Toward Self-Modifying Autonomous Business Process Systems

Achiya Elyasaf[1], Andreas Metzger[2], Sebastian Sardina[3], Arik Senderovich[4], Estefanía Serral Asensio[5] and Niek Tax[6]

[1]Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel
[2]paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen, Germany
[3]School of Computing Technologies, RMIT University, Melbourne, Australia
[4]School of Information Technology, York University, Toronto, Canada
[5]Research Centre for Information Systems Engineering (LIRIS), KU Leuven, Belgium
[6]Meta, London, United Kingdom

Abstract

Autonomous business process systems (ABPS) operate with minimal human intervention while modifying themselves over time. Being 'self-modifying' thereby becomes an intrinsic characteristic that enables a process to become autonomous. In this paper, we define what it means to be self-modifying in autonomous business process systems (ABPS) and outline key characteristics of such capability, including (1) the differentiation between two types of modifications, namely adaptation and evolution, (2) the levels of granularity in which a modification can be carried out, and (3) the reasons why a possible modification may be triggered. After presenting a motivating example, we characterize the goals for advancing self-modifying capabilities in ABPSs, and identify open research challenges, particularly around governance, uncertainty, and continuous learning. This work aims to structure the problem space of self-modification in ABPSs and provide a conceptual foundation for future research and development.

Keywords

Autonomous Business Process Systems (ABPS), Self-Modifying Processes, Adaptation, Evolution

1. Introduction

In an increasingly dynamic and complex digital landscape, business processes (BPs) are expected to operate with minimal human intervention while modifying their own structure and behavior to meet changing goals, new environment conditions, or novel constraints [1]. We refer to such systems as autonomous *business process systems* (ABPSs). An core capability of ABPSs is self-modification.

The notion of ABPSs was elaborated during the 2025 AutoBiz Dagstuhl seminar.[1] The main goal of this seminar was to compile a research agenda toward the realization of ABPSs. Jointly with the seminar participants, we discussed and developed core concepts, challenges, and research directions. Specifically, after a series of stimulating talks by experts, participants split into working groups to further discuss individual topics of the research agenda, including "framed autonomy", "self-modification", "conversational actionability", and "explainability". The results of these breakout-groups were presented to all seminar participants, and their feedback was used to improve the findings. This paper reports on key findings concerning the topic of "self-modification".

PMAI 2025: 4th International Workshop on Process Management in Artificial Intelligence, co-located with ECAI 2025, October 25
achiya@bgu.ac.il (A. Elyasaf); andreas.metzger@paluno.uni-due.de (A. Metzger); sebastian.sardina@rmit.edu.au (S. Sardina); sariks@yorku.ca (A. Senderovich); estefania.serral@kuleuven.be (E. S. Asensio); niek@meta.com (N. Tax)
https://achiya.elyasaf.net/ (A. Elyasaf); https://www.paluno.uni-due.de/en/team/andreas-metzger/ (A. Metzger); https://ssardina.github.io/ (S. Sardina); https://www.ariksenderovich.com/ (A. Senderovich); https://www.kuleuven.be/wieiswie/en/person/00095631 (E. S. Asensio); https://openreview.net/profile?id=~Niek_Tax1 (N. Tax)
0000-0002-4009-5353 (A. Elyasaf); 0000-0001-5629-2477 (A. Metzger); 0000-0003-2962-0118 (S. Sardina); 0000-0003-4728-8024 (A. Senderovich); 0000-0001-7579-910X (E. S. Asensio); 0000-0001-7239-5206 (N. Tax)

[1]See https://www.dagstuhl.de/25192. We express our gratitude to the Scientific Directorate and staff of Schloss Dagstuhl for their invaluable support. We also thank our fellow participants for their engaging discussions.

