Services in Smart Manufacturing: Comparing Automated Reasoning Techniques for Composition and Orchestration

Flavia Monti*, Luciana Silo*, Francesco Leotta, and Massimo Mecella

Sapienza Università di Roma, Italy
{monti, silo, leotta, mecella}@diag.uniroma1.it

Abstract. In recent years, there has been an increase interest in using intelligent methods to control manufacturing processes. Tens of resources to be modeled and offered as services through Industrial APIs, may be used in these processes and orchestrated throughout the various supply chain companies. The orchestration must be flexible and adaptable to disruption since the status of the various services/resources changes over time in terms of their cost, quality, and likelihood of failure. Due to the large amount of services involved and the complexity of their behaviors, manually making judgments quickly becomes impractical, necessitating the use of automated solutions to resolve the issue. By relying on the resources information provided by proper Industrial APIs, we can make current supply chains flexible and robust. In this work, we investigate the potential and limitations of automated reasoning techniques to enable adaptivity and resilience in smart manufacturing.

Keywords: Industrial APIs · Smart manufacturing · Automated reasoning.

1 Introduction

The concept of *smart manufacturing*, commonly also mentioned as *Industry 4.0*, embodies a vision of industrial processes where computing devices are integrated in most of the manufacturing steps. In particular, industrial processes are supposed to be fully (or mostly) automated, adaptive to changes, flexible, evolvable, resilient to errors and attentive to the more knowledgeable operators' skills and needs.

Nevertheless, processes in current manufacturing landscape, must not be considered isolated. Instead, they involve several companies along intricate supply chains networks [3]. Such players co-operate together to accomplish various production goals. They consist of loosely coupled, autonomous entities with equal rights, and their organizational structure is dynamically adapted in accordance

* Authors contributed equally.

with the tasks to be carried out [33]. Supply Chain Design (SCD) [22] is typically a difficult job with numerous competing objectives. Facility location planning, allocation of customers to distribution centers or factories and suppliers selection are some of them. In addition, recovery strategies are a fundamental tactic for dealing with disruption events caused, for instance, by broken machines or environmental perturbation. The literature contains a variety of heading techniques [31], including relational strategies such as supply chain collaboration, communication and information exchange [6]. In this sense, it is essential for alliances to contribute to quickly recover from disruption and to collaboratively plan with other supply chain partners. Furthermore, a flexible supply chain network structure is suitable for developing effective disruption risk recovery strategies [12].

In general, a common goal in this context is the development of the *triple-A* supply chain, which consists of the simultaneous implementation of *agility* – responding to short-term changes in demand or supply quickly, *adaptability* – adjusting supply chain design to accommodate market changes, and *alignment*, establishing incentives for supply chain partners to improve performance of the entire chain [25]. We enrich such a notion by including *resilience*, as the ability to react to disruptions along the chain. It is clear that the ability of a system to adapt to certain conditions, e.g., a rescheduling of the production process, and the capacity to continue the work despite disruptions, such as the breakdown of a machine, are two crucial objectives in a smart manufacturing environment. And besides, these are particularly challenging due to the dynamism and uncertainty of manufacturing processes. As an example, machines are subject to wear and can show unpredictable behaviors, so they may often not perform their job properly.

The overall amount of manufacturing resources in the supply chain is substantial. Also, they belong to several different categories including software systems, machines, robots, and human workers. Each resource offers a specific collection of capabilities and has unique qualities, e.g., speed, costs, and probability of break. Noteworthy, the very same functionality can be offered by different resources, optionally from different categories (e.g., painting a part can be done either by a machine or by a human), and the execution of a multi-party process requires an accurate selection of resources in order to be completed in the most convenient way. Such a selection though, cannot be considered permanent as characteristics of resources change over time as well as needs and conflicting performance measures. Additionally, non-trivial constraints between resources may exist, making the overall task of choosing actions and resources difficult to be performed manually. In this regard, the employment of Artificial Intelligence (AI) techniques can simplify the task. In particular, specific automated reasoning techniques though have their own expressiveness that, in turn, influences the computational costs.

In this paper, we explore how automated reasoning techniques, which are a specific type of AI, can be used to enable adaptivity and resilience to multi-party processes in smart manufacturing. In fact, it is claimed that the use of specialized supporting technologies and techniques enable the advent of AI-augmented

Business Process Management Systems (ABPMSs) [13], an emerging class of process-aware information systems empowered by trustworthy AI technology which enhance the execution of business processes with the aim of making them more adaptable, proactive, explainable, and context-sensitive. To this aim, as proposed in [5], we model the manufacturing resources as components of a Service Oriented Architecture (SOA). Each manufacturing resource involved in the supply chain is a service accessible through *Industrial Application Programming Interfaces (APIs)*. Industrial APIs provide many features like accessing the selected services, enabling quick integration, monitoring the behavior and status information, and invoking commands. In particular, with respect to the status, an Industrial APIs allows to access peculiar information about the Remaining Useful Life (RUL) of a resource, the cost, and the probability of failure, which evolves over time. We embed such an approach in a framework able to support adaptivity and propose different AI methods to assist it.

