

Multimodal Process Monitoring with On-Demand Disambiguation

Marco Franceschetti¹✉, Sejma Sijaric¹, Raúl Jiménez Cruz²,
Olaf Zimmermann¹, César Torres-Huitzil², and Barbara Weber¹

¹ University of St.Gallen, St.Gallen, Switzerland

² Tecnológico de Monterrey, Puebla, Mexico

marco.franceschetti@unisg.ch

Abstract. Process monitoring is increasingly supported by Internet-of-Things (IoT) environments. However, the abstraction of sensor data into process events often leads to ambiguity. Existing approaches that leverage IoT data do not address how to resolve such ambiguity dynamically. To address this challenge, we present MMPMOD, a novel architecture for multimodal process monitoring that detects ambiguity in IoT-based events and orchestrates contextual sensing modalities on-demand to resolve it. Upon ambiguity detection, it activates cameras to capture the visual context, and applies scene understanding to disambiguate events. We implement and evaluate the architecture in a phlebotomy scenario, demonstrating the feasibility of orchestration of complementary sensing modalities for disambiguation in online monitoring in this scenario.

Keywords: Ambiguity · Multimodal process mining · Computer vision.

1 Introduction

Internet of Things (IoT) technologies are increasingly integrated into business processes executed in the physical world. This integration promises unprecedented insights into process executions based on sensor data in near-real time [13]. While this allows monitoring process executions without relying on a process engine or information system producing execution data, it also introduces challenges in abstracting sensor data into meaningful process events [19]. Specifically, IoT data quality issues and imprecise abstraction procedures can lead to ambiguity: multiple process events relating to the same IoT data pattern. This, in turn, reduces the reliability of derived event logs and downstream analysis tasks [9].

The underlying reason for such ambiguity is that IoT data typically provides only local information [13]. For example, in a blood-extraction setting, IoT data may indicate that *something* is moving within the sensing range of a motion sensor, not that a syringe is being inserted into an arm. Such local information alone does not allow distinguishing between process events described by the same IoT pattern, e.g., syringe insertion and arm disinfection. In contrast, contextual information can provide cues that help disambiguate process events.

Knowing that a nurse is manipulating a syringe, for instance, helps rule out arm disinfection as the interpretation of the motion sensor’s data. The disambiguation potential of contextual information motivates integrating IoT data with complementary modalities able to contextualize IoT-based process events [11]. Video data provides such a modality by capturing a visual representation of the process environment. Unlike IoT sensors that measure localized signals, cameras can capture a broader view of the context in which process events occur. Prior work has explored the use of video data for process mining and activity recognition [16,18,25]. However, existing approaches typically assume continuous acquisition and analysis of video streams, which yields substantial computational overhead and may raise privacy concerns in many operational settings. Moreover, multimodal monitoring approaches generally treat sensing modalities as parallel inputs to activity recognition pipelines rather than exploiting them selectively.

These limitations point to the opportunity to leverage complementary sensing modalities in a more adaptive manner, investigating *how on-demand multimodal sensing orchestration supports efficient disambiguation of IoT-based process events*. Instead of continuously processing video data, contextual sensing modalities can be activated only when ambiguity arises. We therefore treat ambiguity detection as a trigger for orchestrating contextual sensing modalities, using video data as contextual evidence to disambiguate events in near-real time, as demonstrated in [22]. Because disambiguation often depends on recognizing the objects involved in an activity and how they are manipulated [26], we rely on scene understanding to identify relevant objects and human-object interactions. Following Design Science Research principles [21], we design, instantiate, and evaluate *MMPMOD*, a novel multimodal process monitoring architecture that combines IoT sensing with on-demand contextualization to disambiguate events. By improving the reliability of IoT-based event abstraction, MMPMOD supports the generation of more accurate event logs for downstream tasks. We instantiate MMPMOD in a realistic lab simulation environment for an IoT-monitored phlebotomy (blood-extraction) process, where contextualization is realized through computer vision. This setup provides a representative setting where people and physical objects interact causing ambiguity. We evaluate MMPMOD’s disambiguation accuracy and run-time performance, showing the effectiveness of on-demand disambiguation, with latencies appropriate for the specific domain.

While prior work [24] focuses on abstracting IoT events into process events, we address the complementary problem of resolving ambiguities that arise during event abstraction. We make three contributions: (1) a multimodal process monitoring architecture that detects ambiguity in IoT-based events and orchestrates contextual sensing modalities on-demand to resolve it; (2) a vision model trained on a phlebotomy dataset for recognizing objects and interactions characteristic of phlebotomy activities; (3) five generalized, actionable design principles for building adaptive, ambiguity-aware multimodal process monitoring systems.

In the remainder of the paper, Section 2 introduces a motivating scenario; Sect. 3 presents the architecture and vision model; Sect. 4 discusses implementation and evaluation; Sect. 5 reviews related work; Sect. 6 concludes the paper.

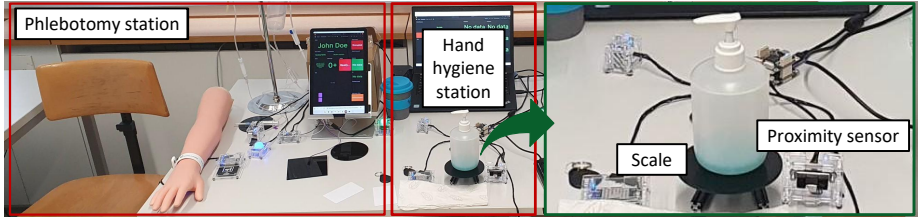


Fig. 1. IoT-augmented environment monitoring a simulated phlebotomy process

2 Motivating Scenario: Ambiguity in Event Abstraction

As a motivating scenario, we consider an IoT-augmented environment for monitoring healthcare worker (HCW) trainees practicing a simplified phlebotomy process using a training arm. The phlebotomy process and the environment implementation in our lab (cf. Fig. 1) have been validated by healthcare domain experts (cf. [8] for further details). Thus, it represents a realistic example of the challenge of ambiguity hindering monitoring in IoT-augmented environments.

