a similar manner, the CI tool can extract network configuration tasks from a YANG file for the Ethernet switch connected to the controller through the distribution frame of the building and deploys the configuration with Ansible. A simple WoT client was implemented to serve a command-line user interface to interact with the created tasks for documenting the manual tasks conducted through the `action` of TDs via the HTTP/1.1 protocol binding.

With this mock system, we verify the multi-disciplinary OOE task coordination can be achieved by the proposed CI/CD pipeline

[5] See https://github.com/admin-shell-io/aasx-package-explorer
[6] See https://brickschema.org/ and https://technical.buildingsmart.org/
[7] See http://www.bacnet.org/ and https://www.knx.org/
[8] See https://opcfoundation.org/flc/
[9] See https://www.ansible.com/
[10] See https://github.com/AutomationML/AMLEngine2.1
[11] See https://docs.github.com/en/actions/hosting-your-own-runners/
[12] See https://github.com/i40-Tools/AutomationMLOntology

with TDs. In creating the prototypical DevOps platform for OOE integration in the operations processes, we benefit from the following aspects of DevOps in industrial CPS:

- The DevOps platform supports the management of tasks fulfilled by engineers of different disciplinary domains as a central axis to coordinate the resolution of dependencies in the infrastructure.
- By representing the tasks as Thing Description in the JSON-LD format, the tasks can be managed in the same DVC ecosystem as the machine-readable engineering data exchange formats.
- The knowledge graph and semantic annotations in the OOE data and the task descriptions establish interoperability and a cross-domain knowledge exchange during the development.
- The systematic mechanism of the CI/CD pipeline contributes to freeing engineers from the anxiety of creating unexpected points of failures and to help identifying missing steps in the operations processes.
- Our approach is platform-agnostic by using file-based aggregation of the heterogeneous information models and TDs as proxy for the HITL methodologies in OOE tasks.

**Listing 2: A GitHub Workflow to run the CI for AML files.**

```
on: push
jobs:
 aml-task-extraction:
  runs-on: [self-hosted, x64, windows-2019]
  steps:
   - uses: actions/checkout@v2
   - uses: ./.github/actions/aml-task-extractor
 task-creation:
  runs-on: [self-hosted, linux]
  needs: aml-task-extraction
  steps:
   - uses: actions/github-script@v4
     with:
      script: |
       const script = require('./task-composer.js')
       console.log(script({github, context}))
```

## 6 DISCUSSION AND CHALLENGES

In the course of this research, the technical challenges and open research questions of the proposed method were identified.

Automatic extraction of the tasks from the heterogeneous data exchange format requires the CI tools implemented specific to the local context of the shopfloor. By implementing a task extraction process in a CI tool, it can describe "*what components require what task*", but the logic composition needs to incorporate expert knowledge and dynamic environmental factors. To this end, the CI tool development should also be integrated in a CI/CD pipeline, supporting the composition process of the logic with knowledge models. For assisting domain experts in constructing knowledge models (e.g., OWL class expressions for AML object data), a semi-automated concept learning system may be useful [27].

From a project owner's point of view, the main concern is to manage the progress of OOE workflows across the shopfloor, but the number of tasks completed based on the engineering data stored in the repository does not necessarily reflect the actual situation. Such discrepancies can be caused by incomplete task coverage or missing engineering data or verification conditions. A mechanism to enhance the completeness of the OOE needs further investigation.

Finally, knowledge documentation itself is a tedious task [39]. In order to effectively integrate the manual work of engineers, we need to consider the HCI aspects of WoT client applications to interact with the tasks described in TDs. Such improvements can reduce the anticipated total cycle time of a task.

## 7 CONCLUSIONS AND OUTLOOK

We investigated the use of DevOps to support the offline and online engineering of industrial CPS. Concretely, the aim was to use DevOps to support the work of a multi-disciplinary engineering team involved in conceptualizing, installing, configuring, and commissioning the components that constitute industrial CPS. Our proposal brings DevOps to industrial CPS while explicitly acknowledging the requirements of *physical* components in such systems, and we suggested the usage of existing machine-readable data exchange format to describe the OOE information. Although the extraction of tasks depends on the data exchange format including its file schema and encoding, the over-arching approach, i.e., generating tasks at the integration point of each object in the topology hierarchy and at the interlinking between objects, can be harnessed by DevOps methodologies to integrate both system components and multi-disciplinary tasks. As an avenue for future work, we continue to implement and deploy this idea, and aim to demonstrate the benefits of integrating different development phases across hard- and software of industrial CPS through DevOps, compared to traditional approaches.

