(IJRST) 2022, Vol. No. 12, Issue No. III, Jul-Sep **e-ISSN: 2249-0604, p-ISSN: 2454-180X**

Planning Failure Recovery Strategies using Artificial Intelligence in Discrete Manufacturing Automation[1]

Rishi Ahuja
New Delhi, India

DOI:10.37648/ijrst.v12i03.007

Received: 27 August 2022; Accepted: 25 September 2022; Published: 03 October 2022

ABSTRACT

Automated Production Systems (aPS) must be more adaptable to adapt to the range of goods because discrete manufacturing is typically small batch and customised; this makes the aPS more error-prone and complex. Strategies for autonomous recovery are needed to improve system performance and decrease downtime brought on by manual intervention. Parts of the control software that treat inevitable failures planned and implemented at design-time carry out automatic recovery. Instead, reputable artificial intelligence planners should produce recovery strategies automatically to reduce engineering effort and handle unforeseen shortcomings. As a result, this study suggests breaking down the functional control software into Control Primitives, which are then used to create generated strategies. The components needed to manually implement the state machines of the various aPS operating modes are the same Control Primitives. Therefore, no further engineering work is required to prepare recoverability during the application development phase. This study presents four methods for modelling and implementing PLCexecutable Control Primitives.

INTRODUCTION

More demands are made on the efficiency, adaptability, and complexity of manufacturing processes in modern industrial production [1]. Cyber-Physical Production Systems are utilised to meet Industry 4.0 criteria and to adjust to growing production demands [2]. However, as a result of the system's increased complexity and need for flexibility, there is a greater chance of failures, and the kinds of failures are now more varied and unpredictable [3]. Extended downtime from failure recoveries needing manual intervention reduces efficiency.

As a result, it is becoming more and more crucial to automatically generate recovery techniques without human intervention.

It is suggested that artificial intelligence (AI) be used to address automated recovery issues as machine learning, deep learning, artificial intelligence (AI), and other fields are progressively applied to the manufacturing sector [4].

Automated production systems (aPS) control software is primarily created in compliance with IEC 61131-3, while IEC 61499, a more recent standard, has yet to gain any industrial traction [5]. Based on production aims and needs, current control software systems establish strict, hierarchical control architectures [6], often containing an automatic operation that may include specified recovery methods. As a result, errors must be anticipated to prevent or recover from them [7], necessitating manual invention if faults arise out of the blue. On more sophisticated contemporary production systems, this conventional method becomes unfeasible.

The control software initiates a sequence of process state transitions that enable the automatic functioning of an aPS.

From a conceptual standpoint, a failure is conceptually equal to a transition whose anticipated state of affairs differs from its actual condition. On the other hand, a recovery strategy consists of several changes from the failure state to the desired state.

AI planning can be used to automatically develop recovery strategies if all state transitions are realised by deterministic Control Primites (CP), which are defined by preconditions and postconditions (together referred to as a “contract” in the following [8]). As a result, several CPs are chosen, and the process is repeatedly brought to its desired condition through their contracts.

[1] *How to cite the article:*
Ahuja R, Planning Failure Recovery Strategies using Artificial Intelligence in Discrete Manufacturing Automation, IJRST, Jul-Sep 2022, Vol 12, Issue 3, 36-45, DOI: http://doi.org/10.37648/ijrst.v12i03.007

This automated strategy creation is complementary to a computerised operation manually programmed and created from the same pool of CPs. It is not ideal to rely solely on generated strategies since human intelligence and experience cannot optimise the process. Functional descriptions from customer needs cannot be verified as satisfied in programmed state machines.

Furthermore, it can be challenging to implement process modifications based on factors like product type or operator input because created strategies do not include control flow modifiers (IF-THEN rules). Regenerating tactics repeatedly based on external inputs would cause delays and increase the computing load substantially. On the other hand, creating partial sequences for the automated procedure could reduce some of the programming effort. Therefore, the primary contribution of this article consists of four different modelling ways to modularize the AI planning-capable discrete manufacturing control software into CPs and compose an automated operation manually programmed from the same CPs.

REQUIREMENTS

To enable the generation of recovery strategies, a model of the technical process must be constructed including the contracts of all available CPs and descriptions of the objects (technical resources, products, etc.) to which the contracts refer. Using the model, strategies in the form of CP sequences must be automatically generated based on problem descriptions, which consist of a specification of the failure (initial) state and expected (goal) states. Finally, this generated strategy needs to be executable by a Programmable Logic Controller (PLC). Details of the requirements are discussed below, and referenced in Fig.1.