In order to show the suitability of the different approaches, we apply them to the tricky case of integrated circuits (chip) manufacturing, analyzing the efficiency, adaptivity, and limitations of the different approaches. Although semiconductor design activities are concentrated in specific regions of the USA, as well as in Europe and Japan, semiconductor manufacturing is more widely dispersed. The industries that provide manufacturing inputs and purchase finished semiconductor products are often dominated by large, multinational organizations [26]. In addition, as witnessed by the recent evolution of international political affairs, this production is strongly influenced by relationships among countries, which may produce unpredictable effects on the supply chain.

The rest of the paper is structured as follows: Section 2 presents a framework enabling adaptivity in smart manufacturing, while Section 3 presents the motivating case study we analyze. Section 4 presents the various approaches we investigate to compute a *resilient* and *adaptive* plan for industrial production. Section 5 compares and discusses the presented approaches. Finally, relevant literature and concluding remarks are presented in Section 6 and Section 7, respectively.

2 A framework Supporting Adaptivity

We propose a service-based framework enabling adaptivity in smart manufacturing (see Figure 1). We identify three main components, i.e., Industrial APIs, Enactor and Controller, each characterized by fundamental roles.

On the one hand, we enable interoperability between the manufacturing resources by modeling each of them as a service. The term resource encompasses here a wide range of actors including machines, humans, companies and provided services. Thus, we create a service-based supply chain consisting of a composition of services. Such services are realized as *Industrial APIs*. We consider these as APIs provided by the resources and employing specific industrial protocols (e.g., MQTT, OPC-UA) rather than classical ones (e.g., REST). The Industrial APIs are used to represent the physical actors and perfectly describe their func-

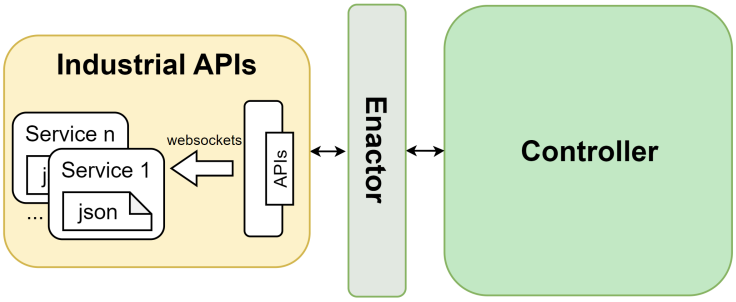

Fig. 1. A service-based adaptive framework

tionalities (or tasks). They allow to monitor the behavior and status information and allow to invoke commands. The core component of the Industrial APIs is a server that allows the management of all the services involved in the process. The server is in turn composed of a WebSocket server and an HTTP server. Particularly, it connects to the services via WebSockets in separate communication channels, and exposes a set of APIs to manage external HTTP requests. The defined APIs allow retrieving both the specification and the current state of the services and request the execution of a task to be performed by a service. Each manufacturing resource is described as a JSON file which is used by the server to "build" the service. The JSON file contains specific elements: *(i)* an *id* to specify the identifier of the service, e.g., name of the resource, *(ii)* some *attributes* that contain the static characteristics of the service, e.g., actions and costs, and *(iii)* some *features* that contains the dynamic characteristics of the service, e.g., status, breaking and quality condition. The way attributes and features are represented varies depending on the scenario in which the Industrial APIs are used.

On the other hand, the *enactor* and *controller* components are responsible both to manage and enable adaptivity in smart manufacturing. The enactor acts as a middleware by interfacing with the Industrial APIs in order to check whether the available services have changed (for instance, because of the wearing out during the execution, some services may become unavailable). The controller, which we modeled as a black box, represents the implementation of the adaptive techniques. It takes as input the specification of the involved resources (in our case a set of services) and the final target (in our case a manufacturing goal) and provides as output an adaptive orchestration[1]. Such an orchestration contains the specification of the identified services required to reach the final goal. Following the output of the controller, the enactor calls (HTTP requests through the Industrial APIs) the designed services.