## REFERENCES


[1] Becker A, Görlitz J, Henke S, and Lecke M. 2020. Charter for Work and Learning in Industry 4.0. https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Charter-for-Work-and-Learning.html (Accessed on 08/28/2021).

[2] Sruthy Agnisarman, Snowil Lopes, Kapil Chalil Madathil, Kalyan Piratla, and Anand Gramopadhye. 2019. A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection. *Automation in Construction* 97 (2019), 52–76. https://doi.org/10.1016/j.autcon.2018.10.019

[3] Matej Artac, Tadej Borovssak, Elisabetta Di Nitto, Michele Guerriero, and Damian A. Tamburri. 2017. DevOps: Introducing Infrastructure-as-Code. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 497–498. https://doi.org/10.1109/ICSE-C.2017.162

[4] Salman Azhar. 2011. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and management in engineering* 11, 3 (2011), 241–252.

[5] Earl T Barr, Christian Bird, Peter C Rigby, Abram Hindle, Daniel M German, and Premkumar Devanbu. 2012. Cohesive and Isolated Development with Branches. In *Fundamental Approaches to Software Engineering*. Springer Berlin Heidelberg, 316–331. https://doi.org/10.1007/978-3-642-28872-2_22

[6] Tegawendé F Bissyandé, David Lo, Lingxiao Jiang, Laurent Réveillere, Jacques Klein, and Yves Le Traon. 2013. Got issues? who cares about it? a large scale investigation of issue trackers from github. In *2013 IEEE 24th international symposium on software reliability engineering (ISSRE)*. IEEE, 188–197. https://doi.org/10.1109/ISSRE.2013.6698918

[7] Martin Björklund. 2016. The YANG 1.1 Data Modeling Language. https://rfc-editor.org/rfc/rfc7950.txt (Accessed on 10/12/2021).

[8] Armando W. Colombo, Stamatis Karnouskos, and Christoph Hanisch. 2021. Engineering human-focused Industrial Cyber-Physical Systems in Industry 4.0 context. *Philosophical Transactions of the Royal Society A* 379, 2207 (2021), 20200366. https://doi.org/10.1098/rsta.2020.0366

[9] Benoit Combemale and Manuel Wimmer. 2019. Towards a model-based DevOps for cyber-physical systems. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 84–94. https://doi.org/10.1007/978-3-030-39306-9_6

[10] Denzil Correa and Ashish Sureka. 2013. Integrating issue tracking systems with community-based question and answering websites. In *2013 22nd Australian Software Engineering Conference*. IEEE, 88–96. https://doi.org/10.1109/ASWEC.2013.20

[11] Laura Daniele, Frank den Hartog, and Jasper Roes. 2015. Created in close interaction with the industry: the smart appliances reference (SAREF) ontology. In *International Workshop Formal Ontologies Meet Industries*. Springer, 100–112. https://doi.org/10.1007/978-3-319-21545-7_9

[12] Paul Danny, Pedro Ferreira, Niels Lohse, and Magno Guedes. 2017. An AutomationML model for plug-and-produce assembly systems. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 849–854. https://doi.org/10.1109/INDIN.2017.8104883

[13] Details of the Asset Administration Shell. 2021. *Part 1 - The exchange of information between partners in the value chain of Industrie 4.0*. Technical Report. Federal Ministry for Economic Affairs and Energy (BMWi). https://idtwin.org/en/

[14] Rainer Drath, Arndt Lüder, Jorn Peschke, and Lorenz Hundt. 2008. AutomationML - the glue for seamless automation engineering. In *2008 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 616–623. https://doi.org/10.1109/ETFA.2008.4638461

[15] Paul M Duvall, Steve Matyas, and Andrew Glover. 2007. *Continuous integration: improving software quality and reducing risk* (1st ed.). Pearson Education.

[16] Paul M Duvall and Michael Olson. 2018. Continuous delivery: Patterns and antipatterns in the software life cycle. https://dzone.com/refcardz/continuous-delivery-patterns (Accessed on 10/12/2021).

[17] ECLASS. 2021. *Classification and Product Description*. Technical Report. ECLASS e. V. https://www.eclass.eu

[18] Rob Enns, Martin Björklund, Andy Bierman, and Jürgen Schönwälder. 2011. Network Configuration Protocol (NETCONF). https://rfc-editor.org/rfc/rfc6241.txt (Accessed on 10/12/2021).