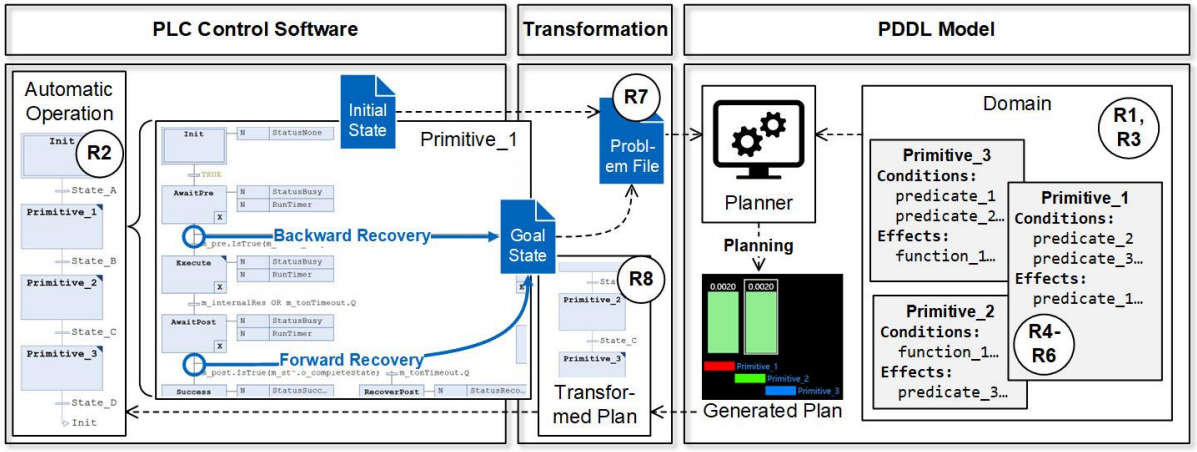

Fig. 1: Overview of the Failure Recovery Process

Since this model serves to solve problems in the AI planning domain, the model must adhere to the syntax and semantics of the related planning languages (R1). To fulfill the requirements of discrete manufacturing systems, the production process is divided into separate CPs, which are realized using technical resources. In this control model, the Control Primitives are equivalent between the PLC and planning domain (R2), and form a discrete composed structure (R3), i.e. no classification and grading of the CPs is performed. Each CP defines a contract (R4), i.e., preconditions and postconditions, or the states in between the CP transitions the technical process, which is assumed as a finite-state machine. State items (R5) include state predicates and state functions, the permutations of which can be used to determine the state of the technical process. This includes sensor data, corresponding to measurable physical properties, but also abstract relations and state information. CP contracts refer to state items. Properties of discrete products are described by characteristic variables represented as object feature items (R6), which are also considered as status items. While the above concerns the domain description, planning problems define the current state and target state of the technical process. The problem should be a classical planning problem (R7), which means the planning domain should be fully observable, deterministic, static and with single agents [10]. To minimize the recovery time, the plans should be efficient, i.e., the resulting strategy should be optimized to take the shortest possible path to the target state. The final generated strategy needs to be executable by PLCs, which means the strategies should be transformable into executable control software (R8). If the control model satisfies the above requirements, the system can use the same CPs for manually programmed normal operation as for automatically generated failure recovery.

CONCEPT

The concept proposed in this paper builds upon PDDL 2.1 due to its large feature set and robust support by established AI planners. The following first describes the incorporation of AI planning in discrete manufacturing such that existing manually programmed operating modes can be ported to the proposed modularized reconfigurable control software. Second, approaches for the definition of domain models are presented for different types of use cases.

