# SIAM DSL: Accessible Structured Text for Bridging the Architecture-Impact Gap in Cyber-Physical Production Systems

Stefan Biffl*[x] Tobias Bein[†] Sebastian Kropatschek[x] Kristof Meixner* Arndt Lüder[†]
*Institute of Information Systems Engineering, TU Wien, Vienna, Austria
[x]Center for Digital Production, TU Wien, Vienna, Austria
E-Mail: [first].[last]@tuwien.ac.at
[†]Institute of Engineering of Products and Systems, Otto von Guericke University, Magdeburg, Germany
E-Mail: [first].[last]@ovge.de

***Abstract*—In semi-automated Cyber-Physical Production Systems (CPPSs) that use operator assistance systems, introducing product variants often induces severe hazards due to misaligned mental models between the assistance system, human operators, and actual production states. These discrepancies create high-impact deviations between the expected and actual runtime states of a production line. To prevent these socio-technical hazards, solution architects require early-stage elicitation of tacit domain knowledge regarding critical production paths and operator conditions. However, multi-domain modeling approaches remain largely inaccessible to production domain experts. To bridge this gap, we present the *System Impact Architecture Model (SIAM) Domain-Specific Language (DSL)*, a lightweight, structured text language that integrates a System Architecture Model with a System Impact Model. The SIAM DSL provides a simple framework for capturing and comparing multi-domain Product-Process-Resource (PPR) variations, facilitating forward and backward reasoning to identify risky production states. We explore the language's efficacy through an illustrative case study of operator assistance design for gearbox assembly and derive a research agenda. The study results indicate that the SIAM DSL is effective for eliciting tacit dependencies and providing input to modeling socio-technical assistance scenarios.**

***Index Terms*—Production Systems Engineering; Industry 4.0; Domain-Specific Language; Model-Based Engineering**

## I. Introduction

Industrial digitalization has become an established reality across manufacturing sectors, driving expectations for enhanced system performance and operational outcomes [1]. While Industry 4.0 (I4.0) has focused on increasing flexibility through highly automated Cyber-Physical Production Systems (CPPSs) [2], the emerging Industry 5.0 vision emphasizes socio-technical production systems that integrate automation with human operators to address changes in a volatile environment [3]. In these systems, Operator Assistance Systems (OASs) are critical for providing real-time guidance and observing task execution to improve the performance of human-machine teams [4]. Despite the potential of OASs, introducing product variants frequently induces severe production hazards and low performance [4], [5].

According to *STAMP* theory [6], a socio-technical production system can be seen as a control system. Production hazards emerge from dysfunctional control systems in which the mental models of various actors, e.g., operators, CPPS engineers, and OAS designers, diverge from the actual system state, leading to unsafe actions. For instance, an operator fails to set the correct pressing force, resulting in invisible defects such as hair cracks, which reduce business performance.

These divergent model views stem from *architecture-impact gaps* among domain experts. In this paper, we define an *architecture-impact gap* as the systemic difference between design-time engineering models of the intended architecture and the actual run-time states of the production line, the operational impact. In multi-domain production engineering, this gap may arise from the use of a collection of partial architecture views, models, and Domain-Specific Languages (DSLs). These have traditionally focused on a fixed Product-Process-Resource (PPR) set for production [7], making it hard to model production states that contribute to emerging hazards due to flexible products, processes, or resources. Following Leveson [6], these diverging model views can be interpreted as open control loops that need to be closed by system design [8].

Bridging architecture-impact gaps is complicated, as the engineering and operation of CPPSs encompass diverse knowledge-intensive processes [9]. Critical domain knowledge on PPR assets is often tacit and scattered across disciplines [10], including product design, mechanical engineering, and quality management, each with distinct engineering habits that are hard to integrate [11]. Unfortunately, traditional modeling frameworks, such as RAMI 4.0 [12], are effective for representing static structures but lack the causal logic required for impact prediction. Graphical DSLs are often too complex for non-modeling experts [13], considering production variants, while simple structured-text languages like *Gherkin* [14] lack the depth required for systems engineering.

Therefore, the necessary integration of architecture and impact knowledge of the different experts requires an approach that supports an easy integration and analysis of domain knowledge that is (i) independent of domain-related vocabularies (neutral), (ii) easy to apply by experts with different levels of qualification and experience (simple), and (iii) facilitates the

Manuscript received July 14, 2026; revised August 12, 2026; accepted August 21, 2026; published August 25, 2026.
Issue category: Special Issue on DSD/SEAA 2026 on Works in Progress (WiP) Session, Krakow, Poland, Sept. 2026.
Paper category: Full
DOI: doi.org/10.64552/wipiec.v12i2.148

integration of further knowledge fields if required (extendable).

As a first step to address these concerns, this paper introduces the *System Impact Architecture Model (SIAM) DSL*. The SIAM DSL is a lightweight, structured-text language designed for non-experts in modeling to elicit and validate partial knowledge from diverse groups of domain experts. By integrating a *System Architecture Model* to represent PPR assets and a *System Impact Model* to capture causal paths to desired and undesired outcomes, the language provides a neutral, simple framework to identify risky production paths.

We illustrate the requirements and solution approach using an OAS use case as a specialization of an Industry 5.0 system (cf. Fig. 1). As indicated in [15], both the integrated OAS knowledge and knowledge presentation to the operator have an important impact on an OAS's effectiveness and efficiency. Especially, the tacit and, among experts across disciplines, fragmented knowledge of impact paths within a production system affecting desired and undesired production outcomes shall be integrated into OAS design methods.

Following Design Science [16], we investigate the following research question: ***RQ:*** *To what extent can a simple DSL represent and integrate architecture-impact knowledge emerging from different domain experts in a neutral and extendable way to improve the design and validation of socio-technical assistance systems?*

We build on the *System Impact Architecture Model* [8] for production to define a *SIAM DSL* for representing and validating the socio-technical system architecture with assumed cause-effect impact paths towards desired and undesired production outcomes as input to advanced approaches for engineering, simulation, application, and reuse. We evaluate the feasibility and efficacy of the SIAM DSL through an illustrative case study of gearbox assembly at the *HumTech Lab*,[1] focusing on the multi-domain dependencies between pressing force specifications and production quality.