Process monitoring in this environment is realized via IoT event abstraction, implemented through *activity detection services* [24]. In short, for each activity in the process specification (e.g., *HCW check-in*, *Perform hand hygiene*), a dedicated Web service runs in the background that, at run time, listens to IoT events generated by relevant sensors in the process environment. Upon a match of the run-time IoT events with a pattern of IoT events pre-specified at design time by a domain expert, each service emits a process event signifying the occurrence of the respective activity. For instance, a domain expert knows that hand hygiene events occur when a HCW is in proximity to the hand hygiene station and applies pressure to the sanitizer dispenser. This results in a detection service with an associated pattern `proximity_sensor <= 30 cm && scale >= 1000 g`, i.e., a pattern of IoT events emitted by a proximity sensor and a scale (cf. Fig. 1). Once deployed, whenever the IoT data produced by these two sensors satisfies the pattern conditions, the service emits a process event *Perform hand hygiene*.

Due to the lack of means (e.g., additional sensors) able to distinguish IoT event patterns at a finer-granular level, multiple services might be associated with the same pattern, each emitting the respective process event, thereby leading to ambiguity [8]. The setup in Fig. 1 leads to ambiguity in event abstraction. Here, we focus on three activities occurring multiple times in a process instance—namely, *Disinfection*, *Tourniquet application*, *Injection*—that are all matched by the same IoT event pattern involving proximity, motion, and illuminance sensors. This is because they are all performed at the same location (the phlebotomy station) and involve similar movements and changes in light conditions. As a result, their IoT patterns are indistinguishable and cannot be uniquely mapped to a single process event. Whenever one of these activities is performed, three services—one per activity—each emit their associated process event. This results in an incorrect trace in the event log, misrepresenting the actual enactment.

We propose to contextualize ambiguous process events by leveraging an additional sensing modality to disambiguate events based on the execution context. Assuming that the physical execution environment is visually observable, we propose to use cameras and scene understanding to identify objects and human-object interactions in the visual scene depicting an ambiguous event. This is motivated by two facts. First, adequately positioned cameras can capture a broad view over the area where a process unfolds, including the human actors and the objects they interact with. Second, often disambiguation depends on the identification of the objects involved in an activity and how they are manipulated [26]—a task that modern computer vision techniques can carry out in near-real time. In this setting, IoT sensing provides candidate process events but cannot distinguish between activities that share indistinguishable IoT patterns, while visual sensing provides complementary contextual evidence for process event disambiguation.

3 Architecture with On-demand Disambiguation

In line with Design Science Research principles [21], we specify design objectives addressing the identified problem and let them guide our architecture design.

3.1 Design Objectives

Drawing on the ambiguity challenges observed in the phlebotomy environment and on insights from IoT-BPM literature (cf. [13,19]), we establish the following non-exhaustive functional (FDO) and non-functional (NDO) design objectives.

- **FDO1: Detect ambiguity.** In IoT-augmented environments, event abstraction mechanisms such as the activity detection services in [24] often operate independently and may emit multiple process events for the same IoT event pattern. As these mechanisms are typically unaware of ambiguities they introduce by operating this way, the architecture should include functionality for detecting ambiguities introduced during event abstraction.

- **FDO2: Disambiguate on-demand.** Introducing an additional sensing modality for disambiguation yields computational overhead. To limit the impact on efficiency and scalability, additional sensing modalities should be used only upon ambiguity detection, instead of continuously acquiring and analyzing data.

- **FDO3: Disambiguate based on context.** Even with additional sensing modalities, data patterns may still appear similar across activities that are distinct (e.g., *tourniquet application* and *injection* both involve contact with the training arm and tool use). To distinguish between such cases and avoid misinterpretations, the architecture should rely on distinguishing contextual features such as relevant objects and human-object interactions.

- **NDO1: Performance.** The architecture should support online process monitoring and feedback provision for ongoing executions. Ambiguity detection and resolution should be performed online rather than offline, with acceptable tradeoffs between accuracy and latencies appropriate for the domain.

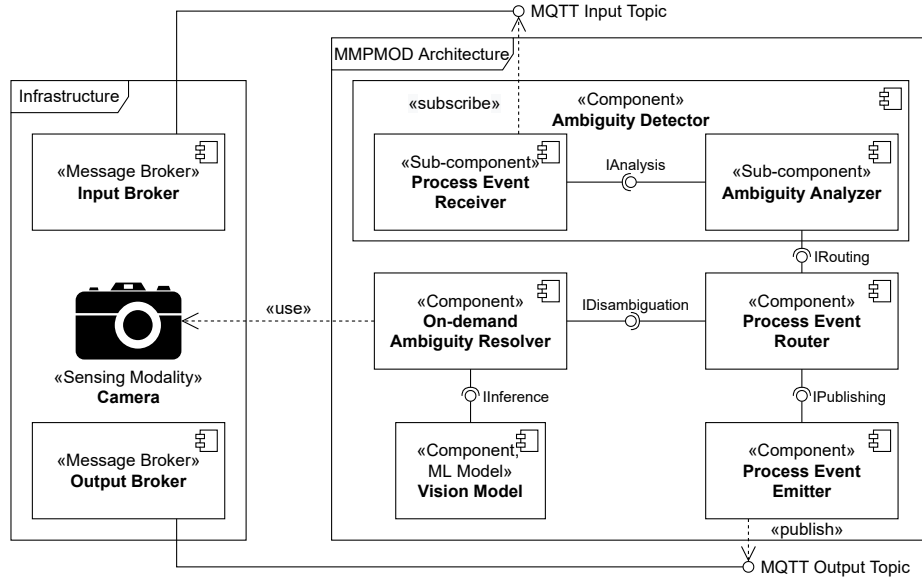


Fig. 2. UML Component diagram for the MMPMOD architecture

- **NDO2: Integrability.** As existing IoT setups are often complex and costly to modify, the architecture should integrate existing IoT-augmented environments without requiring extensive reconfiguration.