A. Inclusion of AI Planning in Discrete Manufacturing

The problem of planning and scheduling aPS for discrete manufacturing is classified and discussed in [27]. Based on that work, this paper focuses on how to build domain models when failure recovery is considered. Thereby, the production process is divided into individual CPs corresponding to PDDL actions. The interaction of the PDDL planner with the PLC in case of a failure is shown in Fig.1. On the PLC, each CP executes a generalized state machine as shown on the left of Fig. 1, that every CP implementation inherits from a base class in the object-oriented application. Thereby, after initialization, the CP first awaits the fulfilment of its precondition, then executes its defined behavior to change the process state, and finally awaits the fulfillment of its postcondition. If precondition or postcondition are not fulfilled within a defined timeout, a problem file is generated to request a recovery. Thereby, the initial state is the current process state recorded by the machine's sensors and the PLC's state tracking. The goal state may be the failed precondition or postcondition (details in IV-A.2). If a suitable recovery strategy can be found, it is transmitted back to the PLC and executed. Otherwise, the machine starts shutting down. The CP's contract (precondition and postcondition) and timeout are abstract properties of the base class to be overwritten by the inheriting CP implementations. Similarly, the Execute method implementing the actual behavior is abstract and must be overwritten.

1) Definition of Control Primitives: The CPs to realize both the automatic operation and the recovery plans are characterized by their contracts and formalized as PDDL actions. Whereas PDDL actions describe a CP's dependence and impact on the technical process, the PLC side actually implements the described behavior. An applicable construction method is mentioned in [31]. The number of states increases exponentially as the number of state items increases. One or more CPs may execute simultaneously if their preconditions are satisfied. When the initial state, target state, and domain are given, it is possible to generate a sequence from a limited combination of CPs. Since the CP is determined only conceptually in PDDL, its actual effect must be determined using sensor data. When building the model, the sensor feedback value must be distinguished from the effect of the CP as a separate predicate, representing the desired and actual value of the effect, respectively.

2) Comparison of Automatic Operation Mode and Error

Recovery Mode: To describe the technical process, expected states, unexpected states, and unreasonable states are differentiated. During automatic operation, the state of the aPS is transposed from the initial state, via CPs, through a series of expected states until it reaches the target state for one cycle. Errors during that operation lead to an unexpected state or an unreasonable state. In an unreasonable state, products or components violate their original purpose, like a product falling off a conveyor. A failure state may also be reasonable but unexpected if, e.g., a product should have been machined but is not. The recovery process can be divided into two types, forward recovery (the system is restored to a subsequent expected state) and backward recovery (the system is restored to a previous expected state, in which it can attempt the required functions again) [12]. Thus, if partial sequences of an automatic mode shall be generated, the target state is set as the final state. The target state for failure recovery from an unreasonable state is set as the pre-error state, characterized by the precondition; for recovery from an unexpected state the target can be the preerror state and final state, as summarized in Table I.

TABLE I: Modelling of Different Scenarios

Scenarios	Initial state	Target state	Recovery type
Normal Operation	expected state	final state	-
Recovery I	unexpected state	final state/ pre-error state	forward/ backward
Recovery II	unreasonable state	pre-error state	backward

3) Definition of Object Types:

Discrete manufacturing is defined as" the production of distinct items such as automobiles, appliances or computers" [32], and is characterized by the discrete character of products, making it possible to define products as objects in planning problems. The objects need to be discrete enumerable individuals. If the described items are continuous, they need to be defined as relations in the form of equations instead of objects.

Discrete manufacturing includes objects for products, component parts, and components as parameters to distinguish different CPs and state items. On the PLC, the product is only partially observable, and the status of the product can only be confirmed or inferred from the sensor data. However, in the PDDL model, each product is an individual with a separate name, and the status of the product is clear, which creates an information gap. One possible approach to make up the information difference lies in matching the appropriate PDDL model state under normal operation scenario with the state returned by the PLC model to complement the unobserved information.

4) System Potential and Constraints:

System potential describes the capacity of technical resources to impact the technical process. However, constraints, i.e., artificially imposed restrictions on technical resources or on products, limit this potential. Constraints can be classified into strong constraints, soft constraints, and variable requirements. Strong constraints are referred to as interlocks in automation technology and are defined in (:constraints) in the PDDL 3.0 syntax. Soft constraints, i.e. constraints that will not cause the plan to fail but will increase its cost, are defined in (:preferences). Variable requirements, most of which are restricted in the problem file, are only observed in automatic production mode and can be violated during failure recovery.

B. Control Primitive Description Method

The CPs in this concept fulfill operations classified as storage, transport, production, or a combination of transport and production [27]. Among these, transport operations, which connect processing steps in series, are focused on and their diversity is investigated in detail. The machine component that undertakes a transport operation is called a carrier.