Fundamentally, self-modifications in ABPSs may be classified into [2]:

- *Adaptation*, which refers to short-term, instance-specific modifications that allow the system to handle unforeseen conditions typically without altering the underlying process schema. These modifications are often triggered in real-time and involve localized decisions. The goal is to ensure continuity and resilience without impacting the design of future process instances. Examples for adaptations include altering the running workflow (such as rerouting, skipping tasks, or requesting human intervention), reallocating resources, and integrating new data sources or services at runtime for a specific (set of) process instance(s).
- *Evolution*, which represents a long-term modification of the process logic and/or model itself that therefore affects all future instances of the process. These modifications are typically informed by patterns observed over time, whether through repeated failures, performance bottlenecks, or shifts in business strategy. Evolution is deliberate and designed to improve future process executions, often requiring model redesign, policy updates, or system reconfiguration. Examples for evolution include revising and redesigning the decision logic and underlying BP models.

Without self-adaptation and self-evolution capabilities, ABPSs risk stagnation, brittleness, and reduced responsiveness in volatile environments. These limitations can compromise process autonomy. Therefore, understanding and engineering self-modifying capabilities is essential for advancing the design, implementation, and governance of ABPSs. The need for BPM systems to be flexible and modify themselves in response to changes has long been recognized in the BPM community [3, 2, 4]. Various works study the adaptation [5, 6, 7, 8, 9, 10], and evolution of BPs [11, 12, 13].
The contributions of this paper are conceptual and are as follows:

- We define the notion of *self-modification* in autonomous business process systems (ABPSs), distinguishing between two key types: *adaptation* and *evolution*.
- We identify and organize the core dimensions, goals, and triggers of self-modification, thereby structuring the problem space.
- We outline key research challenges – particularly around governance, uncertainty, and continuous learning – that must be addressed to advance self-modifying ABPSs.

2. Running Example: Automated Warehouse

To ground our exploration of self-modifying ABPSs, we begin with a running example inspired by real-world operations at a large company's automated warehouses. These warehouses deploy fleets of robots to retrieve and transport shelves (pods) to human pickers, coordinating complex logistics across a dynamic, high-throughput environment.

Suppose a robot malfunctions during the Christmas holiday season, one of the busiest times of the year. Nearby robots quickly reroute their paths to avoid the blocked aisle, and the pending order is reassigned to a functioning robot. Human workers on the floor receive updated instructions via handheld devices, ensuring minimal disruption to order fulfillment. This represents a short-term, operational-level modification (i.e., an adaptation) that addresses the immediate issue without altering the overall process model.

However, the system does not stop there. It logs the failure event and flags it for review. Upon inspection, engineers discover a pattern: similar failures tend to occur after approximately 1,000 picks. In response, a new maintenance rule is introduced, mandating preemptive inspection after every 900 operations. This is a tactical, mid-term modification (i.e., an evolution) – a refinement of the rules and policies guiding robot operations, aimed at reducing future risk.

This illustrates a spectrum of ABPS modifications – ranging from instance-specific adaptations to model-level evolution affecting all future instances – captured and enacted with varying degrees of automation.

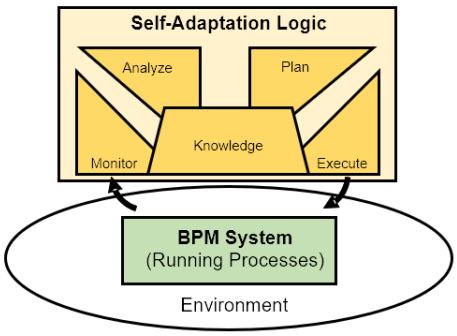

Figure 1: Conceptual model of self-adaption.

3. Types of Modifications in ABPs

Modifications in ABPSs can be classified along several dimensions:

D1: Adaptation vs. Evolution. We introduced this central dimension already in Section 1, referring to adaptations as short-term, instance-specific modifications, and evolution as the long-term, multi-instance modification of the process logic and/or model itself [14].

The concept of adaptation in ABPSs strongly connects to work of the software engineering community on self-adaptive *software* systems [15]. Here, the MAPE-K loop has established itself as a widely adopted conceptual model [16, 17]. This model is depicted in Fig. 1.

The self-adaptation logic is structured into four main conceptual activities (MAPE) that leverage a common knowledge base (K). The knowledge base includes adaptation goals (requirements), adaptation strategies, or adaptation rules. The four activities are concerned with *monitoring* the system and the system's environment via sensors, *analyzing* the monitoring data to determine the need for an adaptation, *planning* adaptation actions, and *executing* these adaptation actions via actuators, thereby modifying the system at run time [9].