The remainder of this paper is structured as follows. Section II summarizes related work. Section III introduces the gearbox assembly use case. Section IV introduces the SIAM DSL syntax. Section V reports on a preliminary application case study. Section VI discusses results of the preliminary case study and outlines a future research agenda.

## II. Related Work

We review human-centric manufacturing, systems control theory, and software abstraction techniques to establish the baseline for the System Impact Architecture Model DSL.

**Operator Assistance System design.** Within the Industry 5.0 vision that places the human operator at the center of volatile manufacturing environments, OASs are defined as technical systems that support operators during assembly or manufacturing tasks without replacing them [17]. OASs for assembly systems are part of a generic system architecture [18] (cf. Fig. 1), which consists of (i) the assembly process with its inputs and outputs, (ii) the process environment, (iii) the human operator, (iv) the CPPS to automate the assembly station, and (v) the OAS, a cyber-physical assistance system.

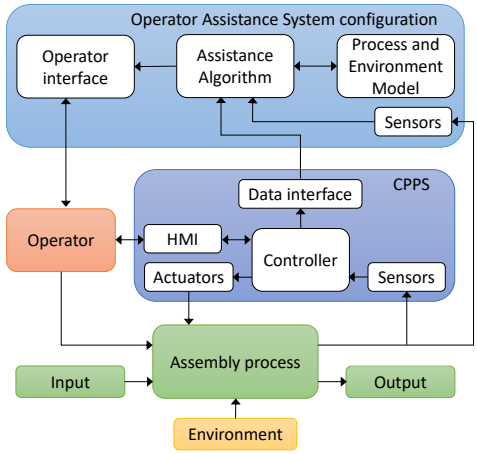



Fig. 1. Generic architecture of assembly and OASs [17], a control system [6].

These systems aim to compensate for human deficits while enhancing existing abilities, primarily through screen-based digital guidelines that reduce operator failures and improve product quality [17], [19]. Therefore, OAS engineers must sufficiently understand the typical control loops of the operator and the CPPS regarding production process performance, to understand what guidance is required or likely to improve operator performance. Traditional OAS design methods [20] exploit product knowledge from product design as well as PPR knowledge [7] from CPPS design.

However, recent work shows that OAS applications increase operator motivation, but frequently fail to reduce cognitive load [21]. A potential cause is insufficient support for quality issue resolution, which stems from a lack of integration of tacit architecture-impact knowledge into the OAS design [20]. An example is the multi-domain relationships between specific pressing forces and the risk of invisible defects. Usually, OAS engineers have only limited access to knowledge on the actual production control loops and need instructions from experts to design sufficiently accurate and complete guidance to the operator, especially based on architecture-impact knowledge related to unintended operator behavior. This work aims to combine PPR knowledge with impact paths to identify open control loops for improving OAS design.

**Socio-Technical Hazard Analysis and Control Theory.** Industry 4.0 and 5.0 require resilient, self-optimizing systems capable of proactive risk identification. While the Failure Mode and Effects Analysis (FMEA) [22] is the industrial risk assessment baseline, it relies on static, macro-level views. Yet, these are prone to information silos and fail to trace how local technical modifications propagate to business outcomes.

Therefore, this work builds on *STAMP* theory [6], which views socio-technical production as a control system. According to Leveson [6], hazards emerge from dysfunctional control systems, where the mental model views of diverse actors, including operators, CPPS engineers, and OAS designers,

[1] *HumTech Lab* at Otto von Guericke U.: https://tugz.ovgu.de/humtech.html

diverge regarding the assumed and actual runtime states (cf. Fig. 1). These discrepancies create high-impact deviations between expected and actual states, e.g., accidents, low production quality, and low business performance, requiring early-stage elicitation of tacit domain knowledge to identify and close critical open control loops.

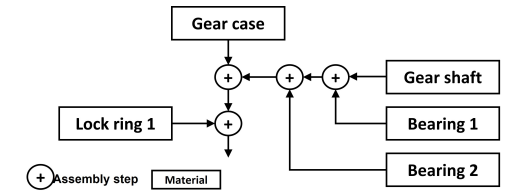
**Traditional Architecture and Impact Modeling.** Accurate CPPS modeling relies on established frameworks such as RAMI 4.0 [12] and the PPR VDI 3682 [23]. While these frameworks are effective for representing static architectures and hierarchical contexts, they lack the causal logic required for impact prediction, contributing to the architecture-impact gap. Further, current causal models often assume all confounding factors are observed [24], creating a simulation-to-reality gap that obscures physical dependencies in the case of technical changes. To address these limitations, Biffl *et al.* [8] introduced the SIAM meta model to integrate fragmented knowledge in condition networks linked to CPPS architecture. Therefore, this work builds on [8] to design the SIAM DSL for eliciting the knowledge required for facilitating the validation of hypothesized impact paths against production reality.

**Foundations for Accessible DSL Design.** A DSL offers focused expressive notations tailored to a particular problem domain [25]. Traditionally, a DSL aimed at domain experts to understand, validate, and modify programs in their own terms [26]. Recent DSLs leverage AI to derive engineering and operation artifacts [27]. Unfortunately, heavy DSLs, such as *SysML* [28] or *AutomationML* [7], are often too complex for non-modeling experts to elicit knowledge on production concerns and causal impacts of change, e.g., due to production variants [29] or volatile environments. In contrast, simple structured-text languages such as *Gherkin* (Given-When-Then) are useful to automate monitoring and testing [14], but lack the depth required for advanced systems engineering design. Generative AI has introduced a paradigm shift by enabling dynamic, context-aware low-level code synthesis and modification [27]. However, it has limited effectiveness in making high-level architectural decisions due to reliability concerns.

This paper takes a first step to bridge the architecture-impact gap for OASs by introducing the lightweight structural SIAM DSL. It serves as a boundary object to capture domain experts' contributions on architecture-related causal impact as input to AI-based solution generation, e.g., actionable operator-assistance scenarios to address risks along validated critical process paths with monitoring, documentation, and adaptation.