- **NDO3: Robustness, resilience, and recoverability.** The architecture should remain operational even when disambiguation cannot determine a single event interpretation. Moreover, it should support recoverability so that process event flow can continue or resume despite temporary failures.

- **NDO4: Security and privacy.** Since additional sensing modalities such as cameras may capture sensitive information (e.g., images of individuals' faces), the architecture should prevent such information from being exposed.

- **NDO5: Modularity.** As process environments evolve over time, the architecture should allow introducing new sensing modalities and ambiguity detection and resolution methods. Thus, it should comprise loosely coupled, interchangeable components that can be updated or replaced independently.

3.2 MMPMOD Architecture

The proposed architecture addresses the challenge of integrating heterogeneous sensing modalities to disambiguate events in an adaptive manner, without continuously processing all data sources. It comprises two external interfaces and five components, illustrated in Fig. 2 and described in detail in the following.

Ambiguity Detector. Ambiguity detection is a critical design objective (**FDO1**) of the architecture. The Ambiguity Detector is the component that fulfills this

design objective: without it, the system would be unable to distinguish between ambiguous and unambiguous events. This component ingests a stream of IoT-based process events and identifies cases where two or more process event candidates exist for the same IoT event pattern. It consists of two sub-components: (i) a *Process Event Receiver*, which provides access to the stream of IoT-based process events through an input channel (e.g., MQTT-based), and (ii) an *Ambiguity Analyzer*, which analyzes the incoming event stream to detect ambiguity.

The Ambiguity Analyzer can employ different strategies to detect ambiguity. A general, implementation-agnostic approach is based on windowing the event stream. Depending on the process characteristics, this may involve multiple concurrent windows—one per process instance or execution thread—identified and correlated through attributes such as case IDs. This strategy supports processes with concurrency or parallelism. Under more restrictive assumptions, for example when the process guarantees no concurrent instances or parallel executions, the windowing approach can be simplified to a single window. The window type and size depend on the process characteristics; ambiguity manifests when a window contains multiple events—one per interpretation of an IoT event pattern. If the implementation of the activity detection procedures can be inspected easily, a static analysis of their code is an alternative strategy. By identifying overlapping conditions in the patterns matched by the procedures, this analysis can reveal ambiguities among the associated process events. The Ambiguity Analyzer can use these results to determine when disambiguation is required. The Analyzer hands control to the Process Event Router to initiate disambiguation if ambiguity is detected, and to initiate event republishing otherwise.

Process Event Router. A core requirement of the architecture is modularity: the system should be structured so that components can be developed, replaced, or extended without affecting others. While the decomposition into an Ambiguity Detector, an On-demand Ambiguity Resolver, and a Process Event Emitter already contributes to this goal, modularity also requires that these components do not embed assumptions about each other’s control flow. To this end, the Process Event Router centralizes the coordination logic. It defines when ambiguity resolution is triggered, how results are processed, and when events are forwarded, thereby preventing the functional components from having to call or depend on one another. This design localizes control-flow changes to a single component: modifications to the handling of ambiguity such as alternative resolution strategies or different policies for unresolved cases affect only the Process Event Router rather than requiring changes in multiple modules (cf. **NDO5**). Concretely, this component handles three cases: (i) it receives requests from the Ambiguity Analyzer to resolve ambiguous events and invokes the On-demand Ambiguity Resolver; (ii) it receives disambiguation results from the On-demand Ambiguity Resolver and forwards the resulting unambiguous or remaining ambiguous events to the Process Event Emitter; and (iii) it receives requests from the Ambiguity Analyzer to publish unambiguous events and forwards these to the Process Event Emitter. By encapsulating coordination decisions, the architecture preserves modularity across detection, resolution, and emission components.

On-demand Ambiguity Resolver. On-demand disambiguation (cf. **FDO2**) and context-based disambiguation (cf. **FDO3**) are two other critical functional design objectives. The On-demand Ambiguity Resolver is the component that satisfies them. This component is invoked by the Process Event Router after the Ambiguity Analyzer identifies ambiguity in the input event stream. Once invoked, it activates the additional modality to acquire data about the current context. Next, it uses this data to attempt disambiguation. In case of successful disambiguation, it communicates the result, i.e., a unique event label, to the Process Event Router for publishing a corresponding disambiguated event. In case of unsuccessful disambiguation, it passes all ambiguous events—to be flagged as such—to the Process Event Router to let downstream tasks handle them. How such unresolved ambiguities are handled by these tasks is out of scope.

This component can adopt various modalities to support contextualization and disambiguation. In our lab setup, we use cameras that capture static images to acquire a visual representation of the current context. Specifically, we use only one downward-facing camera positioned above the training arm to avoid capturing sensitive information such as individuals’ faces, but in principle multiple cameras could be used to capture images from different positions and angles.