The concept of evolution in ABPSs, in contrast, focuses on systematic, long-term modifications that affect the process model and logic across multiple future instances. It is closely related to work on software and process evolution [18, 14]. Rather than responding to immediate runtime conditions, evolution is driven by aggregated insights from monitoring and analysis over time – such as recurring performance issues, changes in strategic direction, or compliance updates. These insights are used to deliberately revise the process design, typically through a structured approach involving *assessment* of existing process outcomes, *planning* of structural or behavioral improvements, *implementation* of these changes in the model, and *validation* to ensure conformance with long-term goals. While adaptation enables resilience and responsiveness, evolution ensures strategic alignment, sustainability, and long-term optimization of ABPSs.

D2: Task vs. Flow vs. Process. Another critical dimension concerns *what is modified*. Drawing from the business process redesign literature [14], we distinguish between:

- **Task-level modifications**, which modify how individual tasks are performed or configured (e.g., modifying a task's duration, logic, input/output data, or resource assignment).
- **Control Flow-level modifications**, which adjust the control flow or routing of tasks (e.g., rerouting orders or skipping tasks).
- **Process-level modifications**, which alter the structure or central resources of the entire process (e.g., introducing new roles, shifting coordination logic, or replacing subsystems).

Note that these levels often interact. For instance, rerouting in the warehouse involves flow-level change, while a new rule for reassigning malfunctioning robots might affect task performance, and revising the robot maintenance schedule constitutes a process-level change.

This dimension provides additional granularity for analyzing the scope and impact of a modification. It also affects the complexity of implementation: task-level changes can often be handled locally, while process-level changes typically require coordination and propagation across components.

D3: Reactive vs. Proactive. Another important distinction concerns the trigger of the modification. Reactive modifications occur in response to specific events or failures, such as a robot malfunction triggering rerouting. Proactive modifications, on the other hand, are initiated based on forecasts or insights derived from predictive analytics. Examples include predictive and prescriptive business process monitoring [19, 20]; e.g., scheduling preventive maintenance based on predicted failure likelihood or preventing robots from entering certain aisles at load times.

D4: Human-Driven vs. Autonomous. Modifictions may be either 1) human-driven, where users detect, decide, and implement modifications; or 2) autonomous, where the system identifies issues and enacts modifications on its own. In our running example, the creation of the new maintenance rule might be initially human-driven, but an autonomous ABPS could eventually learn such patterns and implement similar modifications independently.

D5: Planned vs. Emergent. Planned modifications stem from deliberate, top-down redesign, such as rolling out a new maintenance policy across sites. Emergent modifications arise bottom-up, realized as patterns learned from ongoing process adaptations (e.g., see [19]) or eventual model evolutions. An ABPS that generalizes from local failure logs to propose global improvements exemplifies this emergent capability.

Characterizing the running example according to the described dimensions. All the key dimensions of modifications play out in the warehouse scenario. When robots reroute around a malfunctioning unit, the system performs a short-term, instance-level adjustment – an adaptation. No process model is modified; the system reacts in real time to maintain flow. By contrast, introducing a maintenance rule after repeated failures illustrates evolution – a deliberate, model-level change based on aggregated insights.

Rerouting is reactive when triggered by a specific event. However, it can also be proactive if it is triggered in advanced using predictive monitoring: it anticipates issues and acts in advance. Both forms of change can coexist, occurring within the same process window.

Initially, engineers manually define the maintenance policy – making it human-driven and planned. But an advanced ABPS could learn the same pattern, propose the rule, and apply it autonomously. If the rule itself later evolves based on outcomes through, e.g., reinforcement learning, the change becomes emergent – bottom-up and data-driven.

4. Toward Self-modifying ABPSs

Today's business process systems typically operate at an *augmented* level of autonomy, where intelligent components assist human workers but do not independently drive process execution or change. To move toward truly autonomous business process systems (ABPSs), we propose a structured roadmap of autonomy levels, inspired by the SAE J3016 standard for driving automation [21]. While Sheridan's Levels of Automation [22] offers an alternative, they focus on isolated task automation rather than holistic process-level behavior, making them less suitable for ABPSs.