## III. Use Case Operator Guidance Adaptation for a Semi-Automated Assembly Workstation

This section introduces the use case *assemble a gearbox with operator assistance* to illustrate the requirements for bridging the architecture-impact gap during the design and operation of a socio-technical OAS for assembly processes [20]. We abstracted this use case from a domain analysis at the *HumTech Lab*, focusing on the design of an OAS [20] for semi-automated assembly of an abstracted gearbox to evaluate the efficacy of the SIAM DSL for OAS design.




| Mounting of gearbox | | | |
|---|---|---|---|
| 1 | Pre-mounting of shaft | Clamp shaft | Pick shaft |
| | | | Place shaft in pressing unit |
| | | | Close clamping |
| | | Press bearing 1 on shaft | Take bearing 1 |
| | | | Place bearing in press tool |
| | | | Press bearing on shaft |
| | | Change shaft clamping | Open clamping |
| | | | Pick shaft |
| | | | Place shaft in pressing unit |
| | | | Close clamping |

Fig. 2. Start sequence of a gearbox assembly process as a *gozinto* graph and its related three-level guidance information for a generic gearbox.

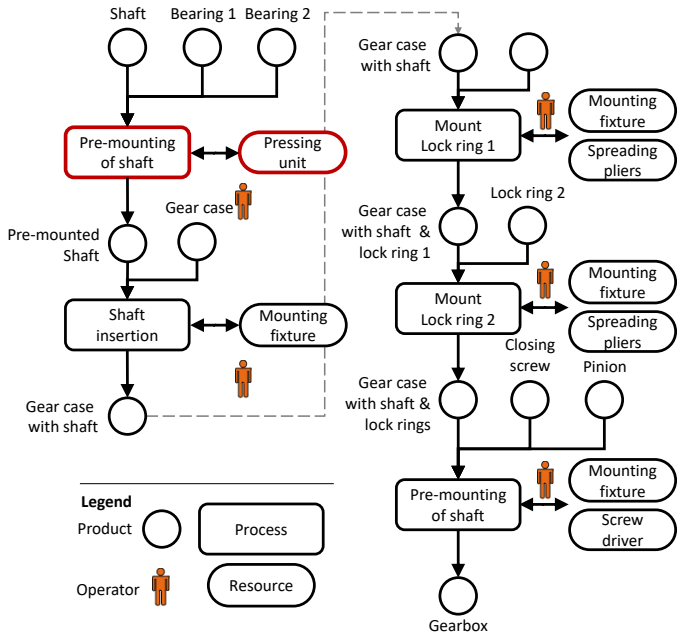



Fig. 3. PPR model [23] for the gearbox assembly process and station.

**OAS Design.** The workstation is designed for the manual assembly of a generic gearbox comprising a gear case, a shaft with two bearings, two lock rings, and a pinion leading to a combined *Bill of Materials* and a *Bill of Operations* modeled as a *gozinto* graph (cf. Fig. 2, upper part).

The production system design concerns an assembly station, based on a PPR network [23] (cf. Fig. 3), where a *pressing unit* executes a *pre-mounting shaft* process, and the OAS concept regarding operator assistance information by assembly task and associated resources (cf. Fig. 2, bottom). The operator can receive guidance, aligned with operator maturity levels, on three levels of detail: (i) naming the high-level process *pre-mounting of shaft*; (ii) naming process steps like *clamp shaft*; or (iii) detailed handling instructions like *pick shaft*.

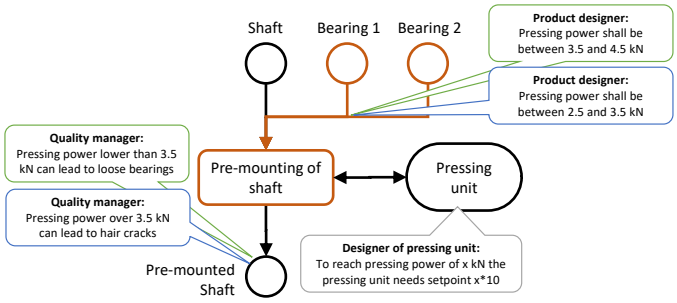



Fig. 4. PPR models [23], [29] on product variability for gearbox assembly.

An OAS, such as the camera-based *Smart Klaus*,[2] supports the process by observing mounting states and providing screen-based digital guidelines. This type of OAS effectively identifies and documents the fulfillment of discrete process steps, such as the successful placement of a shaft. However, it lacks visibility into internal machine parameters, including pressing force parameterization. This results in a divergence between the digital system's state knowledge, the operator's state and view, and the actual technical reality of the assembly.

**Tacit partial design knowledge.** Unfortunately, critical knowledge required to ensure process quality, such as the pressing force required to press the bearing on the shaft, is often tacit and scattered across stakeholder knowledge silos. For the step *Press a Metal Part*, generalized from the shaft pressing steps, the *Product Designer* holds implicit knowledge on the required force ranges based on material characteristics, e.g., surface structures. Simultaneously, the *CPPS Expert* understands the machine-specific logic required to set the pressing unit's parameters to achieve these setpoints. The *Quality Manager* (QM) identifies the causal outcomes of force deviations: a pressing force below thresholds, e.g., 3.5 kN, results in loose bearings, a failure easily detectable in quality control. However, a force exceeding the threshold leads to invisible hair cracks that escape standard inspection and cause product failure at the customer (cf. Fig. 4). The QM knows that different product variants call for different parameter settings. For example, gearbox variants A and B may use bearings of different materials, requiring distinct force thresholds, e.g., 2.5–3.5 kN vs. 3.5–4.5 kN, which pose hazards if operators do not adjust the force during product changes. If operators fail to adjust the pressing force for a specific variant, the resulting *Low-quality pressed metal part* represents a high-impact deviation between intended design and runtime state. Such implicit knowledge is not available to the *operator* of the assembly station, unless the knowledge is collected and integrated in the OAS. The SIAM DSL shall facilitate the elicitation and integration of the relevant knowledge about potential risks and represent the involved assets and links to their characteristics (cf. Fig. 4) to facilitate their evaluation in relation to intended and unintended operator behavior.