[19] Davide Falessi, Freddy Hernandez, and Foaad Khosmood. 2018. Issue Tracking Systems: What Developers Want and Use.. In *In Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*. SCITEPRESS – Science and Technology Publications, Lda., 577–582. https://doi.org/10.5220/0006818405770582

[20] Stefan Feldmann, Sebastian J.I. Herzig, Konstantin Kernschmidt, Thomas Wolfenstetter, Daniel Kammerl, Ahsan Qamar, Udo Lindemann, Helmut Krcmar, Christiaan J.J. Paredis, and Birgit Vogel-Heuser. 2015. Towards Effective Management of Inconsistencies in Model-Based Engineering of Automated Production Systems. *IFAC-PapersOnLine* 48, 3 (2015), 916–923. https://doi.org/10.1016/j.ifacol.2015.06.200

[21] Roy T Fielding and Richard N Taylor. 2002. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)* 2, 2 (2002), 10 pages. https://doi.org/10.1145/514183.514185

[22] Keheliya Gallaba, Yves Junqueira, John Ewart, and Shane Mcintosh. 2020. Accelerating Continuous Integration by Caching Environments and Inferring Dependencies. *IEEE Transactions on Software Engineering* Early Access (2020), 1–1. https://doi.org/10.1109/TSE.2020.3048335

[23] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, 345–355. https://doi.org/10.1145/2568225.2568260

[24] Sten Grüner, Julius Pfrommer, and Florian Palm. 2016. RESTful Industrial Communication with OPC UA. *IEEE Transactions on Industrial Informatics* 12, 5 (2016), 1832–1841. https://doi.org/10.1109/TII.2016.2530404

[25] Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. 2019. The SOSA/SSN Ontology: A Joint W3C and OGC Standard Specifying the Semantics of Sensors, Observations, Actuation, and Sampling. *Semantic Web-Interoperability, Usability, Applicability an IOS Press Journal* 56 (2019), 1–19.

[26] Martin W. Hoffmann, Somayeh Malakuti, Sten Grüner, Soeren Finster, Jörg Gebhardt, Ruomu Tan, Thorsten Schindler, and Thomas Gamer. 2021. Developing Industrial CPS: A Multi-Disciplinary Challenge. *Sensors* 21, 6 (2021), 1991. https://doi.org/10.3390/s21061991

[27] Yingbing Hua and Björn Hein. 2019. Interactive Learning Engineering Concepts in AutomationML. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1248–1251. https://doi.org/10.1109/ETFA.2019.8869182

[28] Jerome Hugues, Anton Hristosov, John J. Hudak, and Joe Yankel. 2020. TwinOps - DevOps Meets Model-Based Engineering and Digital Twins for the Engineering of CPS. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (Virtual Event, Canada) *(MODELS '20)*. ACM, 5 pages. https://doi.org/10.1145/3417990.3421446

[29] IEC 62714-1:2018. 2018. *Engineering data exchange format for use in industrial automation systems engineering - Automation Markup Language - Part 1: Architecture and general requirements*. Technical Report. IEC.

[30] Takuki Kamiya, Michael McCool, Sebastian Käbisch, Matthias Kovatsch, and Victor Charpenay. 2020. *Web of Things (WoT) Thing Description*. Technical Report. W3C. https://www.w3.org/TR/wot-thing-description/

[31] Holger Karl, Sevil Dräxler, Manuel Peuster, Alex Galis, Michael Bredel, Aurora Ramos, Josep Martrat, Muhammad Shuaib Siddiqui, Steven van Rossem, Wouter Tavernier, and George Xilouris. 2016. DevOps for network function virtualisation: an architectural approach. *Transactions on Emerging Telecommunications Technologies* 27, 9 (2016), 1206–1215. https://doi.org/10.1002/ett.3084

[32] Gregg Kellogg, Manu Sporny, and Markus Lanthaler. 2020. *JSON-LD 1.1*. Technical Report. W3C. https://www.w3.org/TR/2020/REC-json-ld11-20200716/

[33] Matthias Kovatsch, Ryuichi Matsukura, Michael Lagally, Toru Kawaguchi, Kunihiko Toumura, and Kazuo Kajimoto. 2020. *Web of Things (WoT) Architecture*. Technical Report. W3C. https://www.w3.org/TR/wot-architecture/

