Cognitive Business Process Management for Adaptive Cyber-Physical Processes

Andrea Marrella(✉) and Massimo Mecella

Dipartimento di Ingegneria Informatica, Automatica e Gestionale,
Sapienza Università di Roma, via Ariosto 25, 00185 Rome, Italy
{marrella,mecella}@diag.uniroma1.it

Abstract. In the era of Big Data and Internet-of-Things (IoT), all real-world environments are gradually becoming cyber-physical (e.g., emergency management, healthcare, smart manufacturing, etc.), with the presence of connected devices and embedded ICT systems (e.g., smartphones, sensors, actuators) producing huge amounts of data and events that influence the enactment of the Cyber Physical Processes (CPPs) enacted in such environments. A Process Management System (PMS) employed for executing CPPs is required to automatically adapt its running processes to anomalous situations and exogenous events by minimising any human intervention at run-time. In this paper, we tackle this issue by introducing an approach and an adaptive Cognitive PMS that combines process execution monitoring, unanticipated exception detection and automated resolution strategies leveraging on well-established action-based formalisms in Artificial Intelligence, which allow to interpret the ever-changing knowledge of cyber-physical environments and to adapt CPPs by preserving their base structure.

Keywords: Cognitive business process management
Cyber-Physical Processes · Process adaptation and recovery
Situation calculus · IndiGolog · Automated planning

1 Introduction

In the last years, we have witnessed the emergence of new computing paradigms, such as Industry 4.0[1], Health 2.0 (e.g., cf. [1]) and mobile-based emergency management [2], in which the interplay of Internet-of-Things (IoT) devices, i.e., devices attached to the Internet, cloud computing, Software-as-a-Service (SaaS), and Business Process Management (BPM) create the so-called *cyber-physical environments* and give rise to the concept of *Cyber-Physical Systems* (CPSs).

[1] cf. H. Kagermann, W. Wahlster and J. Helbig: Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group, 2013, Frankfurt, http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf .

E. Teniente and M. Weidlich (Eds.): BPM 2017 Workshops, LNBIP 308, pp. 429–439, 2018.
https://doi.org/10.1007/978-3-319-74030-0_33

The role of CPSs is to monitor the *physical processes* enacted in cyber-physical environments, create a virtual copy of the physical world and make decentralized decisions, by introducing automated and intelligent support of workers in their increasingly complex work [3].

A relevant aspect in these environments lies in the fundamental role played by the processes orchestrating the different actors (software, humans, robots, etc.) involved in the CPS. We refer to these processes as *cyber-physical processes* (CPPs), whose enactment is influenced by user decision making and coupled with contextual data and knowledge production coming from the cyber-physical environment. According to [4], *Cognitive Process Management Systems* (CPMSs) are the key technology for supporting CPPs. A PMS is said to be *cognitive* when it involves additional processing constructs that are at a semantic level higher than those of conventional PMSs. These constructs are called *cognitive BPM constructs* and include data-driven activities, goals, and plans [4]. Their usage can open opportunities for new levels of automation for CPPs, such as - for example - *the automated synthesis of adaptation strategies at run-time exploiting solely the process knowledge and its expected evolution.*

During the enactment of CPPs, variations or divergence from structured reference models are common due to exceptional circumstances arising (e.g., autonomous user decisions, exogenous events, or contextual changes), thus requiring the ability to properly *adapt* the process behavior. *Process adaptation* can be seen as the ability of a process to react to exceptional circumstances (that may or may not be foreseen) and to adapt/modify its structure accordingly. Exceptions can be either *anticipated* or *unanticipated.* An anticipated exception can be planned at design-time and incorporated into the process model, i.e., a (human) process designer can provide an *exception handler* that is invoked during run-time to cope with the exception. Conversely, *unanticipated exceptions* refer to situations, unplanned at design-time, that may emerge at run-time and can be detected only during the execution of a process instance, when a mismatch between the computerized version of the process and the corresponding real-world process occurs. To cope with those exceptions, a PMS is required to allow *ad-hoc process changes* for adapting running process instances in a context-dependent way.

The fact is that, in cyber-physical environments, the number of possible anticipated exceptions is often too large, and traditional manual implementation of exception handlers at design-time is not feasible for the process designer, who has to anticipate all potential problems and ways to overcome them in advance [5]. Furthermore, anticipated exceptions cover only partially relevant situations, as in such scenarios many unanticipated exceptional circumstances may arise during the process instance execution. Therefore, the process designer often lacks the needed knowledge to model all the possible exceptions at the outset, or this knowledge can become obsolete as process instances are executed and evolve, by making useless her/his initial effort.