**Requirements.** For the SIAM DSL, we derived four essential *modeling requirements* from the use case: (R1) *System architecture model representation*, based on a production model (i) utilizing PPR architecture elements and (ii) identifications of variants and versions; (R2) *System impact model representation*, to connect conditions in the architecture with process outcomes; (R3) *Stakeholder knowledge area representation*, to document required stakeholders with access to knowledge and data; and (R4) *Modeling pragmatics* of the DSL to ensure accessibility to domain experts that are non-experts in modeling, (i) simplicity and domain-neutrality, (ii) extensibility for adaptation to specific domains, and (iii) usability in typical modeling contexts, e.g., easy to read, write, and process in a workshop, considering AI support, to provide input to advanced control system modeling and analysis.

[2]*Smart Klaus* OAS: https://www.optimum-gmbh.de/en/

## IV. System Impact Architecture Model DSL

To address the requirements collected in Section III, this section introduces the syntax of the SIAM DSL elements with examples (cf. Tabs. I to IV), based on the SIAM meta model [8]. To address requirement R1, representation of the *system architecture model* (cf. Fig. 5), Tabs. I to II define assets, asset network relations, and asset characteristics. To address requirement R2, representation of the *system impact model* (cf. Fig. 6), Tab. III defines conditions and condition impact paths. To address requirement R3, representation of *stakeholders with access to knowledge and data*, Tab. IV defines stakeholder knowledge areas, applicable both for assets and conditions. To address requirement R4, *modeling pragmatics*, the definition of the DSL is kept minimal to elements that participants actively used in modeling workshops to elicit domain knowledge, allowing for extending the DSL as required for specific applications, e.g., adapting/adding types, markers, or new syntax elements.

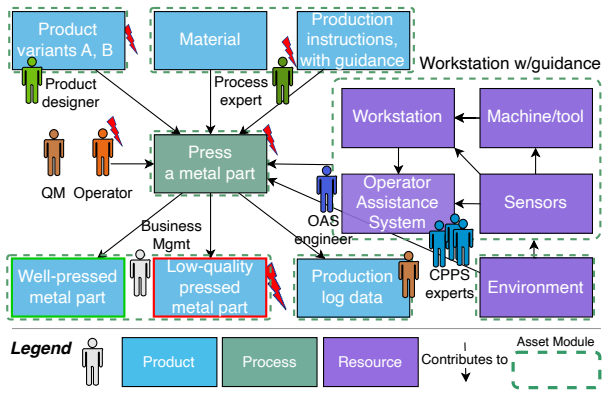



Fig. 5. System architecture model: process with CPPS and OAS (cf. Lst. 1)

The *Asset* (cf. Tab. I), in production a PPR asset [7], is the core of the system architecture model (cf. Fig. 5), identified by a name, unique ID with consideration of asset variants and versions, and an extensible type system. Markers provide a means to annotate assets with metadata for graph queries on the system architecture model, e.g., critical elements or assets that require validation.

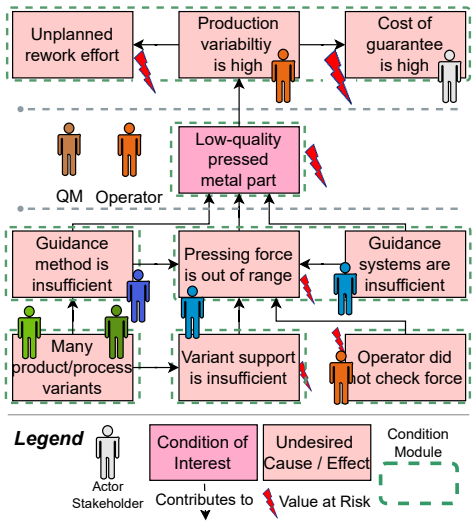



Fig. 6. System impact paths: undesired conditions (cf. Lst. 2).

TABLE I
SIAM DSL: ARCHITECTURE: ASSET NETWORK SYNTAX (CF. FIG. 5).

| **Syntax Rule** | ASSET <name>, <id>, <type>, [<markers>]. |
|---|---|
| <name> | Quoted string (human-readable label) |
| <id> | Unique ID with embedded variant/version |
| <type> | Dot notation set (e.g., product, process, resource.system) |
| [<markers>] | Optional list of symbolic tags in brackets |
| ASSET "Press a Metal Part", PROC-001v01, process, [critical]. | |
| ASSET "Material", PROD-001v01, product, [to_validate]. | |
| ASSET "Low-quality pressed metal part", PROD-002v01, product, [defect]. | |
| **Syntax Rule** | CONTRIBUTES_TO <asset1>, <asset2>, <asset3>, ... . |
| <asset> | Name or ID of an existing asset |
| Impact Path | Assets contribute sequentially: asset1 → asset2 → ... |
| Inputs Tuple | Grouped source assets: (asset1, asset2), asset3 |
| Outputs Tuple | Grouped target assets: asset1, (asset2, asset3) |
| *# — Simple Path and Direct Chains —* | |
| CONTRIBUTES_TO "Operator", "Press a Metal Part". # cf. Fig. 5 | |
| *# — Combined Input Assets —* | |
| CONTRIBUTES_TO ("Orders for Product Variants A, B", "Production Instructions with Guidance", "Material"), "Press a Metal Part". | |
| *# — Combined Output Assets —* | |
| CONTRIBUTES_TO "Press a Metal Part", ("Well-pressed Metal Part", "Low-quality Pressed Metal Part", "Production Log Data"). | |

The asset network relation *Contributes_to* (cf. Tab. I), connects assets to an asset network (cf. Fig. 5), a directed graph, by indicating that an asset contributes to a related asset, e.g., an operator, product, or system to a process. At its simplest, this relation connects two assets, identified by name or ID. For writing convenience, the SIAM DSL provides options to describe an asset impact path, or to group several input assets or output assets.