Transport modes can be classified according to the capacity of the carrier as summarized in Table II. Almost all carriers can be used for single-product transport, such as robotic arms, conveyors, etc. These carriers can be fixed-position, i.e., the deliverables can only appear in dispersed enumerable locations, such as suspension chains, or the opposite, like pipes. The mode of transport can be route-independent, i.e., ignoring the route from the origin to the destination and focusing only on reaching the target location, or conversely, the routes among the locations are fixed. However, only carriers such as belts, suspension chains, or pipes, where products can appear at multiple locations along one single transport route, are able to transport multiple products, which means they can transport products simultaneously. For example, the robotic arm transport method belongs to PM, only one or one batch of deliverables can be transported at a time, no transport routes need to be considered and the deliverables are held by the gripper (fixed carrying position).

Similarly, the unmanned transport trolley is a fixed position route mode belongs to PRM, which differs from the PM in that the transport route needs to be taken into account. Suspension chains are SPRMs because they can transport multi products simultaneously, and differ from SRMs (e.g. conveyor belts) in that the products are carried by fixed positions. How to build transport CPs for each of these transportation modes is discussed below.

Location-oriented Modelling: This modelling approach is suitable for non-simultaneous transport modes, i.e. PM or PRM, characterized by separately defined transport CPs for delivering the products to different target locations. Since each delivery is a separate CP, it is possible to set specialised contracts for each delivery. Production requirements of different products are represented universally in the domain file, rather than part of the goal in the problem file.

TABLE II: Classification of transport modes

	PM[a]	PRM[b]	SRM[c]	SPRM[d]
Simultaneous transport of multi products	No	No	Yes	Yes
Fixed transport routes	No	Yes	Yes	Yes
Fixed carrying positions	Yes	Yes	No	Yes

This modelling is compatible with the strongly constrained production requirements. For example, a robot can transport a product from A to B for processing and finally to location C, as shown in Fig. 2 The transport operation can be represented by three CPs, corresponding to three locations. The advantage of specifying the requirements in the condition of the CPs is that the generated plan is safer and more efficient, because non-conforming solutions are stopped at the time the CP is triggered, not at the time the target is checked. This means that when a planner uses forward search, risky attempts or unnecessary loops are avoided compared to the Carrier-oriented modelling discussed below (there, requirements are specified in the goal). Thus, the strategy is repeatedly modified until the goal-check passes. Consider, e.g., the predicate product is processed being added to the condition of the CPs transport to C (not to the goal), the generated plan will not be A-C-A-B-C, instead avoiding product being transported unprocessed to location C. The disadvantage is that it limits the potential and loses some flexibility of the system, because production requirements can change according to production needs, while the CPs stay unchanged.

2) Carrier-oriented Modelling: This Modelling is suitable for the PM transport mode, i.e., the carrier can move freely between the defined locations. The restrictions on transport are only relevant to the carrier and not to the location. The transport operation from the above example can be one CP move by robot, shown in Fig. 2. The advantages of this modelling are the increased flexibility of the CP definition and the simplicity of the model description. This modelling of transport CPs describes the potential of the device more closely, allowing production requirements to be described separately from the interlocks of the technical resource. The disadvantage is that, in contrast to the Location-oriented modelling, strongly restricted production requirements cannot be taken into account, because the difference between locations is singularly divided by the arrival of the carrier, while the difference arising from complex production requirements is not considered.

3) Mapping-oriented Modelling: For this modelling, suited for PRM transport modes, the location is no longer undifferentiated but formalized as abstract objects characterized by predicates. Here, the connections are either unidirectional (A connect B) or bidirectional (A before B or A after B). The map is thus constituted, the transport routes of the carriers are fixed, and the movements of the carriers are no longer arbitrary. The carrier in the previous example becomes a fixed route carrier. If there is no connection between A and C, the product cannot directly move from A to C. There are continuous transport movements in the industry, such as SRM, which cannot be represented by a discrete model. When constructing such transport movements, it is possible to discretize the continuous movements by cutting the transport route into segmented routes connecting discrete points. The continuous operation is sliced into a sequence of multiple CPs for each unit of time movement. This approach reduces the problem of failure due to floating points in continuous computation and improves the possibility of generating strategies.