We define autonomy levels as follows:

- **Level 0: No Automation** – Execution and orchestration of all tasks are fully manual.
- **Level 1: Process Assistance** – The system provides recommendations or highlights anomalies to human workers (e.g., predictive monitoring [23]).

Table 1
Goals and capabilities for autonomy at different levels and modification objects.

Object of modification	Level 1: Process Assistance	Level 2: Partial Autonomy	Level 3: Contextual Autonomy
Task	Recommending task performers or configurations (e.g., duration, cost).	Automating task assignment or automated execution in narrow scopes.	ABPS decides and executes full task reconfiguration autonomously with human fallback only in edge cases.
Flow	Suggesting alternative paths or detecting bottlenecks.	Automating routing based on real-time conditions or rules.	Rerouting and dynamically altering execution paths with learned policies under uncertainty.
Process	Flagging process-wide issues (e.g., coordination delays).	Automating subprocesses, e.g., resource pooling or exception handling.	Reconfiguring processes, modifying policies or goals, and coordinating across stakeholders with minimal human oversight.

- **Level 2: Partial Autonomy** – The system independently executes isolated tasks (e.g., call routing, task assignment) within predefined boundaries, but without contextual or adaptive behavior.
- **Level 3: Contextual Autonomy** – The system autonomously performs most tasks and orchestrates flows in a context-aware manner, requiring human intervention only in exceptional cases (which closely relates to agentic BPM [24]).

While most current BPM systems reside at Level 1 or 2, we envision ABPSs to reside at Level 3. Achieving this level requires the development of self-modifying capabilities. Specifically, a Level 3 ABPS must:

1. Detect changes in the operating environment (e.g., *concept drift detection* [25, 26, 27]).
2. Decide whether the detected changes require *adaptation* or *evolution*.
3. Select an appropriate modification strategy based on goals, context, and system history.
4. Learn from prior adaptations and generalize from successful strategies.
5. Communicate and explain its decisions, rationale, and uncertainty to human stakeholders.

The extent and nature of these capabilities depend on the *object of modification* (task, flow, or process) as presented in Table 4. For instance, at Level 1, task-level modifications might involve recommending better parameter configurations; at Level 3, the system may autonomously reassign or skip tasks. Similarly, flow-level autonomy ranges from highlighting bottlenecks (Level 1) to real-time rerouting (Level 3), while process-level autonomy progresses from alerting on coordination issues to full reconfiguration of goals, roles, or policies.

For example, in a warehouse scenario, task-level changes may involve altering how picking or navigation tasks are performed; flow-level changes may include rerouting due to blocked paths; and process-level autonomy could entail adjusting global maintenance policies. Thus, progressing toward Level 3 autonomy requires integrating sensing, reasoning, learning, and communication across abstraction levels, while minimizing reliance on human oversight.

5. Challenges in Enabling Autonomous Modifications

In this section, we discuss three challenges related to governance and human oversight, continual learning for adaptation management, and uncertainty quantification and communication.

Challenge 1: Governance, Oversight, and Human Interaction. For ABPSs to operate safely and effectively, governance and human interaction must be built into their design. A key question is when to refrain from automation and return control to a human process worker, e.g., in scenarios where the agent has insufficient confidence and is at risk of making mistakes. This is particularly relevant in ambiguous or high-risk situations. Research in AI planning, runtime monitoring may help establish thresholds or confidence bounds beyond which a process must escalate to human decision-makers. Techniques from the ML literature on prediction with reject option [28] (sometimes called "learning to defer") may also be explored as a solution direction.

Similarly, determining when and how a user should validate a proposed process or policy modification requires a framework for explainable adaptation, where the system presents justifications for its modifications. Solution directions may include methods from the field of explainable AI (XAI) [29], or justifications may be provided in alternative forms such as simulations of expected outcomes.