The *asset characteristic* (cf. Tab. II), refines an asset and provides a connection point for conditions that refer to an asset characteristic. The characteristic refers to its asset by name and ID, is identified by a locally unique name and version, and refers to a unit. The threshold vector holds a list of ascending key-value pairs that conditions can refer to,

TABLE II
SIAM DSL: ASSET CHARACTERISTICS SYNTAX AND EXAMPLES.

| **Syntax Rule** | ASSET_Characteristic <Asset_Name>, <Asset_ID>; <Characteristic_Name>; <Characteristic_ID>; unit <Unit> [; thresholds (<label>: <value>; ...)]. |
|---|---|
| <Asset_...> | Name and ID of an existing asset |
| <Characteristic_...> | Descriptive name and version identifier |
| <Unit> | Standard unit or custom unit (e.g., kN, s) |
| thresholds | Ascending label/value vector configuration |
| ASSET_Characteristic "Press a Metal Part", PROC-001v01; "Cycle time"; V02; unit s; thresholds (min: 5.0; normal: 10.0; max: 30.0). | |

e.g., *pressing force is high* refers to the threshold *high* of the characteristic *pressing force* that belongs to an asset, such as a product, process, or system.

TABLE III
SIAM DSL: CONDITION NETWORK SYNTAX (CF. FIG. 6).

| **Syntax Rule** | CONDITION "<statement>", <id>, <type>, <asset_ref>, [<markers>]. |
|---|---|
| <statement> | Free-form quoted natural language condition |
| <id> | Unique ID with variant/version (e.g., UCON-011v01) |
| <type> | Hierarchical type system, e.g., *condition.undesired* |
| <asset_ref> | Name or ID of an existing asset or characteristic |
| [<markers>] | Optional list of symbolic tags in brackets |
| CONDITION "Pressing force is out of range", UCON-008v01, condition.undesired, "Press a Metal Part", [critical, convergence_point]. | |
| CONDITION "Cycle time is within limits", DCON-212v01, outcome.desired, "Press a Metal Part", [to_validate]. | |
| **Syntax Rule** | CONDITION_CONTRIBUTES_TO <condition1>, <condition2>, <condition3>, ... . |
| <condition> | Statement/name or ID of an existing condition |
| Impact Path | Conditions contribute sequentially: c1 → c2 → ... |
| Inputs Tuple | Grouped source conditions: (c1, c2), c3 |
| Outputs Tuple | Grouped target conditions: c1, (c2, c3) |
| *# — Simple Path and Direct Chains —* | |
| CONDITION_CONTRIBUTES_TO "Low-quality pressed metal part", "Production variability is high", "Cost of guarantee is high". | |
| *# — Combined Input Conditions —* | |
| CONDITION_CONTRIBUTES_TO ("Pressing force is out of range", "Guidance method is insufficient", "Guidance systems are insufficient"), "Low-quality pressed metal part". | |
| *# — Combined Output Conditions —* | |
| CONDITION_CONTRIBUTES_TO "Product/process variance is high", ("Variant support is insufficient", "Guidance method is insufficient"). | |

The *Condition* (cf. Tab. III) is the core of the system impact model (cf. Fig. 6), identified by a statement, unique ID with consideration of condition versions, and an extensible type system. A condition shall refer to an existing asset or characteristic to ensure a valid connection between the system architecture and impact models as input to bridge the architecture-impact gap. Markers provide the means to annotate a condition with meta information for graph queries on the system impact model, e.g., critical conditions, or conditions that require validation or improvement.

The condition impact path relation *Condition_Contributes_to* (cf. Tab. III), connects conditions to a condition path or network (cf. Fig. 6), a directed graph, by indicating that a

condition contributes to a condition, e.g., pre-condition to a post-condition. In the simplest form, this relation connects a list of conditions, referred to by statement/name or ID. For writing convenience, the SIAM DSL provides options to describe a condition impact path, or to group several input conditions or output conditions.

TABLE IV
SIAM DSL: STAKEHOLDER KNOWLEDGE AREA SYNTAX (CF. FIGS. 5/6).

| **Syntax Rule** | PART_OF <stakeholder area> CONTAINS <stk1>, ... ; <item1>, <item2>, ... . |
|---|---|
| <stk. area> | Quoted string (human-readable label) |
| <stk>, ... | List of stakeholders (asset names) |
| <item>, ... | Quoted name/ID of an existing asset or condition |

*# — Asset Areas —*
PART_OF "Output Concerns" CONTAINS "Quality Manager";
"Well-pressed Metal Part", "Low-quality Pressed Metal Part".
*# — Condition Areas —*
PART_OF "Process Concerns" CONTAINS "Operator", "Quality Manager";
"Product/process variance is high", "Pressing force is out of range".

The stakeholder area relation *Part_of ... Contains* (cf. Tab. IV), describes for a stakeholder area the stakeholders and assets or conditions, for which these stakeholders have access to knowledge or data (cf. Figs. 5 and 6). The relation shall refer to existing assets (including stakeholders) and conditions.

## V. CASE STUDY ON THE APPLICATION OF THE SIAM DSL

We conducted a preliminary case study for the use case *gearbox assembly* to evaluate the application of the SIAM DSL with the requirements for a SIAM DSL. It was selected as a typical case, according to Runeson *et al.* [30], as the multi-domain interaction between pressing force specification, application, and validation is representative of high-value dependencies common in semi-automated assembly with assistance. Two researchers facilitated and observed a three-step SIAM workshop involving three domain experts who represented the roles Quality Manager (QM), OAS engineer, operator, and CPPS experts (cf. Section III). In the one-day workshop, participants and facilitators used the SIAM DSL to document architecture and impact models and gain insight on the pragmatic quality of the SIAM DSL.

The preliminary SIAM DSL application consisted of three steps: (1) Scoping of production with hazards and preliminary design of the system architecture model; (2) Socio-technical system architecture and impact analysis with domain experts; and (3) Specification of scenarios for operation assistance to prevent or mitigate production hazards due to the wrong pressing force. The case study yielded the following results.