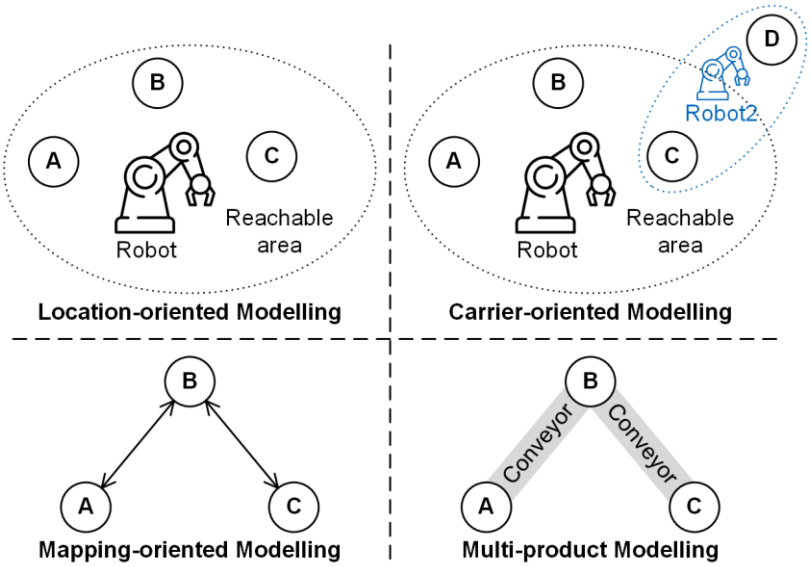

Fig. 2: Examples of four transport Control Primitives

4) Multi-product Modelling: For SRM and SPRM, the simultaneous changes of multiple positions are related rather than independent. In order to achieve this simultaneous change of product positions, the CP should not focus on the product itself, but on the product's position. The position of the product is not only defined as an attribute of the product but as an object. The position on the carrier is defined as a stand point, that is, a discrete point on which a product can stand. This belongs to the predicate of the product and can be declared in the domain file by the state item (product on stand point), initialized in the problem file and changed by manipulating CPs. The difference between SRM and SPRM is that the standing points in SRM are no longer discrete points, but a continuous area. In order to simplify the model, discrete stand points in the continuous areas could be defined, and no products can exist in the area outside the stand points.

To summarize, the location-oriented modelling distinguishes each location in detail, and although the generated strategy is more efficient, it limits the capability and the flexibility of the system. The carrier-oriented modelling blurs the distinction between locations and differentiates them only by whether the carrier is reachable, which is closer to the device's capabilities and simplifies the code. The mapping-oriented modelling defines the route separately to accommodate PRM requirements on a carrier-oriented basis.

The multi-product transport modelling, on the other hand, fulfills the requirements for simultaneous transport of both SRM and SPRM.

EVALUATION

As an application example to evaluate the previously proposed control model, the bench-scale manufacturing system xPPU2 is used. The following subsection provides an introduction of the xPPU and the investigated application. In section V-B, experiments are carried out for three processes based on section IV-A.2. The generated strategies are transformed into IEC 61131-3 code and executed on the xPPU for verification.

A. Application Example xPPU

Multiple xPPU usage scenarios and the operational components involved are detailed in a technical report [9]. The xPPU, in the scenario investigated here, consists of five components, shown in Fig. 3. A stack is used to store the workpieces. At the bottom of the stack, there is a cylinder that pushes out one workpiece at a time. The crane can rotate in both directions and transport one workpieces by using the valve-controlled gripper, one piece at a time. A Large Sorting Conveyor (LSC) is connected to three ramps, one at the end of the conveyor and two on the side. The stamp is used to simulate a machining step of the workpiece. The PickAlfa Conveyor (PAC) transports workpieces that leave the LSC via a separator. Other components of the demonstrator are not in use. Three types of workpieces are considered as products: white plastic, black plastic, and metal. One single workpiece is pushed out by each stack actuation to be picked up by the crane. Black workpieces are transported directly to the LSC, the others are transported to the stamp. When the stamp is finished, the crane can then pick the workpiece up and turn to the LSC. The LSC transports the workpieces to them assigned ramps. If a ramp is full, the remaining workpieces are transported to the discard ramp.