Another major issue is aligning the goals of an ABPs with those of its human operators and organizational stakeholders. Multi-objective optimization techniques [30] can assist in balancing trade-offs between performance, cost, compliance, and user satisfaction while maintaining logical constraints defined in the process model. However, optimizing across conflicting objectives remains a hard problem, especially when human values or non-quantifiable criteria are involved.

Quality assurance in fully autonomous scenarios introduces its own challenges. Without humans routinely checking outcomes, ABPSs must develop internal mechanisms for evaluating whether an adaptation "worked." Here, ML-based anomaly detection, performance baselining, and causal reasoning can help detect regressions or misbehavior. Formal methods for verification and validation of modifying process logic or agent policies also present a relevant research direction here. LLMs could also be used for generating justifications or summaries of decisions for human audits or by providing labels for downstream evaluation – an emerging area sometimes called LLM-as-a-judge [31].

Solutions to the quality assurance problem could use human input purely for evaluation and quality assurance, even in scenarios where the execution is fully automated. Human input is costly, and hence a challenge is to make this cost-efficient. E.g., through methods like active testing [32].

Finally, understanding what data is needed to make ABPSs reliable is a key frontier – systems must know whether they have "enough" context to act or whether to defer or collect more information.

Core Research Questions:

- When should control shift between autonomous processes and humans in ABPSs?
- How can user validation be incorporated into real-time adaptation without bottle-necking autonomy of ABPSs?
- How can ABPSs optimize multiple objectives while remaining within formal and ethical constraints?
- How can ABPSs evaluate the success or failure of modifications when human validation is unavailable?
- What are the data quality and coverage thresholds for safe, autonomous decision-making in ABPSs?

Challenge 2: Continuous Learning and Adaptation Management. A defining feature of self-modifying ABPSs is their ability to learn from experience and improve process behavior over time. To support this, the system must continuously record the adaptations it makes – whether reactive or proactive – and assess their impact both locally (per instance) and globally (across instances). Capturing this meta-knowledge creates a feedback loop where successful modifications reinforce future decisions and ineffective ones are pruned. AI planning and reinforcement learning techniques (while balancing exploration and exploitation; e.g., see [10]) are promising approaches to structuring such learning loops, especially when enriched by causal modeling of intervention outcomes.

Evaluating generalization in system behavior poses a second challenge. It is not enough for an adaptation to work well in one instance; the ABPSs must infer under what conditions it will work again. This raises questions around how process context is encoded and how confidence in the modification is

calibrated. For instance, if a task is repeatedly skipped under certain workload conditions, the system may learn a policy for skipping it when similar signals appear – but must be careful not to overgeneralize. Formal guarantees and empirical validation frameworks are needed to ensure safe generalization. This problem intersects with interpretability and trust, as human stakeholders may require explanations for the shifts in system behavior.

Finally, long-running processes or high-volume ABPSs face constraints of bounded memory and context. The system cannot retain or process each event ever observed. Hence, it must learn to construct and maintain bounded knowledge representations: compressed summaries, predictive state abstractions, or fixed-size windows of relevant execution history. Techniques from stream reasoning, process mining over sliding windows, and transformer-based sequence models may offer solutions. Designing an ABPS that knows which parts of its history to remember, forget, or query becomes a core challenge for sustainable, real-time continuous learning.

Core Research Questions:

- How can ABPSs continuously record adaptations and assess their effectiveness over time?
- What metrics and techniques enable an ABPS to safely generalize learned behavior across varying contexts?
- How can an ABPs maintain bounded knowledge representations for long-running or high-frequency processes?

Challenge 3: Modeling and Measuring Uncertainty. ABPSs must operate under both aleatoric (inherent randomness) and epistemic (lack of knowledge) uncertainty [33]. Addressing these uncertaingties requires both awareness and expressiveness. Aleatoric uncertainty can be modeled through probabilistic distributions over process durations, outcomes, or resources. Epistemic uncertainty, however, often requires exploration or targeted sensing to reduce ambiguity. Knowing which type of uncertainty is present affects which mitigation strategies are appropriate – whether to gather more data or to hedge decisions probabilistically. Techniques from probabilistic modeling, Bayesian inference, and fuzzy logic are key for representing and reasoning about uncertainty in process execution.