**(1) Scoping and design.** The quality manager, as a proxy for the solution architect, defined the scope of work as the use case *assemble a gearbox with operator assistance*, with a focus on the process *Pre-mount shaft* with the critical step *Press a Metal Part*, which is representative of a wide range of similar assembly processes. From the PPR model (cf. Fig. 3), the QM derived the technical part of the architecture, shown in Fig. 5, illustrating the architecture of the socio-technical production system around the process step *Press a metal part*: input products, process, output products, and production systems required for automating production and operator assistance. The figure illustrates the key stakeholders with diverse knowledge on the production assets required to understand conditions that contribute to production outcomes.

**(2) System architecture and impact analysis.** To warm up, the QM and the domain experts followed the process *Pre-mount shaft* to understand the context of and contributions to the process step *Press a Metal Part* towards the desired outcome *Well-pressed metal part*. In parallel to the warm-up session, a facilitator provided LLMs (*Gemini* and *Qwen 3.5*) with the definition of the SIAM DSL and the PPR model (cf. Fig. 3, assets and asset network, without stakeholders) to extract the system architecture model in SIAM DSL (cf. Lst. 1). The domain experts validated the translation of the graphic model to the SIAM DSL, which was mostly correct.

***# — Process Assets ———————————***
ASSET "Press a Metal Part", PROC-001v01, process, [bolt].
*# — Output & Input Assets ———————————*
ASSET "Well-pressed Metal Part", OUT-001v01, product, [critical].
ASSET "Low-quality Pressed Metal Part", OUT-002v01, product, [defect].
ASSET "Production Log Data", DATA-002v01, data.
ASSET "Production Instructions with Guidance", DATA-001v01, data, [bolt].
ASSET "Orders for Product Variants A, B", DATA-003v01, data.
*# — Resource Assets ———————————*
ASSET "Operator", HUM-001v11, resource.human.
ASSET "Guidance System", SYS-001v01, resource.system.
ASSET "Sensors", SYS-002v01, resource.system.
ASSET "Workstation", SYS-003v01, resource.system.
ASSET "Machine/Tool", SYS-004v01, resource.system.
***# — Asset Network ———————————***
CONTRIBUTES_TO "Operator", "Press a Metal Part".
CONTRIBUTES_TO ("Orders for Product Variants A, B",
"Production Instructions with Guidance", "Material"), "Press a Metal Part".
CONTRIBUTES_TO ("Workstation", "Machine/Tool", "Guidance System"),
"Press a Metal Part".
CONTRIBUTES_TO "Press Metal Part", ("Well-pressed Metal Part",
"Low-quality Pressed Metal Part", "Production Log Data").
***# — Asset Characteristics: Pressing Force (Product A vs B) —***
ASSET_Characteristic "Press a Metal Part", PROC-001v01;
"Pressing force"; V12; unit kN;
thresholds (min: 0.0; low: 2.5; high: 3.5; max: 5.0).
ASSET_Characteristic "Press a Metal Part", PROC-001v01;
"Pressing force"; V13; unit kN;
thresholds (min: 0.0; low: 3.5; high: 4.5; max: 5.0). # cf. Section III

Listing 1: SIAM DSL: Asset Network Architecture (cf. Fig. 5).

Then, the QM and the domain experts explored direct and indirect causes that contribute to the undesired production outcome *Low-quality pressed metal part*. First, they inverted desired contributions along the sun-shine path to identify undesired conditions. Then they reasoned backward from the outcome *Low-quality pressed metal part* along the edges of the asset network (cf. Fig. 5) to systematically identify undesired conditions, resulting in a candidate list.

In parallel, a facilitator asked the LLMs for main causes of low-quality pressing and received a list of conditions with explanations. The domain experts discussed these candidate cause conditions and integrated new conditions into their list of conditions to validate.

Then they validated the conditions with the assets and identified the asset characteristics in Lst. 1 to represent different versions of thresholds for pressing force for product variants A and B with different material characteristics (cf. Fig. 4).

```
# — Conditions ————————————————
CONDITION "Pressing force is out of range", UCON-008v01,
condition.undesired, "Press a Metal Part", [critical, convergence_point].
CONDITION "Operator did not check force", UCON-004v01,
condition.undesired, "Operator", [critical].
CONDITION "Production variability is high", UCON-006v01,
condition.undesired, "Press Metal Part", [critical].
CONDITION "Pressing force is high", UCON-012v01,
condition.undesired, "Press a Metal Part".
# — Condition Impact paths ————————————
CONDITION_CONTRIBUTES_TO "Low-quality pressed metal part",
"Production variability is high", "Cost of guarantee is high".
CONDITION_CONTRIBUTES_TO ("Pressing force is out of range",
"Guidance method is insufficient", "Guidance systems are insufficient"),
"Low-quality pressed metal part".
CONDITION_CONTRIBUTES_TO ("Operator did not check force",
"Variant support is insufficient", "Guidance method is insufficient",
"Guidance systems are insufficient"), "Pressing force is out of range".
CONDITION_CONTRIBUTES_TO "Many product/process variants",
("Variant support is insufficient", "Guidance method is insufficient").
```

Listing 2: SIAM DSL: Condition Impact Paths (cf. Fig. 6).

Further, they identified impact paths (cf. the condition impact paths in Lst. 2), resulting in a list of main undesired impact paths. Shared conditions across several impact paths resulted in a network of main undesired impact paths (cf. Fig. 6) that the domain experts sketched on a whiteboard to understand visually the interaction of main impact paths.

Fig. 6 shows, for the undesired condition of interest, *Low-quality pressed metal part*, impact paths in the socio-technical production system that contribute to this condition. These impact paths follow the architecture impact paths (cf. Fig. 5).