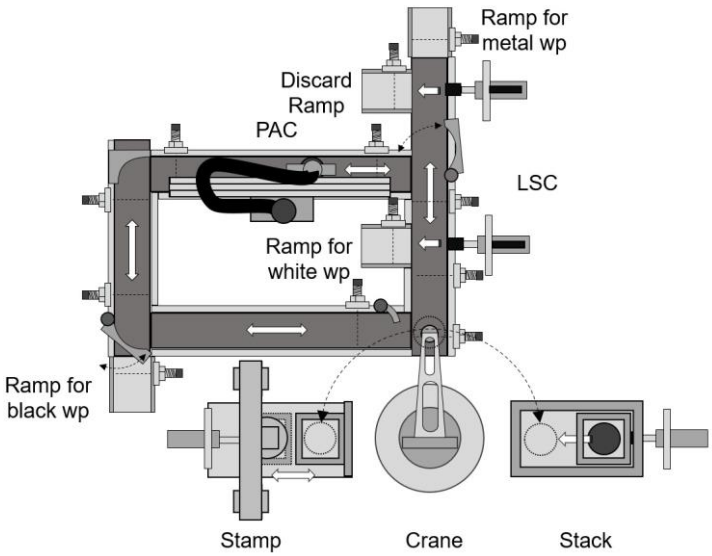

Fig. 3: A schematic view of the xPPU

B. Experiments Performed with Discrete Control Models

Based on the PDDL requirements deriving and their fulfilment by different planners, Optic3 is chosen. Planners that do not support temporal calculation were excluded as time requirements are essential for efficient manufacturing. Among planners which plan temporal domains the Partial-Order Planning Forward planners (POPF) belong to the most popular planners. Among POPF planners the Optic planner supports most of the required PDDL requirements. All performances mentioned in the remainder have been collected on a consumer-grade laptop computer, equipped with a 4-core, hyper threaded 1.8 GHz CPU and 16 GB RAM. The generated strategies of below use cases4 are visualized using the PDDL extension of VS Code developed by Jan Dolejsi5.

CONCLUSION

This paper presents a modelling and implementation approach for aPS to use AI planning for the automatic generation of failure recovery strategies in discrete manufacturing. The model is applicable for solving failure recovery problems, but currently only considers discrete manufacturing with a focus on transport operations. It is unlikely to be applicable to any non-discrete processes.

REFERENCES

[1] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, “How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective,” JICE, vol. 8, no. 1, pp. 37–44, 2014.

[2] J. Yang, X. Wang, and Y. Zhao, “Parallel manufacturing for industrial metaverses: A new paradigm in smart manufacturing,” IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 12, pp. 2063–2070, 2022.

[3] C. Baydar and K. Saitou, “Off-line error prediction, diagnosis and recovery using virtual assembly systems,” Journal of Intelligent Manufacturing, vol. 15, no. 5, pp. 679–692, 2004.

[4] M. Ghahramani, Y. Qiao, M. C. Zhou, A. O’Hagan, and J. Sweeney, “Ai-based modeling and data-driven evaluation for smart manufacturing processes,” IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 4, pp. 1026–1037, 2020.

[5] V. Vyatkin, “Software engineering in industrial automation: State-of the- art review,” IEEE Transactions on Industrial Informatics, vol. 9, no. 3, pp. 1234–1249, 2013.

[6] P. Leit˜ao, “Agent-based distributed manufacturing control: A state of- the-art survey,” Engineering applications of artificial intelligence, vol. 22, no. 7, pp. 979–991, 2009.

[7] M. Zhou and F. DiCesare, “Adaptive design of petri net controllers for error recovery in automated manufacturing systems,” IEEE Trans. Syst., Man, Cybern., vol. 19, no. 5, pp. 963–973, 1989.

[8] B. Meyer, “Applying ’design by contract’,” Computer, vol. 25, no. 10, pp. 40–51, 1992.

[9] B. Vogel-Heuser, S. Bougouffa, and M. Sollfrank, Researching Evolution in Industrial Plant Automation: Scenarios and Documentation of the extended Pick and Place Unit. Institute of Automation and Information Systems, TU Munich, 2018.

[10] S. J. Russell, Artificial intelligence a modern approach. Pearson Education, Inc., 2010.

[11] E.-M. Neumann, B. Vogel-Heuser, and J. Fischer, “Challenges for motion systems in automated production systems – an industrial field study,” in IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society, 2022, pp. 1–6.

[12] “Ieee standard glossary of software engineering terminology,” IEEE Std 610.12-1990, pp. 1–84, 1990.

[13] A. White, A. Karimoddini, and M. Karimadini, “Resilient fault diagnosis under imperfect observations - a need for industry 4.0 era,” IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 5, pp. 1279–1288, 2020.