By taking into account the various possibilities for the inputs and outputs of the controller, a conceptual classification of the potential strategies produced as

[1] In a SOA, and in this paper specifically, orchestration and process can be considered as equivalent.

output by the controller can be specified. We can distinguish among the deterministic and probabilistic behaviors of manufacturing resources and the complete and under specification of the manufacturing goals. Three distinct categories of strategies are described in the paragraphs that follow.

Instance repair. The process behind the supply chain is well-defined. If an unexpected exception happens (e.g., a machine breaks), automated reasoning is employed in order to take the states of resources back to the expected ones. In this case, adaptivity is applied locally, but the overall forthcoming orchestration (i.e., the remaining part after the exception) remains unchanged.

Instance planning. Every time that a new process instance is needed, automated reasoning is applied taking as input the most recent information about resources and producing as output an entire orchestration specification. If, at a certain point of the execution, something (e.g., a broken resource) prevents the plan to be completed, automated reasoning is applied again.

Policy-based. Automated reasoning is employed to obtain a policy, i.e., a function that for each state proposes the next action. Differently from the *instance planning* case, here, if something unexpected happens, there is no need to reapply planning, as all the possibilities have been already computed.

The intuitions of three methods implementing *(i)* an instance planning approach based on deterministic services and loosely specified target, *(ii)* a policy-based approach with stochastic behaviors and fully specified target, and *(iii)* a policy-based approach with stochastic behaviors and a loosely specified target are described in Section 4. We are not considering any instance repair approach as we focused on the data perspective more than the control flow perspective, which is fundamental as full adaptivity requires the process structure to be very flexible. An example of a technique of this type, even if not applied in a smart manufacturing scenario but in a ubiquitous computing one, has been proposed and investigated in [30].

3 Motivating Case Study

In this section, we present the supply chain case study, i.e., *ChipChain*. It represents the chip supply chain production which involves several actors associated with different operations[2]. The main goal is the production of chips and we distinguish two main phases, the raw materials and design assortment phase, and the manufacturing process phase. The manufacturing operations involved in the *chip* supply chain are outlined below:

- *Raw materials and design assortment*: consists of the collection of the chip design (e.g., CAD model) and the essential raw materials, i.e., (silicon) wafer, silicon, boron, phosphor, aluminum, resistance, plastic, copper frame, and chemicals. The design and the raw materials constitute the objects involved in the *manufacturing process*.

[2] Cf. https://www.screen.co.jp/spe/en/process and https://www.asml.com/en/news/stories/2021/semiconductor-manufacturing-process-steps.

– *Manufacturing process*: upon the assortment phase is completed, the manufacturing process begins. It represents the effective set of operations for the manufacturing of chips. Such operations include: *(i)* cleaning of the silicon wafers; *(ii)* deposition of thin film of conducting and isolating materials, e.g., silicon and aluminum; *(iii)* coating of the wafer surface with resistance; *(iv)* exposing of the wafer with ultraviolet radiation; *(v)* "development" and *(vi)* etching of the wafer; *(vii)* implantation of phosphor, boron ions or silicon; *(viii)* creation of micro transistors through a heat processing; *(ix)* stripping off of resistance (using chemicals); *(x)* separation of the wafer into individual chips connected to a copper frame; *(xi)* testing phase of the chips; and *(xii)* enclosing in plastic by leaving only the contact pins on the outside.

We based the manufacturing process in the United States (US), which is home to the vast majority of the top semiconductor suppliers in the world[3]. Additionally, we identified the list of states involved in the assortment phase. In particular, we identified the countries that produced the raw materials and considered them as the organizations part of the supply chain. In addition, we determined the costs of carrying out each operation. The distance between the US and the identified states is used to calculate them (if the object is made in the US, the associated operation cost is unitary). The *manufacturing process* is a different matter. Indeed, the manufacturing actors (machines and operators) are located in a unique factory (in the US) and the cost of the operations is set to 1. However, we take into account the possibility of more than one of the same actor performing the same action, in which case their costs are increased by a factor greater than 1.

4 Adaptive Supply Chain Approaches

In this section, we explain in detail how we implement the approaches used for finding a manufacturing production plan that ensures adaptivity in the *ChipChain* case study.

4.1 Instance Planning

Generally, by leveraging automated planning techniques, it is possible to automatically orchestrate the supply chain in order to fulfill specific manufacturing goals while respecting expected Key Performance Indicators (KPIs) [5]. Automated planning and scheduling (AI planning) has already proven its potential and could have a huge impact on industrial manufacturing too. It concerns the automated synthesis of autonomous behaviors (i.e., plans) from a model that describes the behavior of the environment in a mathematical and compact form.