Further, these impact paths concern knowledge of diverse stakeholders with access to engineering and runtime data to validate hypotheses about impact paths. Therefore, the domain experts added the stakeholder areas to both the asset and condition networks to validate a stakeholder role for each model element. They tracked the stakeholder areas in a table, consistent with the SIAM DSL (cf. Lst. 3). Lst. 3 represents the stakeholder areas for the asset network (cf. Fig. 5) as input to propagate these stakeholders to related conditions in the condition network (Fig. 6). In the workshop, this activity revealed model elements without stakeholders, a major source of risk to architecture-impact quality, and consequently, the introduction of new stakeholders, e.g., business management.

**(3) OAS scenario specification**. The SIAM DSL conditions provided input for the QM and OAS engineer to specify OAS behavior on critical paths, e.g., production states to observe and operator assistance to provide. Lst. 4 illustrates a feature in *Gherkin* [14] for operating assistance for *pressing a metal part with product type variation*, to warn the operator (cf. lines in violet color) after a change of product type, a production state that is prone to error, and to collect operator feedback.

Therefore, the workshop provided preliminary evidence for the pragmatic use of the SIAM DSL as a lightweight *hub DSL* to collect, integrate, and exchange architecture and impact

```
# — Asset Areas ———————————————
PART_OF "Process Concerns" CONTAINS "Operator", "Quality Manager";
"Press a Metal Part". # cf. Fig. 5 and Tab. IV
PART_OF "Output Concerns" CONTAINS "Quality Manager";
"Well-pressed Metal Part", "Low-quality Pressed Metal Part".
PART_OF "Input Concerns" CONTAINS "Quality Manager"; "Material",
"Orders for Product Variants A, B", "Production Instructions".
PART_OF "Systems" CONTAINS "CPPS expert";
"Workstation", "Guidance System", "Sensors", "Machine/Tool".
# — Condition Areas ———————————————
PART_OF "Process Concerns" CONTAINS "Operator", "Quality Manager";
"Product/process variance is high", "Pressing force is out of range".
```

Listing 3: SIAM DSL: Selected stakeholder knowledge areas.

```
Feature: Press a metal part with product type variation
As an assembly workstation operator,
I want to produce a high share of good metal parts, for varying product types,
So that the share of NOK gearboxes is low And the OEE is high.
Background:
Given an order for batches of several gearbox types # in production history
Given sufficient input materials of good quality
Given the assembly workstation and the operator assistance system work well
Given the operator is trained well # but may have more or less experience
@SadPath @High Waste in Production
Scenario: damage due to wrong force setting after product type change
Given the product type changed recently # compare input product with history
When the operator does not check the correct parameter value
When the operator presses the metal part with the assembly workstation
Then the share of bad metal parts is high # 95%, in-process or output
@Operator Assistance @AdaptationPath @Risk-Waste Reduction
Scenario: assemble after product type change
Given the product type changed recently # compare input product with history
When the operator assistance warns with the correct parameter value
When the operator sets, checks, and confirms the correct parameter value
When the operator presses the metal part with the assembly workstation
Then the share of good metal parts is high # 95%, in-process or output
Then the share of gearboxes NOK is low # 5%, inspection of output
```

Listing 4: Feature: Adaptive operator assistance instructions.

knowledge collected in different formats for validation, integration, and as input to efficiently design new model versions based on previous model parts in the SIAM DSL.

## VI. Discussion and Conclusion

This section evaluates the preliminary research results concerning the research question (RQ) *to what extent a simple DSL can represent and integrate architecture-impact knowledge emerging from different domain experts in a neutral and extendable way.* The application of the SIAM DSL demonstrates that a lightweight, structured-text language can effectively contribute to bridging the architecture-impact gap: by documenting validated architectural paths that support assumed impact logic, the DSL provides a simple framework to identify risky production states that traditional, static modeling frameworks often miss. In the gearbox assembly use case, the SIAM DSL facilitated the elicitation of tacit and scattered domain knowledge that had previously been isolated in the silos of product design, CPPS engineering, and QM.

**Preliminary assessment.** The SIAM DSL (i) by design represents asset networks (R1), condition impact paths (R2), and stakeholder knowledge areas (R3); (ii) is simple as it focuses on the required elements, described in a structured language, in the core close to the natural language that domain

experts use for naming assets, conditions, and impact paths; (iii) is pragmatic and minimal as the elements asset network, impact paths, and stakeholder areas are required to clarify valid solutions or open issues; (iv) is flexible and extensible to take up new asset types, characteristics, variants, and versions as required in the discussion as input to assessment and comparison, e.g., of local architecture views.

**Lessons Learned.** Several insights were derived from the feasibility study at the *HumTech Lab*. *Pragmatic integration*: The SIAM DSL served as a robust boundary object that unified inputs from spoken language, tables, and whiteboard sketches into a structured, textual representation suitable for both human experts and machine agents, enabling quick capture of elusive expert contributions. *AI-Readiness and Efficiency*: LLM functions proved instrumental in effectively and efficiently transforming human shorthand into complete SIAM DSL syntax while validating consistency across multi-domain inputs. *Exposing Open Control Loops*: Following STAMP theory, the DSL revealed critical open control loops where the traditional OAS lacked visibility into internal machine parameters, such as pressing force for product variants.

**Limitations.** The SIAM DSL is designed primarily for the elicitation and representation of knowledge in workshop settings. It is not intended to replace heavy systems engineering or simulation languages. In addition, the empirical results are subject to potential researcher and selection bias inherent in case studies derived from a standard production process.

**Conclusion.** The SIAM DSL provides a neutral, simple, and extendable framework for bridging the Architecture-Impact Gap in socio-technical CPPSs. By linking asset characteristics to condition impact paths, the language enables non-modeling experts to specify and validate multi-domain dependencies required for making OASs resilient to volatile environments.

**Research agenda.** *Empirical evaluation of the SIAM DSL*: We plan to strengthen the evaluation with PPR variability models to derive OAS configuration artifacts for mitigating risks of high-mix production variants. *Explore architecture-impact gap variants*: To identify and close open control loops, we plan to investigate to what extent scenarios can reveal architecture-impact gaps by documenting design-time intents and monitoring run-time states [31] as input to quality data and gap analysis. *Scaling of workshop results*: We plan to explore AI functions for the efficient integration of SIAM DSL artifacts into large CPPS and library models.