Vision Model. A fine-tuned Vision Model (described in detail in Sect. 4.2) is invoked by the On-demand Ambiguity Resolver to disambiguate ambiguous process events. The Resolver provides the Vision Model with the relevant image data—captured around the time the ambiguity was detected—and the model performs a classification task that distinguishes between the possible interpretations of the IoT event pattern. The set of labels used in this task corresponds to the activity types defined in the monitored process model; these labels are either specified as part of a process ontology or derived from annotated training data collected during system setup. Although our implementation uses computer vision for this task, alternative modalities such as audio signals or normative process specifications (when run-time deviations are impossible or can be controlled for) can be integrated analogously into the Ambiguity Resolver.

Process Event Emitter. Analyzed process events—whether unambiguous or the result of a disambiguation attempt—must be made available to external systems consuming them (cf. **NDO2**). The Process Event Emitter fulfills this role. It receives process events forwarded by the Process Event Router, originating either from the Ambiguity Analyzer (for events deemed unambiguous) or from the On-demand Ambiguity Resolver (for events for which disambiguation was attempted). It then publishes each event as an individual message to an output channel. To ensure clarity for downstream consumers, the Emitter marks disambiguated events as unambiguous, while events for which disambiguation was inconclusive are explicitly flagged as ambiguous. This information facilitates subsequent analysis such as manual disambiguation [22] or algorithms that must account for uncertainty. In this way, the Process Event Emitter provides external systems, including conformance checking algorithms, with an event stream that reflects the current system-level interpretation of the IoT data.

4 Implementation and Evaluation

Here, in line with the demonstration and evaluation phases of [21], we discuss the architecture implementation, the phlebotomy vision model, and their evaluation.

4.1 Architecture Implementation

To demonstrate the feasibility of the proposed architecture and evaluate it, we implemented a prototype¹ for the phlebotomy scenario described in Sect. 2..

The prototype implements the Ambiguity Detector, On-demand Ambiguity Resolver, Process Event Router, and Process Event Emitter in Java (Spring Boot, Modular Monolith). The On-demand Ambiguity Resolver uses a camera as an additional data source and computer vision to achieve disambiguation. Its sub-components responsible for image capture and scene understanding are implemented in Python to leverage libraries dedicated to these tasks (cf. Sect. 4.2).

To integrate our preexisting phlebotomy environment (cf. **NDO2**), the Process Event Receiver in the Ambiguity Detector subscribes to an existing MQTT topic where activity detection services publish process events [24]. Likewise, the Process Event Emitter publishes messages encapsulating process events on another MQTT topic. This allows the new components to consume the preexisting input and produce the output without modifying the original infrastructure.

The Ambiguity Detector finds ambiguities via a *windowing strategy* based on three characteristics of the phlebotomy process in our lab setup. First, the process does not allow for parallel activity executions (cf. *parallel split* in [5]); likewise, blood extractions are carried out on one training arm at a time, i.e., there are no concurrent instances. Therefore, only one process event can occur at any given point in time. Second, based on our experience with the process enactment by HCW trainees [8], each activity lasts at least five seconds. Third, the activity detection services typically emit process events without significant latencies. Consequently, we treat multiple events observed within a one-second window as candidate interpretations of the same IoT event pattern originating from different activity detection services, i.e., to be ambiguous. Thus, the Ambiguity Detector leverages a one-second session window to detect ambiguity.

4.2 Vision Model for the Phlebotomy Process

The vision model focuses on three activities of the phlebotomy process: *Disinfection*, *Tourniquet application*, and *Injection*. To enable scene-based disambiguation, the model follows a three-stage pipeline comprising: (1) object segmentation and detection, (2) spatial interaction and temporal aggregation and reasoning, and (3) rule-based activity reasoning to infer clinically meaningful activities from camera images. This design provides an explainable and efficient mechanism for disambiguating events. The model is based on YOLOv11, selected for its trade-off between real-time inference and instance segmentation accuracy [23].

¹ The source code is publicly available at <https://github.com/ics-unisg/mmpmod>.

Object segmentation and detection. The model was trained on a dataset from [14], which contains 11,884 annotated images of realistic blood-drawing scenes with five annotated object classes: syringe, tourniquet, disinfectant wipe, training arm, and gloves. Each object instance is labeled with bounding boxes and pixel-level masks in YOLO format. During inference, the model outputs both bounding boxes and segmentation masks for each object, enabling fine-grained spatial reasoning, such as measuring relative distances and contact areas. This allows the model to capture manipulation cues that IoT data alone cannot capture [18].

Interaction and temporal reasoning. Detected gloves act as proxies for the HCW’s hands. When both gloves are present, their segmentation masks are merged into a single manipulation region. For each detected object, the model computes a fused overlap score, balancing spatial precision and geometric coverage by weighting the intersection over union between masks and bounding boxes. The score gives 70% weight to mask overlap and 30% to bounding-box overlap, increasing robustness under partial occlusions. This score is complemented by the normalized Euclidean distance between glove and object centroids, capturing proximity cues. Within a fixed aggregation window of ten images (one every 100 *ms* for sufficient temporal separation between interaction states, yielding a 1 *s* window for stable interaction capture), detected activities are combined via majority voting.

Activity reasoning. Once glove-object interactions are established, a symbolic reasoning layer interprets them into phlebotomy activities. The system identifies *injection* when gloves interact with both the *training arm* and the *syringe*, *tourniquet application* when gloves interact with the *training arm* and the *tourniquet* but not the syringe, and *disinfection* when gloves interact with the *training arm* and optionally the *disinfectant wipe* while the syringe is absent. Each activity’s confidence is calculated as the mean detection confidence of its contributing objects, further adjusted by temporal consistency across the aggregation window. A robust confidence value is then derived by subtracting half of the standard deviation from the mean, penalizing unstable detections. This rule-based reasoning ensures explainability aligned with the semantics of the phlebotomy process.