To tackle this issue, in this paper we summarize the main ideas discussed in [6] and introduce our work on SmartPM[2], a CPMS able to *automatically adapt CPPs at run-time* when *unanticipated exceptions* occur, thus requiring no specification of recovery policies at design-time. The general idea builds on the dualism between an *expected reality* and a *physical reality*: process execution steps and exogenous events have an impact on the physical reality and any deviation from the expected reality results in a mismatch to be removed to allow process progression.

To that end, we have resorted to three popular *action-based formalisms* and *technologies* from the field of Knowledge Representation and Reasoning (KR&R): situation calculus [9], IndiGolog [10], and automated planning [11,12]. We used the situation calculus logical formalism to model the underlying domain in which processes are to be executed, including the description of available tasks, contextual properties, tasks' preconditions and effects, and the initial state. On top of such model, we used the IndiGolog high-level agent programming language for the specification of the structure and control flow of processes. Importantly, we customized IndiGolog to monitor the online execution of processes and detect potential mismatches between the model and the actual execution. If an exception invalidates the enactment of the processes being executed, an external state-of-the-art classical planner is invoked to synthesise a recovery procedure to adapt the faulty process instance.

The choice of adopting action-based formalisms from the KR&R field is motivated by their ability to provide the right *cognitive* level needed when dealing with dynamic situations in which data (values) play a relevant role in system enactment and automated reasoning over the system progress. In the field of BPM, many other formalisms (in particular Petri Nets-based and process algebras) have been successfully adopted for process management, but all of them are somehow based on synthesis techniques of the control-flow, when considering their automated reasoning capabilities. This implies the level of abstraction over dealing with data and dynamic situations is fairly "raw", when compared with KR&R methods in which automated reasoning over data values and situations is much more developed [9,13,14]. The choice of KR&R technologies allowed us to develop a principled, clean and simple-to-manage framework for process adaptation based on relevant data manipulated by the process, without compromising efficiency and effectiveness of the proposed solution.

The rest of the paper is organized as follow. Section 2 introduces conceptual architecture for CPMSs that manage CPPs. Such an architecture is then instantiated in the SmartPM approach and system outlined in Sect. 3. Finally Sect. 4 concludes the paper by discussing our approach in the larger context and presenting possible future evolutions.

[2] The reader interested to the very technical details may refer to [6–8].

2 A Conceptual Architecture for Managing CPPs

CPSs are having widespread applicability and proven impact in multiple areas, like aerospace, automotive, traffic management, healthcare, manufacturing, emergency management [15]. According to [16], any physical environment which contains computing-enabled devices can be considered as a cyber-physical environment.

The trend of managing CPPs, i.e., processes enacted in cyber-physical environments, has been fueled by two main factors. On the one hand, the recent development of powerful mobile computing devices providing wireless communication capabilities have become useful to support mobile workers to execute tasks in such dynamic settings. On the other hand, the increased availability of sensors disseminated in the world has lead to the possibility to monitor in detail the evolution of several real-world objects of interest. *The knowledge extracted from such objects allows to depict the contingencies and the context in which processes are carried out, by providing a fine-grained monitoring, mining, and decision support for them.*

We devise in the following a conceptual architecture to concretely build an adaptive CPMS in cyber-physical environments. The management of a CPP requires additional challenges to be considered if compared with a traditional "static" business process. On the one hand, there is the need of representing explicitly real-world objects and technical aspects like device capability constraints, sensors range, actors and robots mobility, etc. On the other hand, since cyber-physical environments are intrinsically "dynamic", a CPMS providing real-time monitoring and automated adaptation features during process execution is required.

To this end, the role of the data perspective becomes fundamental. Data, including information processed by process tasks as well as contextual information, are the main driver for triggering process adaptation, as focusing on the control flow perspective only - as traditional PMSs do - would be insufficient. In fact, in a cyber-physical environment, a CPP is genuinely knowledge and data centric: its control flow must be coupled with contextual data and knowledge production and process progression may be influenced by user decision making. This means that traditional imperative models have to be extended and complemented with the introduction of specific *cognitive constructs* such as *data-driven activities* and *declarative elements* (e.g., tasks preconditions and effects) which enable a precise description of data elements and their relations, so as to go beyond simple process variables, and allow establishing a link between the control flow and the data perspective.