Classical planning is one of the most basic models in planning and is concerned with the selection of actions for achieving goals when the initial situation

[3] Cf. https://macropolo.org/digital-projects/supply-chain/ai-chips/ai-chips-supply-chain-mapping.

is fully known and actions have deterministic effects [16]. A *classical planning problem* [18] $P = (A, I, \gamma, O)$ consists of a set of state variables A, a description (i.e., a valuation over A) of the initial state of the system I, a goal γ represented as a formula over A, and a list of operations O over A that can lead to state transition. State variables A and actions O constitute the *planning domain* P_D of the planning problem. The solution for a classical planning problem is a sequence of operators (a plan) whose execution transforms the initial state I into a state satisfying the goal G. The computation of the solution may not be an easy task, however, over the years, many algorithms and heuristics have been proposed and embedded into planning systems (known as *planners*) to perform automated planning in an efficient way.

We propose a resilient and scalable approach based on classical planning for the agile orchestration of services aiming at achieving a production goal and adapting to failures by using the framework described in Section 2.

The manufacturing resources, represented as services with the Industrial APIs, contain the list of actions runnable by the specific actor modeled in a PDDL-like fashion (PDDL defined in the following). The JSON file describing the services is structured as follows: the *attributes* value is actions-based and each action contains the *parameters*, the *requirements* and the *effects*, both positive and negative, necessary upon the execution of such an action finally, the *features* value contains the characteristics of the actors and their current status.

The *controller* component comprises two sub-modules, i.e., the translator and the planner. The core module of the controller is the *translator*. It transforms the service descriptions (acquired through HTTP requests) and a description of the environmental context (i.e., the production goal and environmental context description) into a planning problem specified in PDDL 2.2. (Planning Domain Definition Language). PDDL [15] is a standard language to describe planning domains and problems. There exist different versions of PDDL which provide an increasing expressive power. The model of the planning problem is separated into two main files: the *domain* and the *problem* descriptions. Such a division allows for an intuitive separation of elements present in every specific problem of the given domain (i.e., types, predicates, functions and actions), and elements that determine the specific problem (i.e., available objects, initial state and goal). The *planner* module represents a planning system able to generate a plan given the domain and problem files. The current prototype of such an approach is based on the FastDownward planner, which supports PDDL 2.2 [20]. The output of the planner, i.e., a plan, contains the list of actions (or operations) needed to reach the final goal by minimizing the total cost.

The *enactor* takes each action contained in the plan file produced by the planner and makes HTTP requests to relative Industrial APIs to perform such actions. Prior to the execution of an action, the current status (a *feature*) of the service is checked. On the one hand, if the status is `available`, the enactor dispatches the action request. On the other hand, if the status is `broken` (meaning that the service is unavailable), the enactor requests the controller the

re-calculation of the plan. Particularly, the new plan is generated starting from the current environmental state where one or more services result broken.

4.2 Stochastic Policy

Typically, actors in industrial manufacturing can be thought of as stochastic players. Indeed, their behaviors can be expressed as dependent on a probability that indicates the possibility to act either as functioning or broken actors. In this sense, the manufacturing process becomes a probabilistic planning problem. Adaptivity in such a scenario can be achieved by employing Markov Decision Processes (MDPs) to compose the services representing the manufacturing actors. Particularly, MDPs are able to take into account both the probability and cost of breaking upon the execution of an action enabling a full insight of the entire supply chain.

A MDP [32] is a discrete-time stochastic control process containing *(i)* a set Σ of states which represent the status of the service, *(ii)* a set A of actions i.e., the set of tasks that the service can perform, *(iii)* a transition function P that returns for every state s and action a a distribution over the next state i.e., the probability of the service to end in a certain status performing a certain task, *(iv)* a reward function R that specifies the reward when transitioning from state s to state s' by executing action a, and *(v)* a discount factor $\lambda \in (0, 1)$ which determines how important future rewards are to the current state. If $\lambda = 0$, the service is "myopic" in being concerned only with maximizing immediate rewards. As λ approaches 1, the method becomes more "farsighted", more strongly considering future rewards.

By relying on the framework described in Section 2, we propose a stochastic policy approach that employs MDPs to orchestrate services in order to produce an adaptive process. A formal description of such an approach is presented in [10].