4.3 Evaluation

The evaluation assesses the extent to which the proposed architecture fulfills the design objectives, using MMPMOD’s implementation and its integration in an end-to-end monitoring scenario. In the spirit of transparency and reproducibility, the complete datasets and evaluation results are available in an online appendix².

Vision model evaluation. We first evaluated the vision model’s ability to detect phlebotomy-relevant objects reliably in isolation using the test set with ground-truth annotations from [14]. Since YOLO inference is deterministic for fixed model weights and inputs, a single evaluation run was sufficient. The model

² <https://doi.org/10.5281/zenodo.18892612>

achieved high performance across all objects. Gloves and the training arm were detected with near-perfect precision and recall; the syringe and tourniquet also showed strong performance, each exceeding an F1-score of 0.88. The disinfectant wipe proved most challenging due to its small size and motion blur, yet its recall of 0.69 was sufficient for robust recognition when combined with temporal reasoning. Qualitative inspection of full video sequences further confirmed that the model correctly identifies the temporal order of actions and transitions between activities. Most false detections occurred during brief ambiguous movements, for instance when gloves passed near the syringe without manipulation. Temporal smoothing and contextual reasoning effectively filtered out these inconsistencies.

End-to-end evaluation. To evaluate the run-time performance of MMPMOD, we instantiated it in our lab setup. We deployed the prototype on a MacBook Pro M3 Max (36GB RAM), connected with a Logitech StreamCam camera (1920×1080 resolution). We performed 50 complete phlebotomy enactments, thereby generating, through existing activity detection services, 1766 process events that the prototype ingested on the fly. We recorded the time required for each disambiguation, which includes camera activation overhead (a fixed average time of 0.85 s intrinsic to our technical setup), capture of ten images (one every 100 ms, cf. Sect. 4.2), and label inference. Moreover, we saved the events in output to compare them with a manually-defined ground truth to compute disambiguation accuracy. The ground truth was derived from predefined enactment scripts specifying the exact sequence of activities to be performed during each enactment. During enactment, two authors monitored adherence to the script in real time to detect deviations. Since the activity sequence was predetermined and verified during enactment, no post-hoc interpretive annotation process was required.

During process enactment, we witnessed 259 occurrences of multiple events appearing in the same session window, which were flagged as ambiguous by the Ambiguity Detector. 18 out of these 259 flagged ambiguities were false ambiguities caused by erroneous sensor readings rather than by an ambiguous activity execution. For these 18 false ambiguity occurrences, the Ambiguity Resolver was triggered and correctly returned no unique label in 16 occurrences; it incorrectly returned an activity label in 2 occurrences. The remaining 241 occurrences of flagged ambiguous events corresponded to genuine ambiguities that required resolution: of the respective disambiguation attempts, 22 did not result in a correct label. Specifically, in 16 attempts the system could not determine a label, and in 6 it produced an incorrect one. For all the other 219 attempts, the system returned the correct label, resulting in a disambiguation accuracy of $219/241 = 0.91$.

Unambiguous events were typically emitted within 10 ms. The time required for disambiguation attempts was on average 3.69 s (median: 3.63, σ : 0.17)—a value lower than the typical duration of the ambiguous activities. As disambiguation always completed before the start of the next activity, we conclude that on-demand vision-based disambiguation meets latency requirements in this scenario.

Ablation study. To assess the impact of temporal visual aggregation on disambiguation, we conducted a post-hoc ablation study. In the implementation,

disambiguation is performed by aggregating predictions from ten images captured over one second. To isolate the contribution of temporal aggregation, we re-executed disambiguation while varying the number of images used to derive a prediction. Specifically, we evaluated settings using zero images (using a common *null-model* baseline that randomly selects an interpretation, akin to [20]), one image (vision model without temporal aggregation), two, three, four, and five images (reduced temporal aggregation). In each case, all other components remained unchanged. This design allows examining how the amount of temporal visual context affects disambiguation accuracy and time. Because the images captured during process enactment were stored, we were able to perform this analysis *ex post* by re-running the vision model on them. For the one-image condition, we used the first captured image to predict the activity label. For larger image counts we aggregated predictions from the first $N \in \{2..5\}$ stored images.

Fig. 3 summarizes the accuracy results of the ablation study; the full results, including execution times, are in the online appendix. The null-model baseline achieved an accuracy of 0.31 (95% Wilson CI [0.26, 0.37]), while using visual contextualization with a single image reached 0.79 (95% Wilson CI [0.73, 0.84]). Aggregating multiple images further improved accuracy, with most of the gain obtained within five images, achieving 0.91 (95% Wilson CI [0.87, 0.94]), like with ten images. Since the five-image and ten-image conditions were evaluated on the same ambiguity instances, we compared them using McNemar’s test. The test results indicated no statistically significant difference between the two conditions ($p = 1.0$). The two conditions differed in only three cases, suggesting that aggregating more than five images yielded no additional benefit in our setup.

The time for label inference decreased moderately and approximately linearly when reducing the number of images processed. For example, inference alone using a single image required 81% of the ten-image inference time; using a five-image aggregation required 89% of the ten-image inference time but already achieved the same accuracy. Thus, an accuracy of 0.91 can already be attained with five-image aggregates, each taking 2.99 s (0.85 s fixed camera activation overhead per disambiguation attempt, 0.5 s image acquisition, and 1.64 s inference) compared to 3.69 s for ten images. These results indicate that IoT data alone does not provide sufficient information for reliable disambiguation in our setup, while visual context provides substantial benefit. In our setup most of this benefit is already achieved with a five-image aggregation.