[14] M. Ramdani, L. Kahloul, M. Khalgui, Z. Li, and M. Zhou, “Rctl: New temporal logic for improved formal verification of reconfigurable discrete-event systems,” IEEE Transactions on Automation Science and Engineering, vol. 18, no. 3, pp. 1392–1405, 2021.

[15] P. Bareiß, D. Sch¨utz, R. Priego, M. Marcos, and B. Vogel-Heuser, “A model-based failure recovery approach for automated production systems combining sysml and industrial standards,” in IEEE ETFA, 2016, pp. 1–7.

[16] D. Sch¨utz, M. Schraufstetter, J. Folmer, B. Vogel-Heuser, T. Gmeiner, and K. Shea, "Highly reconfigurable production systems controlled by real-time agents," in IEEE ETFA, 2011, pp. 1–8.

[17] R. Priego, D. Sch¨utz, B. Vogel-Heuser, and M. Marcos, "Reconfiguration architecture for updates of automation systems during operation," in IEEE ETFA, 2015, pp. 1–8.

[18] C. Legat, D. Sch¨utz, and B. Vogel-Heuser, "Automatic generation of field control strategies for supporting (re-) engineering of manufacturing systems," Journal of Intelligent Manufacturing, vol. 25, no. 5, pp. 1101–1111, 2014.

[19] S.-O. Bezrucav, G. Canal, A. Coles, M. Cashmore, and B. Corves, "Towards automatic state recovery for replanning," in ICAPS Workshop on Integrated Planning, Acting, and Execution, 2022.

[20] M. Ghallab, D. Nau, and P. Traverso, Automated Planning: theory and practice. Elsevier, 2004.

[21] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," Artificial intelligence, vol. 2, no. 3-4, pp. 189–208, 1971.

[22] E. P. Pednault, "Adl: Exploring the middle ground between," in Proceedings of the first international conference on Principles of knowledge representation and reasoning. Morgan Kaufmann Pub, 1989, p. 324.

[23] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson, et al., "Pddl— the planning domain definition language," Technical Report, Tech. Rep., 1998.

[24] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," Journal of artificial intelligence research, vol. 20, pp. 61–124, 2003.

[25] S. Edelkamp and J. Hoffmann, "Pddl2. 2: The language for the classical part of the 4th international planning competition," Technical Report 195, University of Freiburg, Tech. Rep., 2004.

[26] A. Gerevini and D. Long, "Plan constraints and preferences in pddl3: The language of the fifth international planning competition," Tech. Rep., 2005.

[27] A. Rogalla, A. Fay, and O. Niggemann, "Improved domain modelling for realistic automated planning and scheduling in discrete manufacturing," in IEEE ETFA, vol. 1, 2018, pp. 464–471.

[28] J. Huckaby, S. Vassos, and H. I. Christensen, "Planning with a task modeling framework in manufacturing robotics," in IEEE/RSJ Conference on Intelligent Robots and Systems, 2013, pp. 5787–5794.

[29] B. Wally, J. Vysko˘cil, P. Nov´ak, C. Huemer, R. ˇ Sindel´a˘r, P. Kadera, A. Mazak-Huemer, and M. Wimmer, "Leveraging iterative plan refinement for reactive smart manufacturing systems," IEEE T-ASE, vol. 18, no. 1, pp. 230–243, 2021.

[30] Plattform Industrie 4.0, "Information model for capabilities, skills & services," Berlin, 2022. [Online]. Available: https://www.plattformi40. de/IP/Redaktion/DE/Downloads/Publikation/CapabilitiesSkills Services.pdf

[31] B. Wally, J. Vysko˘cil, P. Nov´ak, C. Huemer, R. ˇ Sindelar, P. Kadera, A. Mazak, and M. Wimmer, "Flexible production systems: Automated generation of operations plans based on isa-95 and pddl," IEEE RA-L, vol. 4, no. 4, pp. 4062–4069, 2019.

[32] J. F. Cox and J. H. Blackstone, APICS dictionary. APICS Educational Society for Resource Management, 2002.

[33] V. Mokhtari, A. Sathya, N. Tsiogkas, and W. Decr´e, "Safe-Planner: A single-outcome replanner for computing strong cyclic policies in fully observable non-deterministic domains," in Proceedings of the 20th International Conference on Advanced Robotics (ICAR). IEEE, 2021, pp. 974–981.