Starting from the above considerations, coupled with the experience gained in the area and lessons learned from several projects involving CPSs, we have devised a conceptual architecture to build a CPMS for the management of CPPs, which supports the so-called *Plan-Act-Learn* cycle for cognitively-enabled processes [4]. As shown in Fig. 1, we identified 5 main architectural layers that we present in a bottom-up fashion.

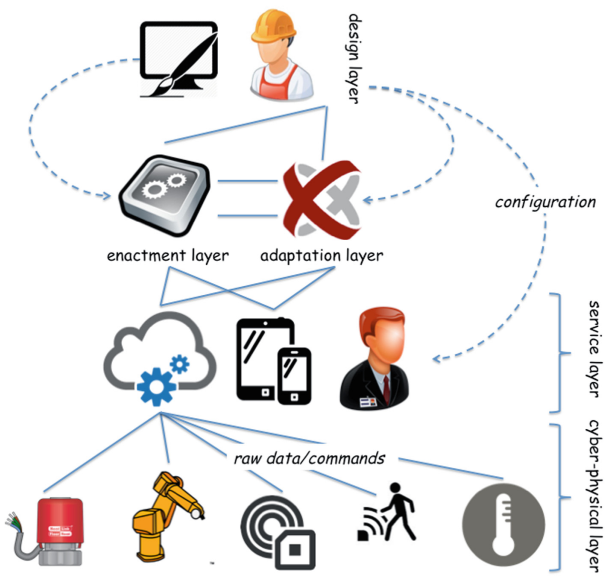

Fig. 1. A conceptual architecture for CPPs.

The ***cyber-physical layer*** consists mainly of two classes of physical components: *(i)* sensors (such as GPS receivers, RFID chips, 3D scanners, cameras, etc.) that collect data from the physical environment by monitoring real-world objects and *(ii)* actuators (robotic arms, 3D printers, electric pistons, etc.), whose effects affect the state of the physical environment. The cyber-physical layer is also in charge of providing a physical-to-digital interface, which is used to transform *raw* data collected by the sensors into machine-readable events, and to convert *high-level* commands sent by the upper layers into *raw* instructions readable by the actuators. The cyber-physical layer does not provide any intelligent mechanism neither to clean, analyse or correlate data, nor to compose high-level commands into more complex ones; such tasks are in charge of the uppers layer.

On top of the cyber-physical layer lies the ***service layer***, which contains the set of services offered by the real-world entities (software components, robots, agents, humans, etc.) to perform specific process tasks. In the service layer, available data can be aggregated and correlated, and high-level commands can be orchestrated to provide higher abstractions to the upper layers. For example, a smartphone equipped with an application allowing to sense the position and the posture of a user is at this layer, as it collects the raw GPS, accellerometer and motion sensor data and correlates them to provide discrete and meaningful information.

On top of the service layer, there are two further layers interacting with each other. The ***enactment layer*** is in charge of *(i)* enacting complex processes by deciding which tasks are enabled for execution, *(ii)* orchestrating the different available services to perform those tasks and *(iii)* providing an execution monitor to detect the anomalous situations that can possibly prevent the correct execution of process instances. The execution monitor is responsible for deciding if process adaptation is required. If this is the case, the ***adaptation layer*** will provide the required algorithms to *(i)* reason on the available process tasks and contextual data and to *(ii)* find a recovery procedure for adapting the process instance under consideration, i.e., to re-align the process to its expected behaviour. Once a recovery procedure has been synthesized, it is passed back to the enactment layer for being executed.

Finally, the ***design layer*** provides a GUI-based tool to define new process specifications. A process designer must be allowed not only to build the process control flow, but also to explicitly formalize the data reflecting the contextual knowledge of the cyber-physical environment under study. It is important to underline that data formalization must be performed without any knowledge of the internal working of the physical components that collect/affect data in the cyber-physical layer. To link tasks to contextual data, the GUI-based tool must go beyond the classical "task model" as known in the literature, by allowing the process designer to explicitly state what data may constrain a task execution or may be affected after a task completion. Finally, besides specifying the process, configuration files should also be produced to properly configure the enactment, the services and the sensors/actuators in the bottom layers.