## References


[1] M. Abramovici and O. Herzog, *Engineering im Umfeld von Industrie 4.0*. Acatech, Utz Verlag, 2016.

[2] B. Vogel-Heuser, T. Bauernhansl, and M. Ten Hompel, *Handbuch Industrie 4.0 Bd. 4*, 2nd ed. Springer, 2017.

[3] Y. Lu, H. Zheng *et al.*, "Outlook on human-centric manufacturing towards industry 5.0," *Jour. Manuf. Sys.*, vol. 62, pp. 612–627, 2022.

[4] C. Bechinie, S. Zafari, L. Kroeninger, J. Puthenkalam, and M. Tscheligi, "Toward human-centered intelligent assistance system in manufacturing: challenges and potentials for operator 5.0," *Procedia Computer Science*, vol. 232, pp. 1584–1596, 2024, int. Conf. Industry 4.0 and Smart Manuf.

[5] G. Garcia-Garcia, Y. Singh, and S. Jagtap, "Optimising changeover through lean-manufacturing principles: A case study in a food factory," *Sustainability*, vol. 14, no. 14, 2022.

[6] N. G. Leveson, *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[7] M. Schleipen and R. Drath, "Three-view-concept for modeling process or manufacturing plants with AutomationML," in *IEEE ETFA*, 2009.

[8] S. Biffl, H. Rahmani, K. Meixner, D. Hoffmann, and A. Lüder, "Value-based Change Impact Analysis in Production with System Impact Architecture Modeling," in *Proc. IEEE Int. Conf. ETFA (in press)*, 2026.

[9] C. Di Ciccio, A. Marrella, and A. Russo, "Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches," *Jour. Data Semantics*, vol. 4, no. 1, pp. 29–57, Apr. 2014.

[10] S. Biffl, A. Lüder, and D. Gerhard, *Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects*. Springer, 2017.

[11] A. Lüder, D. Hoffmann, P. Hünecke *et al.*, "Entwicklung einer informationslogistik zur automatisierung des auomatisierungsengineering," in *Entwurf Komplexer Automatisierungssysteme (EKA)*, vol. 1, 2026.

[12] A. Shirbazo, B. Li, S. Ata, H. L. Ramandi, and S. Saydam, "A guideline for the standardisation of smart manufacturing and the role of rami 4.0 in digitising the industrial sector," *IEEE Internet of Things Journal*, 2025.

[13] D. Moody, "The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering," *IEEE Transactions on software engineering*, vol. 35, no. 6, pp. 756–779, 2009.

[14] M. S. Farooq, U. Omer, A. Ramzan, M. A. Rasheed, and Z. Atal, "Behavior driven development: A systematic literature review," *IEEE Access*, vol. 11, pp. 88 008–88 024, 2023.

[15] C. Stockinger, L. Polanski-Schräder, and I. Subtil, "The effect of information level of digital worker guidance systems on assembly performance," *Applied Ergonomics*, vol. 106, p. 103896, 2023.

[16] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[17] M. König and H. Winkler, "Investigation of assistance systems in assembly in the context of digitalization: A systematic literature review," *Journal of Manufacturing Systems*, vol. 78, pp. 187–199, 2025.

[18] H. Oestreich, M. Heinz-Jakobs, P. Sehr, and S. Wrede, *Human-Centered Adaptive Assistance Systems for the Shop Floor*. Cham: Springer International Publishing, 2023, pp. 83–125.

[19] M. König and H. Winkler, "Empirische ergebnisse zum einsatz von assistenzsystemen in der montage," *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, vol. 119, no. 7-8, pp. 563–568, 2024.

[20] B. Pokorni, D. Popescu, and C. Constantinescu, "Design of cognitive assistance systems in manual assembly based on quality function deployment," *Applied Sciences*, vol. 12, no. 8, 2022.

[21] M. Walczok and T. Bipp, "Investigating the effect of intelligent assistance systems on motivational work characteristics in assembly," *Journal of Intelligent Manufacturing*, vol. 35, pp. 1949–1962, 2022.

[22] DIN60812, *DIN EN 60812:2015-08: Failure Mode and Effects analysis (FMEA)*, International Standard, DIN EN Std. 60 812:2015–08, 2015.

[23] *VDI Guideline 3682 Formalised Process Descriptions*, Beuth Std., 2015.

[24] E. A. Akinluyi, K. Ison, and P. J. Clarkson, "Mapping outcomes in quality improvement and system design activities: the outcome identification loop and system impact model," *BMJ Open Quality*, vol. 8, no. 3, 2019.

[25] A. Van Deursen *et al.*, "Domain-specific languages: An annotated bibliography," *ACM Sigplan Notices*, vol. 35, no. 6, pp. 26–36, 2000.

[26] S. Greiner, B. Wiesmayr, K. Feichtinger, K. Meixner *et al.*, "Maturity evaluation of domain-specific language ecosystems for cyber-physical production systems," in *Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2023, pp. 1–8.

[27] M. Schieseck, P. Topalis, and A. Fay, "A graphical modeling language for artificial intelligence applications in automation systems," in *Int. Conf. on Ind. Inf. (INDIN)*. IEEE, 2023, pp. 1–7.

[28] L. Beers, H. Nabizada, M. Weigand, F. Gehlhoff, and A. Fay, "A sysml profile for the standardized description of processes during system development," in *Int. Sys. Conf. (SysCon)*. IEEE, 2024, pp. 1–8.

[29] K. Meixner, K. Feichtinger, H. S. Fadhlillah, S. Greiner, H. Marcher, R. Rabiser, and S. Biffl, "Variability modeling of products, processes, and resources in cyber–physical production systems engineering," *Journal of Systems and Software*, vol. 211, p. 112007, May 2024.

[30] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines*. John Wiley & Sons, 2012.

[31] S. Biffl, M. Martinelli, H. Rahmani, and M. Picone, "Business intelligence architecture for process quality monitoring with bdd," in *Proc. Software Quality Days (SWQD 2026)*. Springer LNCS, 2026, pp. 1–22.