We define the actors as *stochastic services* modeled as MDPs and maintained by the Industrial APIs. The JSON file describing the characteristic of the actors contains the set of transitions an actor is able to perform and the information relative to the initial, final, and current state. Each actor may represent a *complex breakable service* that includes the set of states (i.e., Ready, Configuration, Executing, Broken, Repairing) and actions, or may represent a *generic breakable service* that includes a subset of states (i.e., Available, Done, Broken) and a subset of actions. Moreover, each actor may represent also *human workers* that can perform the same action of a machine and are preferred when a machine becomes broken, or a *warehouse*. Such an approach allows the flexibility required to model a manufacturing actor operating in its environment. As an example, specific states can be defined to model unavailability conditions (e.g., a broken machine) and the probability of ending in such states. In addition, rewards can be used to model the costs of performing an operation. Different actors can offer the same operation and, as a consequence, an actor chosen for a specific process instance could be discarded for the later instance. The characteristics of the stochastic services is combined into a community of stochastic services, i.e., a *stochastic system service*. Intuitively, the

stochastic system service status includes the current status of all the composing services, and a specific action performed on the system service changes only one component of the current state, corresponding to the service selected to execute that action.

Among others, we propose a model to define the manufacturing target by using the concept of *target service*. Such a concept is used to denote a composite service obtained by composing the functions of the stochastic services. Noticeably, the target service itself, as the stochastic services, is a particular case of MDP. However, different from the stochastic services, in the vast majority of cases, the target service is deterministic.

The solution technique of the proposed approach is based on finding an optimal policy for the *composition MDP*. Given the specification of the stochastic system service and the target service, we compute the composition MDP which contains: the cartesian product between all the states of the target service and the stochastic system service, all the actions of the services, the probability of ending in a certain system state performing a system action, and the reward function that is the reward observed from doing a system action summed to the reward coming from the target. In practice, according to a specific target (manufacturing goal), the composition MDP computes all the possible executions of the manufacturing process, i.e., by combining together the specifications of all the actors (stochastic services) and the goal it identifies all the possible status of the actors at any step. The optimal policy of the composition MDP is computed through *policy iteration* and/or *value iteration* [34].

By leveraging on the optimal policy, the *enactor* dispatches the HTTP requests to the chosen services (Industrial APIs) according to the solution. Notice that the policy assigns the services to each action taking into account both probabilities and costs. It is not straightforward indeed to determine a-priori which service a certain action must be assigned to. Additionally, before dispatching the request, it checks whether the current status and the transition functions have changed (for instance because of the wearing out during the execution). We distinguish two different adaptive scenarios. On the one hand, when only the status of an actor changes, the enactor is able to choose the next action to be performed by checking the result of the optimal policy from the new state formed. On the other hand, when both the status and the transition function of an actor change, the enactor requests the re-computation of the optimal policy from an up-to-date composition MDP which includes the latest condition of the service.

4.3 Stochastic Constraint-based Policy

It is quite common that the manufacturing process is represented using a structured process formalism, such as BPMN or Petri Nets [14]. We are doing more than that, we employ the flexible formalism named DECLARE, directly based on LTL_f, to define the manufacturing process. This permits to model the process as a set of logical conditions, so as to more easily specify those processes in which

human experience plays a key role or in which the rules of precedence between operations cannot simply be modeled as a sequence.

The proposed technique is an extension of the previous approach. Note that the definitions of both stochastic services and stochastic system service remain the same. By contrast, the target specification of the manufacturing process is represented as an LTL$_f$ formula φ [11] derived from the DECLARE formalization.

The collection of services representing the actors can perform actions in $\mathcal{P}$ and, moreover, to make our model richer we allow services to execute a broader set of actions. In addition, we put each LTL$_f$ formula in conjunction in order to compute the equivalent deterministic finite automaton (DFA) (made by `Lydia` tool [9]), i.e., *target* DFA.

Given both the stochastic system service and the target DFA we compute the *composition MDP* that in this case contains: the cartesian product between all the states of the target DFA and the stochastic system service, the product between the DFA action and the service that performs the action, the probability of ending in a certain system state performing a system action, and the reward function formed by the reward observed from doing a system action. In practice, according to a specific target (manufacturing goal), the composition MDP computes all the possible executions of the manufacturing process, i.e., by combining together the specifications of all the actors (stochastic services) and the goal (LTL$_f$ formula), it identifies all the possible status of the actors at any step.

We compute the optimal policy of the composition MDP, as in the previous case through *policy iteration* and/or *value iteration* [34]. Such a policy contains the specification of the optimal actions (and related services) to execute from each possible state in order to reach the final goal. The *enactor* acts as a middleware that interfaces with the Industrial APIs in order to check whether the current status and the transition functions have changed (for instance because of the wearing out during the execution). As in the previous approach, we can distinguish two different resilience scenarios. On the one hand, when only the status of an actor changes, the *controller* is able to choose the next action to be performed by checking the result of the optimal policy from the new state formed. On the other hand, when both the status and the transition function of an actor change, the *controller* re-computes the optimal policy from an up-to-date composition MDP which includes the latest condition of the service. Through the Industrial APIs, the *enactor* calls the services identified in the optimal policy computed by the *controller*.