4.4 Discussion

Fulfillment of design objectives. The evaluation indicates that MMPMOD fulfills the functional design objectives (**FDO1**, **FDO2**, **FDO3**) set out in Sect. 3.1. Moreover, it indicates that the non-functional design objectives can be satisfied by implementations of the architecture. The one presented in Sect. 4 was able to operate with latencies appropriate for the domain, fulfilling **NDO1**. We could deploy MMPMOD in our preexisting lab setup without reconfiguration;

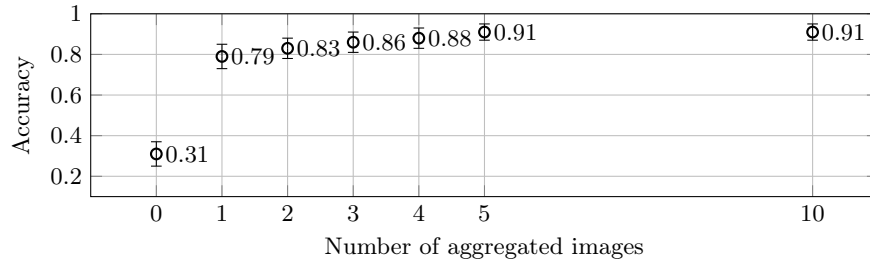


Fig. 3. Disambiguation accuracy for different temporal aggregation settings with 95% Wilson confidence intervals

the Process Event Receiver consumes the same inputs and the Process Event Emitter produces outputs in the same format as the original setup. Therefore, **NDO2** is met. **NDO3** calls for robustness and recoverability, achieved with MQTT topics serving as the only external interfaces and messages as reliable processing snapshots; forwarding events that cannot be disambiguated—due to either sensor-induced false ambiguities or the inability to determine an activity—promotes resilience. Using a downward-facing camera, we can achieve confidentiality without compromising video data integrity and availability, thus fulfilling **NDO4**. Through its modular design, the architecture also meets **NDO5**.

Transferability and generality of the architecture. Except for the Vision Model, the components of MMPMOD are domain-agnostic. The architecture is therefore designed to be extensible to domains beyond phlebotomy. Its Vision Model, however, must be domain-specific, as it requires training on domain-relevant objects. From an external interface perspective, only two channels are needed: one for ingesting process events and another for publishing disambiguated events.

Our phlebotomy scenario does not involve concurrency or parallel execution; accordingly, in MMPMOD’s implementation we rely on a single detection and resolution window. In settings that allow concurrent process instances or parallelism, the architecture can employ multiple windows, provided that events can be correlated with their respective instance or execution thread, for example via a *case ID* attribute (see Sect. 3.2). Shifting to a multi-window strategy requires changing the implementation of the *Ambiguity Analyzer* to ensure the isolated analysis of each individual window for instance- or thread-specific ambiguity detection. The mechanisms used to analyze the windows are domain-specific, since some processes may exhibit true parallelism, where multiple events can occur simultaneously in concurrent execution threads. For some processes, a minimum temporal separation between events may be guaranteed, which can be leveraged when defining the window size. Therefore, domain-specific strategies are required to isolate, correlate, and analyze events within their respective windows. Because no changes to the interface of the *Ambiguity Analyzer* are required, the remaining architectural components and their interactions remain unaffected. Overall, the architecture is equally flexible with respect to process changes and to the strate-

gies used for ambiguity detection and resolution. Alternative detection mechanisms (e.g., heuristic, rule-based) and resolution strategies (e.g., audio-based analysis) can be integrated without changing the overall control flow. Additional modalities such as audio streams can be added, with integration effort arising primarily from incorporating the respective devices and adapters, adjusting detection and resolution strategies, and introducing or training new siblings of the Vision Model. This requires the ambiguous events to be known to train the models accordingly; no changes to the interfaces between components are required. Importantly, the Process Event Router and the Process Event Emitter remain unaffected, as MMPMOD operates on process events rather than raw IoT data.

Limitations. Our evaluation focuses on a single process with three recurring ambiguities in a realistic IoT-augmented environment. This process does not exhibit concurrency, therefore we adopted a single-window ambiguity detection strategy and did not empirically evaluate multi-window strategies (cf. Sect. 3.2). While this setting allows controlled assessment of ambiguity detection and on-demand disambiguation, further studies across additional domains and more complex processes including multi-window strategies would help assess the architecture’s generality and scalability. Moreover, the evaluation focuses on how contextual sensing complements IoT-based event abstraction rather than comparing with monitoring approaches that continuously acquire video data. This design choice was deliberate and resulted in only 241 out of 1766 (13.6%) process events requiring video data acquisition, limiting the associated processing. Nevertheless, such comparisons represent an interesting direction for future work to assess the architecture’s strengths and limitations in relation to state-of-the-art approaches.

From design objectives and decisions to design principles. Reflecting on our problem and requirements analysis, our architecture design, and the evaluation of the architecture implementation, we derived five guiding principles for designing multimodal process monitoring systems with disambiguation support:

- **DP1: Selective modality activation.** *Activate additional sensing modalities when and only when needed, taking the confidence in the current modality and the detection of ambiguity into account.* This principle generalizes the on-demand activation of the Ambiguity Resolver to reduce computational overhead and latency. In MMPMOD and its implementation, a reactive, rule-based mechanism activates the vision modality upon ambiguity detection. On a more general level of knowledge, we can see this reusable design as a combination of two Enterprise Integration Patterns [12]: Selective Consumer and Content-Based Router.