3 The SmartPM Approach and System

SmartPM (Smart Process Management) is an approach and an adaptive CPMS implementing a set of techniques that enable to automatically adapt process instances at run-time in the presence of unanticipated exceptions, without requiring an explicit definition of handlers/policies to recover from tasks failures and exogenous events. SmartPM adopts a *layered service-based* approach to process management, i.e., *tasks are executed by services*, such as software applications, humans, robots, etc. Each task can be thus seen as a single step consuming input data and producing output data.

To monitor and deal with exceptions, the SmartPM approach leverages on [17]'s technique of adaptation from the field of agent-oriented programming, by specializing it to our CPP setting (see Fig. 2). We consider adaptation as *reducing the gap* between the *expected reality* **Exp**, the (idealized) model of reality used by the CPMS to reason, and the *physical reality* **Phy**, the real world with the actual conditions and outcomes. While **Phy** records what is *concretely* happening in the real environment during a process execution, **Exp** reflects what it is *expected* to happen in the environment. Process execution steps and exogenous events have an impact on **Phy** and any deviation from **Exp** results in a mismatch to be removed to allow process progression. At this point, a state-of-the-art automated planner is invoked to synthesise a recovery procedure that adapts the faulty process instance by removing the gap between the two realities.

To realize the above approach, the implementation of SmartPM covers the modeling, execution and monitoring stages of the CPP life-cycle. To that end, the architecture of SmartPM relies on five architectural layers.

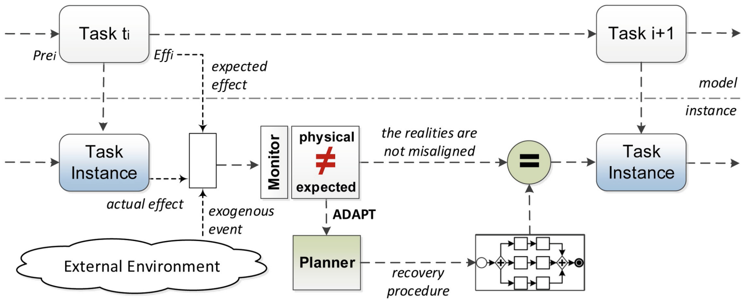

Fig. 2. An overview of the SmartPM approach.

The ***design layer*** provides a graphical editor developed in Java that assists the process designer in the definition of the process model at design-time. Process knowledge is represented as a *domain theory* that includes all the contextual information of the domain of concern, such as the people/services that may be involved in performing the process, the tasks, the data and so forth. Data are represented through some *atomic terms* that range over a set of *data objects*, which depict entities of interest (e.g., capabilities, services, etc.), while atomic terms can be used to express properties of domain objects (and relations over objects). *Tasks* are collected in a repository and are described in terms of *preconditions* - defined over atomic terms - and *effects*, which establish their expected outcomes. Finally, a process designer can specify which *exogenous events* may be caught at run-time and which atomic terms will be modified after their occurrence. Once a valid domain theory is ready, the process designer uses the graphical editor to define the process control flow through the standard BPMN notation among a set of tasks selected from the tasks repository.

The ***enactment layer*** is in charge of managing the process execution. First of all, the domain theory specification and the BPMN process are automatically translated into situation calculus [9] and IndiGolog [10] readable formats. Situation calculus is used for providing a declarative specification of the domain of interest (i.e., available tasks, contextual properties, tasks preconditions and effects, what is known about the initial state). Then, an *executable model* is obtained in the form of an IndiGolog program to be executed through an IndiGolog engine. To that end, we customized an existing IndiGolog engine[3] to *(i)* build a physical/expected reality by taking the initial context from the external environment; *(ii)* manage the process routing; *(iii)* collect exogenous events from the external environment; *(iv)* monitor contextual data to identify changes or events which may affect process execution. Once a task is ready for being executed, the IndiGolog engine assigns it to a proper process participant (that could be a software, a human actor, a robot, etc.) that provides all the required capabilities for task execution.

[3] http://sourceforge.net/projects/indigolog/.

The ***service layer*** acts as a middleware between process participants, the enactment layer and the cyber-physical layer. Specifically, in the service layer, process participants interact with the engine through a *Task Handler*, an interactive GUI-based software application realized for Android devices that supports the visualization/execution of assigned tasks by selecting an appropriate outcome. Possibly such an Android application can exploit sensors and actuators (e.g., an Arduino board connected through Bluetooth, as currently realized in our implementation), thus effectively offering services over the *cyber-physical layer*. Every step of the task life cycle - ranging from the assignment to the release of a task - requires an interaction between the IndiGolog engine and the task handlers. The communication between the IndiGolog engine and the task handlers is mediated by the *Communicator Manager* component (which is essentially a web server) and established using the Google Cloud Messaging service.