5 Discussing the Approaches

We conduct preliminary tests by applying the proposed approaches to the *ChipChain* case study[4].

[4] Source codes of the implemented approaches, and tests, are available for repeatability the instance planning approach at https://tinyurl.com/instanceplanning and the

We divided the case study into two phases, i.e., the collection of the necessary raw materials and chip design and the effective production process. Additionally, we run several experiments by increasing the number of service copies available for the fulfillment of a specific task involved in the process and we measure the execution time and the memory usage. In the following we highlight some important aspects we identify in the results.

Performance values vary greatly depending on the approach used to find the adaptive process. In the planning approach, increasing the number of services, time and memory consumption does not change significantly. This is possible because planning solvers employ well-known heuristics able to derive, in an efficient way, a solution. However, this approach does not consider the stochasticity typical of an industrial context. In particular, does not take into account for example the probability for a certain machine to end in a failure situation, represented instead in the stochastic approaches.

On the other hand, the values of the stochastic policy approaches increase exponentially, as the number of services increases. This happens because of the definition of the composition MDP. It consists of a cartesian product operation that takes into account both the target and system service sets. Although the target is static and well-defined, the system service may increase with the added services resulting in an increment of memory consumption and execution time. Additionally, the stochastic constraint-based policy approach is more time and memory consuming with respect to the stochastic policy approach. This notable difference derives from the target service concept where we define a set of logical constraints between actions. Also, we consider the "auxiliary actions", i.e., actions that do not concern the process being realized but are needed by the services to get ready for the execution of main actions.

Even though planning is often a more effective strategy, it may not be the only factor to consider when it comes to production. Depending on the situation, it is necessary to take into account not just how quickly calculations are performed but also how the system is modeled and responds to unknown events. As they reflect the stochasticity behaviors in the manufacturing domains, both the stochastic techniques, which end up being much slower than planning, offer more expressive power.

The proposed approaches are based on the definition of constraints regarding the process execution. Depending on the used approach, such constraints are modeled in a different way. The approach based on planning requires not to be generic in the specification of the actions which are essential in the computation of the plan (given a goal). On the other hand, it permits to model the involved objects by monitoring the production progress.

The approach based on stochastic policy requires a full definition of the manufacturing process (defined as an automaton). This is different in its extension which employs LTL_f. Here the process is defined by specifying constraints between the tasks. In the modeling, we do not include relationship effects possibly

two stochastic policy approaches at https://tinyurl.com/stochasticpolicy. The source code of the Industrial APIs layer is available at https://tinyurl.com/IndustrialAPIs.

existing between the services (e.g., if the service A is used, then service B cannot be used). In the case of planning, this type of constraint could be modeled with the PDDL conditional effects which, however, have consequences in the computation costs. It is not easy to model this behavior in the stochastic policy approaches because the target service only knows manufacturing tasks and has no idea of the services employed. A possibility could be to introduce related constraints when the composition MDP is computed. Anyway, in this paper, we model such constraints by increasing the services costs of affected services.

Moreover, in this paper we focus on a case study focusing on a batch production, i.e., production of a batch of one product [19]. In this sense, we study the adaptivity by taking into consideration the fact that a specific task of the supply chain production is executed on a batch, thus if a decision is taken at the beginning of a task, it is maintained until the end of it. Such an approach influences adaptivity by discarding the possibility of adapting the production inside a specific batch and considering only the adaptivity at the end of a task.

6 Related Works

In this paper, we focus on approaches leveraging automated reasoning techniques to enable adaptation in smart manufacturing, specifically in the supply chain context. We can contextualize our work in the broader research area that applies automated planning techniques. We refer to automated planning as the application of AI technologies to the problem of generating a correct and efficient sequence of actions [28]. Furthermore, we can distinguish between classical planning, which deals with deterministic scenarios, and so-called decision-theoretic planning [4], which deals with stochastic behavior.

Examples of classical planning in smart manufacturing are provided in several works. The authors in [29] employ automated planning in order to cope with exceptional and unanticipated events. In particular, planning is employed to fix a process instance, restoring the conditions to continue with the standard, manually defined process. In [35], the authors show how to plan the assembly of small trucks from available components and how to assign specific production operations to available production resources. In [27], the authors develop an evaluation with a physical smart factory that resolves detected exceptional situations and continues process execution. However, all the solutions based on classical planning do not consider a crucial aspect of manufacturing production, i.e., the uncertainty typical of the entire production process and of the manufacturing actors.