Quantifying uncertainty involves both qualitative and quantitative measures. In some domains, thresholds or confidence levels may be set by human experts (qualitative), while in others, Bayesian models or statistical metrics (quantitative) provide actionable measures. As an example, reliability estimates for proactive adaptation may be derived from ensembles of prediction models [9]. ABPSs must combine these forms of knowledge, learning from past executions when quantitative models are feasible, while falling back on qualitative heuristics when data is sparse or ambiguous. Hybrid methods that integrate fuzzy logic or ensemble learning could further improve uncertainty modeling in domains with imprecise information.

Quantifying uncertainty strongly connects to both challenges 1 and 2. For challenge 1, deciding when to defer the decision-making back to a human operator is a classic task where most solutions involve uncertainty quantification [28]. Regarding challenge 2, it is well known from the literature on uncertainty quantification and active learning that accurate estimates of specifically the epistemic uncertainty are a prerequisite to learn and adapt policies to new environments in a data-efficient way [33, 34, 35].

A final challenge is communicating uncertainty effectively to human stakeholders. Especially in semi-autonomous systems, operators must understand not only what the system plans to do, but also how confident it is in that choice. As pointed out by Miller "probabilities probably don't matter" [36]. This means uncertainty quantification should be augmented with explainable AI (XAI) techniques (e.g., [37]). Visualized confidence intervals, and natural language generation via LLMs may help bridge this gap (e.g., as done for adaptive software systems [38]). Additionally, representing the impact of uncertainty on downstream outcomes – such as potential delays, risks, or degraded service quality – will help human decision-makers intervene appropriately.

Core Research Questions:

- How can ABPs differentiate between epistemic and aleatoric uncertainty during execution?

Table 2
Challenge mapping across different levels of modification granularity.

Object of modification	Challenge 1: Governance, Oversight, and Human Interaction	Challenge 2: Continuous Learning and Adaptation Management	Challenge 3: Modeling and Measuring Uncertainty
Task	When should a task's execution or performer be modified manually vs. autonomously?	How can the system learn better ways to perform tasks over time?	How confident is the system in altering task parameters or execution strategy?
Flow	Who approves new routing logic or control decisions?	Can routing rules be generalized across similar cases without overfitting?	What are the risk profiles of rerouting or skipping tasks?
Process	When must a human approve structural redesign (e.g., policy updates)?	How does the system record, assess, and evolve high-level process modifications?	How can uncertainty in aggregated outcomes be modeled across multiple processes?

- How can ABPSs combine qualitative and quantitative uncertainty metrics for robust decision-making?
- What are effective representations of uncertainty (e.g., using probabilistic or fuzzy paradigms)?
- How should ABPSs communicate uncertainty and associated risk to users in a transparent and actionable way?

Challenges and object of modification. Like in Section 3, extent and relevance of the aforementioned challenges depend on the *object of modification* (task, flow, or process), which is presented in Table 5.

6. Conclusion and Outlook

In this paper, we have laid the conceptual groundwork for understanding and engineering self-modifying capabilities in autonomous business process systems (ABPSs). We defined what it means for an ABPS to self-modify, distinguishing between the short-term reactivity of adaptation and the long-term reconfiguration of evolution, and introduced a structured framework for levels of business process autonomy. We further mapped these autonomy levels across the different objects of modification – task, flow, and process – and connected them to system goals and capabilities.

As ABPSs strive toward higher levels of autonomy, we identified three core research challenges that must be addressed to enable safe, intelligent, and human-aware self-modification: (1) establishing robust governance mechanisms and human oversight; (2) managing continuous learning to support sustainable and generalizable adaptations; and (3) modeling and communicating uncertainty in ways that foster trust and informed decision-making.

Together, these challenges highlight a critical insight: autonomy is not simply about replacing humans but about redesigning systems that can responsibly decide when to act, when to adapt, and when to defer. The path forward requires integrating techniques from AI planning, machine learning, explainable AI, adaptive software systems, causal inference, process mining, and human-computer interaction into coherent architectures that balance autonomy with accountability.