To enable the automated synthesis of a recovery procedure, the ***adaptation layer*** relies on the capabilities provided by a planner component (the LPG-td planner [18]), which assumes the availability of a classical planning problem, i.e., an initial state and a goal to be achieved, and of a planning domain definition that includes the actions to be composed to achieve the goal, the domain predicates and data types. Specifically, if process adaptation is required, we translate *(i)* the domain theory defined at design-time into a planning domain, *(ii)* the physical reality into the initial state of the planning problem and *(iii)* the expected reality into the goal state of the planning problem. The planning domain and problem are the input for the planner component. If the planner is able to synthesize a recovery procedure δ_a, the *Synchronization* component combines δ' (which is the remaining part of the faulty process instance δ still to be executed), with the recovery plan δ_a, builds an adapted process $\delta'' = (\delta_a; \delta')$ and converts it into an executable IndiGolog program so that it can be enacted by the IndiGolog engine. Otherwise, if no plan exists for the current planning problem, the control passes back to the process designer, who can try to manually adapt the process instance.

The ***cyber-physical layer*** is tightly coupled with the physical components available in the domain of interest. Since the IndiGolog engine can only work with defined discrete values, while data gathered from physical sensors have naturally continuous values, the system provides several web tools that allow process designers to associate some of the data objects defined in the domain theory with the continuous data values collected from the environment. For example, we developed several web tools to associate the data collected from sensors (GPS, temperature, noise level, etc.) to discrete values. We provided a concrete example of a location web tool that allows process designers to mark areas of interest from a real map and associate them to discrete locations. The mapping rules generated are then saved into the Communication Manager and retrieved at run-time to allow the matching of the continuous data values collected by the specific sensor into discrete data objects.

4 Concluding Remarks

We are at the beginning of a profound transformation of BPM due to advances in AI and Cognitive Computing [4]. Cognitive systems offer computational capabilities typically based on large amount of data, which provide cognition power that augment and scale human expertise. The aim of the emergent field of cognitive BPM is to offer the computational capability of a cognitive system to provide analytical support for processes over structured and unstructured information sources. The target is to provide proactivity and self-adaptation of the running processes against the evolving conditions of the application domains in which they are enacted.

In this direction, our paper summarizes the most interesting results reported in [6], which have been devoted to the realization of a general approach, a concrete framework and a CPMS implementation, called SmartPM, for automated adaptation of CPPs. Our purpose was to demonstrate that the combination of procedural and imperative models with cognitive BPM constructs such as data-driven activities and declarative elements, along with the exploitation of techniques from the field of AI such as situation calculus, IndiGolog and classical planning, can increase the ability of existing PMSs of supporting and adapting CPPs in case of unanticipated exceptions.

Existing approaches dealing with unanticipated exceptions typically rely on the involvement of process participants at run-time, so that authorized users are allowed to manually perform structural process model adaptation and ad-hoc changes at the instance level. However, CPPs demand a more flexible approach recognizing the fact that in real-world environments process models quickly become outdated and hence require closer interweaving of modeling and execution. To this end, the adaptation mechanism provided by SmartPM is based on execution monitoring for detecting failures and context changes at run-time, without requiring to predefine any specific adaptation policy or exception handler at design-time (as most of the current approaches do).

From a general perspective, our planning-based automated exception handling approach should be considered as complementary with respect to existing techniques, acting as a "bridge" between approaches dealing with anticipated exceptions and approaches dealing with unanticipated exceptions. When an exception is detected, the run-time engine may first check the availability of a predefined exception handler, and if no handler was defined it can rely on an automated synthesis of the recovery process. In the case that our planning-based approach fails in synthesizing a suitable handler (or an handler is generated but its execution does not solve the exception), other adaptation techniques need to be used. For example, if the running process provides a well-defined intended goal associated to its execution, we could resort to the van Beest's work [19] and do planning from first-principle to achieve such a goal. Conversely, if no intended goal is associated to the process, a human participant can be involved, leaving her/him the task of manually adapting the process instance. Future work will include an extension of our approach to "stress" the assumptions imposed by the usage of classical planning techniques for the synthesis of the recovery procedure, which frame the scope of applicability of the approach for addressing more expressive problems, including incomplete information, preferences and multiple task effects.