[34] Paulo Leitão, Stamatis Karnouskos, Luis Ribeiro, Panayiotis Moutis, José Barbosa, and Thomas Strasser. 2018. Integration patterns for interfacing software agents with industrial automation systems. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2908–2913. https://doi.org/10.1109/IECON.2018.8591641

[35] Qinghua Liu, Juanqiong Gou, and Luis M Camarinha-Matos. 2020. Towards Agile Operation for Small Teams in Knowledge Intensive Organizations: A Collaboration Framework. In *Working Conference on Virtual Enterprises*. Springer, 263–272. https://doi.org/10.1007/978-3-030-62412-5_22

[36] Iori Mizutani, Jonas Brütsch, and Simon Mayer. 2021. Towards Provenance Integration for Field Devices in Industrial IoT Systems. In *Provenance and Annotation of Data and Processes*. Springer International Publishing, 250–255. https://doi.org/10.1007/978-3-030-80960-7_21

[37] Jeff Morgan and Garret E. O'Donnell. 2017. Enabling a ubiquitous and cloud manufacturing foundation with field-level service-oriented architecture. *International Journal of Computer Integrated Manufacturing* 30, 4-5 (2017), 442–458. https://doi.org/10.1080/0951192X.2015.1032355

[38] W Patrick Neumann, Sven Winkelhaus, Eric H Grosse, and Christoph H Glock. 2021. Industry 4.0 and the human factor–A systems framework and analysis methodology for successful development. *International Journal of Production Economics* 233 (2021), 107992. https://doi.org/10.1016/j.ijpe.2020.107992

[39] María Luciana Roldán, Silvio Gonnet, and Horacio Leone. 2016. Operation-based approach for documenting software architecture knowledge. *Expert Systems* 33, 4 (2016), 313–348. https://doi.org/10.1111/exsy.12152

[40] Stefan Runde, Gerrit Wolf, and Michael Braun. 2013. EDDL and semantic web - From field device integration (FDI) to Future Device Management (FDM). In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*. IEEE, 1–8. https://doi.org/10.1109/ETFA.2013.6647962

[41] Marta Sabou, Fajar Ekaputra, Olga Kovalenko, and Stefan Biffl. 2016. Supporting the engineering of cyber-physical production systems with the AutomationML analyzer. In *2016 1st International Workshop on Cyber-Physical Production Systems (CPPS)*. IEEE, 1–8. https://doi.org/10.1109/CPPS.2016.7483919

[42] Shinobu Saito, Yukako Iimura, Aaron K Massey, and Annie I Antón. 2017. How much undocumented knowledge is there in agile software development?: Case study on industrial project using issue tracking system and version control system. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 194–203. https://doi.org/10.1109/RE.2017.33

[43] Nicole Schmidt, Arndt Lüder, Heinrich Steininger, and Stefan Biffl. 2014. Analyzing requirements on software tools according to the functional engineering phase in the technical systems engineering process. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 1–8. https://doi.org/10.1109/ETFA.2014.7005144

[44] Georg Ferdinand Schneider, Hendro Wicaksono, and Jivka Ovtcharova. 2019. Virtual engineering of cyber-physical automation systems: The case of control logic. *Advanced Engineering Informatics* 39 (2019), 127–143. https://doi.org/10.1016/j.aei.2018.11.009

[45] Luca Sciullo, Sushmit Bhattacharjee, and Matthias Kovatsch. 2020. Bringing deterministic industrial networking to the W3C web of things with TSN and OPC UA. In *Proceedings of the 10th International Conference on the Internet of Things*. IEEE, 1–8. https://doi.org/10.1145/3410992.3410997

[46] Antti Tenhiälä and Fabrizio Salvador. 2014. Looking inside glitch mitigation capability: The effect of intraorganizational communication channels. *Decision Sciences* 45, 3 (2014), 437–466. https://doi.org/10.1111/deci.12076

[47] Shoji Tomita and Mori Hiroshi. 1999. *FIELDBUS SYSTEM ENGINEERING*. Technical Report. Yokogawa Technical Report English Edition.

[48] Miriam Ugarte Querejeta, Leire Etxeberria, and Goiuria Sagardui. 2020. *Towards a DevOps Approach in Cyber Physical Production Systems Using Digital Twins* (1st ed.). Springer. https://doi.org/10.1007/978-3-030-55583-2_15