- **DP2: Explicit modeling and management of ambiguity.** *Treat ambiguity as a first-class concern detected, communicated, and handled holistically.* This principle generalizes the ambiguity detection and disambiguation functionalities of the Ambiguity Detector and the On-demand Ambiguity Resolver. It emphasizes that ambiguity should not be treated merely as a technical side effect of IoT data abstraction, but as an explicit phenomenon addressed across the BPM life cycle. Current practice rarely models ambiguity explicitly; doing so provides a shared conceptual foundation for structuring detection procedures and

resolution strategies. Domain-Driven Design [7] offers one way to support this principle: the domain model can include elements for ambiguity types, detection rules, and resolution strategies, forming a Ubiquitous Language that supports consistent communication and interdisciplinary stakeholder interactions.

- **DP3: Contextualized disambiguation.** *Disambiguate by incorporating contextual information such as detected objects and human-object interactions obtained from complementary sensing modalities.* This principle generalizes the use of contextual cues, such as objects and human-object interactions, for event interpretation. It emphasizes the role of context as an enabler of higher disambiguation accuracy. Like ambiguity, contextualization should be a first-class design concern. The context dimensions should be defined as part of the requirements elicitation and domain modeling [7]. Domain-specific event patterns also require attention: recurring combinations of event sequences as well as provenance data (e.g., sensor type, configuration, and placement) typically constitute these patterns. Time series databases, event sourcing and streaming, and other event processing concepts [6] are eligible design options to detect event pattern occurrences. The Strategy pattern [10] can be leveraged to account for contextual differences, flexible event pattern detection, and resolution of ambiguities.

- **DP4: High cohesion in, loose coupling of process monitoring components.** *Structure multimodal monitoring systems with independently deployable components.* The deployability of multimodal monitoring architectures benefits from structuring the system as a set of independently deployable components, each exposing a single responsibility and interacting through loosely coupled, open interfaces. This domain-specific refinement of general software engineering principles supports maintainability and manageability by allowing components to scale, change, or be replaced independently, without impacting the rest of the system. In multimodal monitoring systems, it facilitates integration with existing infrastructures and the addition of sensing modalities. Contemporary architectural patterns such as microservices and modular monoliths, ports-and-adapters [3], and asynchronous messaging [12] make this principle actionable.

- **DP5: Value- and cost-driven prioritization of disambiguation requirements.** *Identify modality-specific conflicts between quality goals and find case-specific tradeoffs resolving them.* Tradeoffs between latency and accuracy are inherent to handling ambiguities in context, as acquiring and processing contextual information improves disambiguation accuracy but introduces additional delays. For low latency, our On-demand Ambiguity Resolver can be configured to capture fewer camera images that are processed faster but with limited accuracy; if accuracy is critical, more images can be used to disambiguate an event at the price of higher latency. Our ablation study illustrates this principle empirically: while aggregating more images improved disambiguation accuracy, most gains were achieved with five images, highlighting the need to balance latency and accuracy. Such quality goal conflicts should be identified early in any process monitoring system so that relative priorities can be negotiated, tradeoffs be found and informed decisions be made. Attribute-Driven Design and Architec-

ture Tradeoff Analysis Method [1], customized (i.e., extended or trimmed down) for the engineering of continuously changing systems, may support this principle.

Each principle abstracts a design rationale that is essential for fulfilling the functional and non-functional design objectives. Collectively, the principles constitute tangible design knowledge for crafting adaptive multimodal process monitoring systems with on-demand disambiguation, which is general enough to be applicable across domains and scenarios. The principles are detailed enough to be actionable and testable: explicit ambiguity management (**DP2**) enables selective activation (**DP1**), which employs contextualized disambiguation (**DP3**), realized within a modular architecture (**DP4**) with adequate tradeoffs (**DP5**).

5 Related Work

Multimodal process mining (MMPM) is an emerging sub-discipline that investigates the integration of data of different types—e.g., sensor, textual, visual, auditory—for a holistic process mining [11]. The proposed MMPMOD architecture can be seen as an instance of MMPM specifically focused on process monitoring, which integrates IoT sensor data and video data. A related instance is found in [22], which proposes a multimodal activity recognition approach to process discovery. Similar to MMPMOD, it explicitly addresses ambiguity introduced with event abstraction. Unlike our on-demand architecture, it continuously acquires video data, requiring dedicated resources for video analysis. Moreover, it performs full-frame rather than semantics-informed classification. A similar approach to detect activities from pixel values in specific regions of camera images without extracting semantic information is found in [15]. Another instance of MMPM is the pipeline for activity recognition in robotic systems proposed in [4]. It fuses visual and auditory robotic data with foundational models by leveraging the models’ zero-shot reasoning capabilities. While zero-shot reasoning alleviates the need to train a model—which is a prerequisite in our prototype—, it is currently challenging to leverage foundational models beyond offline settings: further advancements of these models, however, could enable online applications.

Beyond MMPM, process mining has successfully leveraged video data analysis before, for instance for conformance checking [2] and behavioral pattern discovery [18]. Unlike approaches that derive process events directly from video data [11,18], MMPMOD assumes IoT-based event abstraction and uses computer vision only as contextual evidence for disambiguating candidate events. This makes direct quantitative comparisons difficult, as these systems address different monitoring assumptions and data acquisition strategies. In [17], a reference architecture (ViProMiRA) has been proposed to extract process information from video data, thereby enabling post-mortem analysis tasks such as conformance checking and process discovery. While ViProMiRA leverages supervised learning, the more recent RAVEE [25] advances the former by enabling extracting process information from video data in an unsupervised setting. These advancements offer an opportunity to alleviate the effort to train a fine-tuned model in our prototype. More generally, along with other work, e.g., [4,16], they

point to the opportunity to integrate our MMPMOD architecture with models able to reliably analyze data from additional modalities without specific training.