The current implementation of SmartPM is developed to be effectively used by process designers and practitioners[4]. Users define processes in the well-known BPMN language, enriched with semantic annotations for expressing properties of tasks, which allow our interpreter to derive the IndiGolog program representing the process. Interfaces with human actors (such as specific graphical user applications in Java) and software services (through Web service technologies) allow the core system to be effectively used for enacting processes. Although the need to explicitly model process execution context and annotate tasks with preconditions and effects may require some extra modeling effort at design-time (also considering that traditional process modeling efforts are often mainly directed to the sole control flow perspective), the overhead is compensated at run-time by the possibility of automating exception handling procedures. While, in general, such modeling effort may seem significant, in practice it is comparable to the effort needed to encode the adaptation logic using alternative methodologies like happens, for example, in rule-based approaches.

Acknowledgments. This work is partly supported by the projects Social Museum and Smart Tourism (CTN01_00034_23154), NEPTIS (PON03PE_00214_3), RoMA (SCN_00064), and by the Sapienza project "Data-aware Adaptation of Knowledge-intensive Processes in Cyber-Physical Domains through Action-based Languages".

References

1. Cossu, F., Marrella, A., Mecella, M., Russo, A., Bertazzoni, G., Suppa, M., Grasso, F.: Improving operational support in hospital wards through vocal interfaces and process-awareness. In: 25th International Symposium on Computer-Based Medical Systems (CBMS). IEEE (2012)
2. Humayoun, S.R., Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Bortenschlager, M., Steinmann, R.: Designing mobile systems in highly dynamic scenarios: the WORKPAD methodology. Knowl. Technol. Policy **22**(1), 25–43 (2009)
3. Seiger, R., Keller, C., Niebling, F., Schlegel, T.: Modelling complex and flexible processes for smart cyber-physical environments. J. Comput. Sci. **10**, 137–148 (2014)
4. Hull, R., Motahari Nezhad, H.R.: Rethinking BPM in a cognitive world: transforming how we learn and perform business processes. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_1
5. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer, Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-30409-5
6. Marrella, A., Mecella, M., Sardiña, S.: Supporting adaptiveness of cyber-physical processes through action-based formalisms. AI Communications (2017, to appear)

[4] See: http://www.dis.uniroma1.it/~smartpm.

7. Marrella, A., Mecella, M., Sardina, S.: SmartPM: an adaptive process management system through situation calculus, indigolog, and classical planning. In: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning, KR 2014 (2014)
8. Marrella, A., Mecella, M., Sardiña, S.: Intelligent process adaptation in the SmartPM system. ACM TIST **8**(2), 1–25 (2017)
9. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
10. De Giacomo, G., Lespérance, Y., Levesque, H., Sardina, S.: IndiGolog: a high-level programming language for embedded reasoning agents. In: El Fallah Seghrouchni, A., Dix, J., Dastani, M., Bordini, R. (eds.) Multi-Agent Programming: Languages, Tools and Applications. Tools and Applications, Springer, Boston (2009). https://doi.org/10.1007/978-0-387-89299-3_2
11. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco (2004)
12. Geffner, H., Bonet, B.: A Concise Introduction to Models and Methods for Automated Planning. Morgan & Claypool Publishers, San Rafael (2013)
13. Reichgelt, H.: Knowledge Representation: An AI perspective. Greenwood Publishing Group Inc., Westport (1991)
14. Brachman, R., Levesque, H.: Knowledge Representation and Reasoning. Elsevier, Amsterdam (2004)
15. Rajkumar, R.R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: Proceedings of the 47th Design Automation Conference, DAC 2010, pp. 731–736. IEEE (2010)
16. Lee, E.A.: Cyber physical systems: design challenges. In: Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC 2008, pp. 363–369. IEEE (2008)
17. De Giacomo, G., Reiter, R., Soutchanski, M.: Execution monitoring of high-level robot programs. In: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning, KR 1998, pp. 453–465 (1998)
18. Gerevini, A., Saetti, A., Serina, I., Toninelli, P.: LPG-TD: a fully automated planner for PDDL2.2 domains. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling, ICAPS 2004 (2004)
19. van Beest, N.R., Kaldeli, E., Bulanov, P., Wortmann, J.C., Lazovik, A.: Automated runtime repair of business processes. Inf. Syst. **39**, 45–79 (2014)