The application of decision-theoretic planning approaches might be a solution. An example is [8], where the authors define a set of degrading planning domains. The planner tries to find a solution in the most restrictive, optimal domain. If during the execution, assumptions of a plan are not verified, due for example to failures, more and more sub-optimal domains are employed. The approach focuses on the entire process and the non-determinism of manufacturing

actors is modeled. Furthermore, MDPs are a widely used model for decision-making problems.

Nevertheless, the literature presents limited research on the application of MDPs in the manufacturing domain. Authors in [21] propose a self-adaptive Automated Guided Vehicles (AGVs) control model, depicted as an MDP, that enables AGVs to avoid collisions efficiently, safely, and economically. The work [7] presents a hierarchical MDP approach for adaptive multi-scale prognostics and health management for smart manufacturing systems. The goal is to create a policy for making sequential decisions that will maximize the expected gain under the set of constraints. Authors in [2] use an MDP for finding an optimal cost-effective maintenance decision based on the condition revealed at the time of inspection on a single diesel engine. In these cases, the use of MDPs fits very well in the manufacturing context and in particular non-deterministic domains, because it always allows making the best choice.

Finally, in this paper, we discuss how to solve triple-A and resilience in smart manufacturing processes by adopting a service-based approach and automated reasoning techniques; this is not completely new, at least in the service computing literature, as seminal approaches go to [24] and more recently to [23]. An interesting survey on how planning techniques, not including MDPs, have been applied to service composition problems is [17].

7 Concluding Remarks

In this paper, we have proposed and discussed how automated reasoning techniques can be employed with the goal of adaptivity and resilience in smart manufacturing supply chains. The need for these techniques emerges when the number of resources involved and the constraints among them make a manual analysis from human experts unfeasible. In this sense, we have outlined and discussed the application of several techniques to a challenging use case concerning the manufacturing of integrated circuits. Our service-based approach, and its application to smart manufacturing, is another example of the challenges that service composition will have to cope in the next few years, as discussed in [1].

With respect to AI, in this paper, we only consider automated reasoning, without showing the potential of applying machine learning (especially reinforcement learning) techniques. If, on the one hand, machine learning approaches do not require any manual modeling effort, they usually require datasets to be trained, which are difficult to obtain in the smart manufacturing scenario, especially at a supply chain scale.

Also, this paper does not include approaches from classical numerical optimization techniques. These techniques are available in the form of very fast implementations. The main drawback is that modeling must be done in the form of equations, which are more complex to compose and validate with respect to formalisms employed in automated reasoning.

Acknowledgements

This work is partially funded by the ERC project WhiteMech (no. 834228), the PRIN project RIPER (no. 20203FFYLK), the Electrospindle 4.0 project (funded by Ministero per lo Sviluppo Economico, Italy, no. F/160038/01-04/X41), the Piano Nazionale di Ripresa e Resilienza (PNRR), Missione 4 "Istruzione e ricerca" – Componente 2 "Dalla ricerca all'impresa" – Investimento 1.3, funded by European Union - NextGenerationEU, and in particular by PE1 (CUP B53C22003980006) and PE11 (CUP B53C22004130001).

References

1. Aiello, M.: A Challenge for the Next 50 Years of Automated Service Composition. In: Service-Oriented Computing - 20th International Conference, ICSOC 2022, Proceedings. Lecture Notes in Computer Science, vol. 13740, pp. 635–643. Springer (2022)
2. Amari, S.V., McLaughlin, L., Pham, H.: Cost-effective condition-based maintenance using markov decision processes. In: RAMS'06. Annual Reliability and Maintainability Symposium, 2006. pp. 464–469. IEEE (2006)
3. Bicocchi, N., Cabri, G., Mandreoli, F., Mecella, M.: Dynamic digital factories for agile supply chains: An architectural approach. Journal of Industrial Information Integration **15**, 111–121 (2019)
4. Blythe, J.: Decision-theoretic planning. AI magazine **20**(2), 37–37 (1999)
5. Catarci, T., Firmani, D., Leotta, F., Mandreoli, F., Mecella, M., Sapio, F.: A conceptual architecture and model for smart manufacturing relying on service-based digital twins. In: 2019 IEEE International Conference on Web Services (ICWS). pp. 229–236. IEEE (2019)
6. Chen, H.Y., Das, A., Ivanov, D.: Building resilience and managing post-disruption supply chain recovery: Lessons from the information and communication technology industry. International Journal of Information Management **49**, 330–342 (2019)
7. Choo, B.Y., Adams, S.C., Weiss, B.A., Marvel, J.A., Beling, P.A.: Adaptive multiscale prognostics and health management for smart manufacturing systems. International journal of prognostics and health management **7** (2016)
8. Ciolek, D., D'Ippolito, N., Pozanco, A., Sardiña, S.: Multi-tier automated planning for adaptive behavior. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 30, pp. 66–74 (2020)
9. De Giacomo, G., Favorito, M.: Compositional approach to translate ltlf/ldlf into deterministic finite automata. In: ICAPS. pp. 122–130. AAAI Press (2021)
10. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M., Silo, L.: Digital twins composition in smart manufacturing via markov decision processes. Computers in Industry (2023)
11. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. pp. 854–860 (2013)
12. Dubey, R., Gunasekaran, A., Childe, S.J.: The design of a responsive sustainable supply chain network under uncertainty. The International Journal of Advanced Manufacturing Technology **80**, 427–445 (2015)