6 Conclusion

We proposed MMPMOD as an objectives-driven component architecture for multimodal process monitoring with on-demand disambiguation. Its instantiation in a phlebotomy scenario demonstrated the feasibility of orchestrating complementary sensing modalities for disambiguation in online monitoring. Further investigation of the latency-disambiguation accuracy tradeoff remains a future research opportunity. While our vision model proved reliable in detecting objects and interactions, we plan to further enhance its detection capabilities. Finally, although we instantiated MMPMOD in a single scenario, its modular design suggests potential applicability beyond phlebotomy and may serve as a design blueprint for similar systems, which should be validated in future work. Our five generalized, actionable design principles support such reuse.

Acknowledgments. This work has received funding from the Swiss National Science Foundation under Grant No. IZSTZ0_208497 and under Grant No. 10002384.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, Fourth Edition. Addison-Wesley Professional (2021)
2. Chen, S., Zou, M., Cao, R., Zhao, Z., Zeng, Q.: Video process mining and model matching for intelligent development: Conformance checking. *Sensors* **23**(8), 3812 (2023)
3. Cockburn, A., de Paz, J.M.G.: Hexagonal Architecture Explained: How the ports and adapters architecture simplifies your life, and how to implement it, Updated 1st Ed. Humans and Technology Inc (2025)
4. Corradini, F., Pettinari, S., Re, B., Rossi, L., Sampaolo, M.: Multimodal zero-shot activity recognition for process mining of robotic systems. In: International Conference on Business Process Management. pp. 263–275. Springer (2025)
5. van Der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. *Distributed and parallel databases* **14**(1), 5–51 (2003)
6. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications Co., USA, 1st edn. (2010)
7. Evans, E.: Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Longman Publishing Co., Inc., USA (2003)
8. Franceschetti, M., Seiger, R., García-Bañuelos, L., Weber, B.: Proambition, reloaded: A two-year retrospection. Research Projects Exhibition@ CAiSE 2025 (2025)
9. Franceschetti, M., Seiger, R., López, H.A., Burattin, A., García-Bañuelos, L., Weber, B.: A characterisation of ambiguity in BPM. In: International Conference on Conceptual Modeling. pp. 277–295. Springer (2023)

10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman, USA (1995)
11. Gavric, A., Bork, D., Proper, H.: Multimodal process mining. In: 2024 26th International Conference on Business Informatics (CBI). pp. 99–108. IEEE (2024)
12. Hohpe, G., Woolf, B.: Enterprise integration patterns: designing, building, and deploying messaging Solutions. Addison-Wesley Longman Publishing (2003)
13. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., et al.: The internet of things meets business process management: a manifesto. *IEEE Systems, Man, and Cybernetics Magazine* **6**(4), 34–44 (2020)
14. Jimenez Cruz, R., González González, M.J., Torres-Huitzil, C., Franceschetti, M., Seiger, R., García-Bañuelos, L., et al.: Labeled images of a simulated phlebotomy procedure on a training arm (2025). <https://doi.org/10.5281/zenodo.16924786>
15. Knoch, S., Ponpathirkootam, S., Fettke, P., Loos, P.: Technology-enhanced process elicitation of worker activities in manufacturing. In: International Conference on Business Process Management. pp. 273–284. Springer (2017)
16. Knoch, S., Ponpathirkootam, S., Schwartz, T.: Video-to-model: Unsupervised trace extraction from videos for process discovery and conformance checking in manual assembly. In: International Conference on Business Process Management. pp. 291–308. Springer (2020)
17. Kratsch, W., König, F., Röglinger, M.: Shedding light on blind spots—developing a reference architecture to leverage video data for process mining. *Decision Support Systems* **158**, 113794 (2022)
18. Lepsien, A., Koschmider, A., Kratsch, W.: Analytics pipeline for process mining on video data. In: International Conference on Business Process Management. pp. 196–213. Springer (2023)
19. Mangler, J., Seiger, R., Benzin, J.V., Grüger, J., Kirikkayis, Y., Gallik, F., et al.: From internet of things data to business processes: Challenges and a framework. In: Internet of Things Meets Business Process Management: A Synergistic Integration, pp. 25–61. Springer (2026)
20. Metzner, C., Schilling, A., Traxdorf, M., Tziridis, K., Maier, A., Schulze, H., Krauss, P.: Classification at the accuracy limit: facing the problem of data ambiguity. *Scientific reports* **12**(1), 22121 (2022)
21. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. *Journal of management information systems* **24**(3), 45–77 (2007)
22. Rebmann, A., Emrich, A., Fettke, P.: Enabling the discovery of manual processes using a multi-modal activity recognition approach. In: International Conference on Business Process Management. pp. 130–141. Springer (2019)
23. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (2016)
24. Seiger, R., Kurz, A.F., Franceschetti, M.: Online detection of process activity executions from iot sensors using generated event processing services. *Future Generation Computer Systems* **174**, 107987 (2026)
25. Wördehoff, N., Egger, A., Kratsch, W., König, F., Röglinger, M.: Beyond assumptions: A reference architecture to enable unsupervised process discovery from video data. *Decision Support Systems* p. 114544 (2025)
26. Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., Rehg, J.M.: A scalable approach to activity recognition based on object use. In: Int. conf. on computer vision. pp. 1–8. IEEE (2007)