We envision ABPSs of the future not as black-box automation engines but as collaborative agents capable of engaging with their environments and human counterparts in a transparent, contextual, and goal-aligned manner. To that end, we encourage the community to build benchmarks, share evaluation frameworks, and develop modular toolkits that bring us closer to truly self-modifying, trustworthy, and adaptive business processes.

Declaration on Generative AI

During the preparation of this work, the authors used OpenAI Deep Research and Google Gemini to: Complement our manual literature analysis of related work, Grammar and spelling check, Paraphrase and reword, Improve writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Dumas, F. Fournier, L. Limonad, A. Marrella, M. Montali, J.-R. Rehse, R. Accorsi, D. Calvanese, G. De Giacomo, D. Fahland, et al., Ai-augmented business process management systems: a research manifesto, ACM Transactions on Management Information Systems 14 (2023) 1–19.

[2] M. Reichert, B. Weber, Enabling flexibility in process-aware information systems: challenges, methods, technologies, volume 54, Springer, 2012.

[3] P. Dadam, M. Reichert, The adept project: a decade of research and development for robust and flexible process support: Challenges and achievements, Computer Science-Research and Development 23 (2009) 81–97.

[4] N. R. Jennings, P. Faratin, M. Johnson, T. J. Norman, P. O'brien, M. E. Wiegand, Agent-based business process management, International Journal of Cooperative Information Systems 5 (1996) 105–130.

[5] K. Jander, L. Braubach, A. Pokahr, W. Lamersdorf, K.-J. Wack, Goal-oriented processes with GPMN, International Journal on Artificial Intelligence Tools 20 (2011) 1021–1041.

[6] D. Greenwood, R. Ghizzioli, Goal-oriented autonomic business process modelling and execution, in: Multiagent Systems, IntechOpen, 2009.

[7] L. Sabatucci, C. Lodato, S. Lopes, M. Cossentino, Towards self-adaptation and evolution in business process., in: Aibp@ ai* ia, 2013, pp. 1–10.

[8] E. Serral, J. De Smedt, M. Snoeck, J. Vanthienen, Context-adaptive petri nets: Supporting adaptation for the execution context, Expert Systems with Applications 42 (2015) 9307–9317.

[9] A. Metzger, T. Kley, A. Palm, Triggering proactive business process adaptations via online reinforcement learning, in: D. Fahland, C. Ghidini, J. Becker, M. Dumas (Eds.), Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings, volume 12168 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 273–290. URL: https://doi.org/10.1007/978-3-030-58666-9_16. doi:10.1007/978-3-030-58666-9_16.

[10] A. Palm, A. Metzger, K. Pohl, Online reinforcement learning for self-adaptive information systems, in: S. Dustdar, E. Yu, C. Salinesi, D. Rieu, V. Pant (Eds.), Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings, volume 12127 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 169–184. URL: https://doi.org/10.1007/978-3-030-49435-3_11. doi:10.1007/978-3-030-49435-3_11.

[11] H. Kir, N. Erdogan, A knowledge-intensive adaptive business process management framework, Inf. Syst. 95 (2021) 101639. doi:10.1016/J.IS.2020.101639.

[12] E. Serral, P. Valderas, V. Pelechano, Addressing the evolution of automated user behaviour patterns by runtime model interpretation, Software & Systems Modeling 14 (2015) 1387–1420.

[13] E. Serral, P. Valderas, V. Pelechano, Supporting runtime system evolution to adapt to user behaviour, in: Advanced Information Systems Engineering: 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings 22, Springer, 2010, pp. 378–392.

[14] H. A. Reijers, Process design and redesign, Process-Aware Information Systems: Bridging People and Software through Process Technology (2005) 205–234.

[15] D. Weyns, An introduction to self-adaptive systems: A contemporary software engineering perspective, John Wiley & Sons, 2020.