13. Dumas, M., Fournier, F., Limonad, L., Marrella, A., Montali, M., Rehse, J.R., Accorsi, R., Calvanese, D., De Giacomo, G., Fahland, D., et al.: Ai-augmented business process management systems: a research manifesto. ACM Transactions on Management Information Systems **14**(1), 1–19 (2023)
14. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)
15. Fox, M., Long, D.: Pddl2. 1: An extension to pddl for expressing temporal planning domains. Journal of artificial intelligence research **20**, 61–124 (2003)
16. Geffner, H.: Computational models of planning. Wiley Interdisciplinary Reviews: Cognitive Science **4**(4), 341–356 (2013)
17. Georgievski, I., Aiello, M.: Automated Planning for Ubiquitous Computing. ACM Comput. Surv. **49**(4), 63:1–63:46 (2017)
18. Ghallab, M., Nau, D., Traverso, P.: Automated planning and acting. Cambridge University Press (2016)
19. Groover, M.P.: Automation, production systems, and computer-integrated manufacturing. Pearson Education India (2016)
20. Helmert, M.: The fast downward planning system. Journal of Artificial Intelligence Research **26**, 191–246 (2006)
21. Hu, H., Jia, X., Liu, K., Sun, B.: Self-adaptive traffic control model with behavior trees and reinforcement learning for agv in industry 4.0. IEEE Transactions on Industrial Informatics **17**(12), 7968–7979 (2021)
22. Ivanov, D., Dolgui, A., Sokolov, B., Ivanova, M.: Literature review on disruption recovery in the supply chain. International Journal of Production Research **55**(20), 6158–6174 (2017)
23. Kaldeli, E., Lazovik, A., Aiello, M.: Domain-independent planning for services in uncertain and dynamic environments. Artif. Intell. **236**, 30–64 (2016)
24. Lazovik, A., Aiello, M., Papazoglou, M.P.: Planning and monitoring the execution of web service requests. In: Service-Oriented Computing - ICSOC 2003, First International Conference, Proceedings. Lecture Notes in Computer Science, vol. 2910, pp. 335–350. Springer (2003)
25. Lee, H.L., et al.: The triple-A supply chain. Harvard business review **82**(10), 102–113 (2004)
26. Macher, J.T., Mowery, D.C., Simcoe, T.S.: e-business and disintegration of the semiconductor industry value chain. Industry and Innovation **9**(3), 155–181 (2002)
27. Malburg, L., Hoffmann, M., Bergmann, R.: Applying mape-k control loops for adaptive workflow management in smart factories. Journal of Intelligent Information Systems pp. 1–29 (2023)
28. Marrella, A.: Automated planning for business process management. Journal on data semantics **8**(2), 79–98 (2019)
29. Marrella, A., Mecella, M., Sardina, S.: Intelligent process adaptation in the smartpm system. ACM Trans. on Intell. Systems and Technology **8**(2), 1–43 (2016)
30. Marrella, A., Mecella, M., Sardiña, S.: Supporting adaptiveness of cyber-physical processes through action-based formalisms. AI Commun. **31**(1), 47–74 (2018)
31. Paul, S.K., Chowdhury, P.: Strategies for managing the impacts of disruptions during covid-19: an example of toilet paper. Global Journal of Flexible Systems Management **21**, 283–293 (2020)
32. Puterman, M.L.: Markov Decision Processes (1994)
33. Stadtler, H.: Supply chain management: An overview. Supply chain management and advanced planning: Concepts, models, software, and case studies pp. 3–28 (2015)

34. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction (2018)
35. Wally, B., Vyskočil, J., Novák, P., Huemer, C., Šindelář, R., Kadera, P., Mazak-Huemer, A., Wimmer, M.: Leveraging iterative plan refinement for reactive smart manufacturing systems. IEEE Transactions on Automation Science and Engineering **18**(1), 230–243 (2020)