[16] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè,

M. Shaw, Engineering self-adaptive systems through feedback loops, in: Software engineering for self-adaptive systems, Springer, 2009, pp. 48–70.
[17] A. Computing, et al., An architectural blueprint for autonomic computing, IBM White Paper 31 (2006) 1–6.
[18] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, W. M. Turski, Metrics and laws of software evolution – the nineties view, in: Proceedings of the Fourth International Software Metrics Symposium (METRICS '97), IEEE Computer Society, 1997, pp. 20–32. URL: https://doi.org/10.1109/METRIC.1997.637156. doi:10.1109/METRIC.1997.637156.
[19] A. Metzger, T. Kley, A. Rothweiler, K. Pohl, Automatically reconciling the trade-off between prediction accuracy and earliness in prescriptive business process monitoring, Inf. Syst. 118 (2023) 102254. URL: https://doi.org/10.1016/j.is.2023.102254. doi:10.1016/J.IS.2023.102254.
[20] Z. D. Bozorgi, I. Teinemaa, M. Dumas, M. L. Rosa, A. Polyvyanyy, Prescriptive process monitoring based on causal effect estimation, Inf. Syst. 116 (2023) 102198. URL: https://doi.org/10.1016/j.is.2023.102198. doi:10.1016/J.IS.2023.102198.
[21] SAE International, Taxonomy and Definitions for Driving Automation Systems for On-Road Motor Vehicles, SAE Recommended Practice J3016, 2021.
[22] T. B. Sheridan, Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: Distinctions and modes of adaptation, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 41 (2011) 662–667.
[23] A. E. Márquez-Chamorro, M. Resinas, A. Ruiz-Cortés, Predictive monitoring of business processes: a survey, IEEE Transactions on Services Computing 11 (2017) 962–977.
[24] H. Vu, N. Klievtsova, H. Leopold, S. Rinderle-Ma, T. Kampik, Agentic business process management: Practitioner perspectives on agent governance in business processes, 2025. URL: https://arxiv.org/abs/2504.03693. arXiv:2504.03693.
[25] F. Hinder, V. Vaquet, B. Hammer, One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift, Frontiers in Artificial Intelligence 7 (2024) 1330257.
[26] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE transactions on knowledge and data engineering 31 (2018) 2346–2363.
[27] D. M. V. Sato, S. C. De Freitas, J. P. Barddal, E. E. Scalabrin, A survey on concept drift in process mining, ACM Computing Surveys (CSUR) 54 (2021) 1–38.
[28] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, J. Davis, Machine learning with a reject option: A survey, Machine Learning 113 (2024) 3073–3110.
[29] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan, et al., Explainable ai (xai): Core ideas, techniques, and solutions, ACM Computing Surveys 55 (2023) 1–33.
[30] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering, Structural and multidisciplinary optimization 26 (2004) 369–395.
[31] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., Judging llm-as-a-judge with mt-bench and chatbot arena, Advances in Neural Information Processing Systems 36 (2023) 46595–46623.
[32] J. Kossen, S. Farquhar, Y. Gal, T. Rainforth, Active testing: Sample-efficient model evaluation, in: International Conference on Machine Learning, PMLR, 2021, pp. 5753–5763.
[33] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, Machine learning 110 (2021) 457–506.
[34] V.-L. Nguyen, M. H. Shaker, E. Hüllermeier, How to measure uncertainty in uncertainty sampling for active learning, Machine Learning 111 (2022) 89–122.
[35] A. Tharwat, W. Schenck, A survey on active learning: State-of-the-art, practical challenges and research directions, Mathematics 11 (2023) 820.
[36] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artif. Intell. 267 (2019) 1–38. URL: https://doi.org/10.1016/j.artint.2018.07.007. doi:10.1016/J.ARTINT.2018.07.007.

[37] D. Watson, J. O'Hara, N. Tax, R. Mudd, I. Guy, Explaining predictive uncertainty with information theoretic shapley values, Advances in Neural Information Processing Systems 36 (2023) 7330–7350.
[38] A. Metzger, J. Laufer, F. Feit, K. Pohl, A user study on explainable online reinforcement learning for adaptive systems, ACM Trans. Auton. Adapt. Syst. 19 (2024) 15:1–15:44. URL: https://doi.org/10.1145/3666005. doi:10.1145/3666005.