ADAPT: an Automatic Diagnosis of Activity and Processes in auTomation environments

Igor Azkarate Fernández[1], Aitor Aguirre Ortuzar[1], Josu Uranga Andrés[2], and Luka Eciolaza Echeverría[1]
Department of Electronics and Computing
[1]{iazkarate,aaguirre,leciolaza}@mondragon.edu
[2]josu.uranga@alumni.mondragon.edu
Mondragon Unibertsitatea, Arrasate-Mondragón, Spain

***Abstract*—In an automated industrial environment, a large volume of data and signals is available, both from sensors and actuators in machinery and from the interaction with operators and users. Operation diagnosis can have multiple applications from a learning point of view (e.g. staff training) or in terms of process assessment. This work proposes a methodology for implementing an Intelligent System by means of any interactive system connected through OPC UA standard. A digital twin of the process supports configuration and validation, prior to commissioning. Activity is interpreted and diagnosed according to the context in which it occurs. Step order in sequence, step duration and sequence duration are analyzed in a use case based on a PLC-controlled robotic cell in which it is operated both in automatic and manual mode for adjusting a linear table's positioning.**

***Index Terms*—Activity recognition, Artificial Intelligence algorithms, context awareness, digital twin, emulation, fault diagnosis, manufacturing automation**

I. Introduction

Intelligent industrial environments within the context of Industry 4.0 are based on their digitalization. This requires acquiring more and more data in order to monitor and control different aspects of processes. The use of these process data is enabling to develop systems with higher level of autonomy, in order to ensure safe process operation, product quality control and integrity assessment of different equipment.

For systems to be endowed with greater autonomy, they must be able to sense and interpret their physical environment and adapt their behaviour accordingly. This capability could be referred to as a context-aware system.

Literature reflects works related to context-awareness, with applications in industrial areas such as advanced material handling [1], or monitoring, where [2] integrates context data and existing plant information to provide dynamically only the most relevant information. In addition, a tool developed by [3] makes available a context-aware (re)configuration of field devices of a system.

The vast majority of industrial processes have clear procedures defined for their correct operation. For every different role and tasks involved with the process (i.e.: normal automatic operation, manual adjusting, maintenance, etc.), different procedures define sequences of actions that must be fulfilled in order to perform the operation correctly.

The aim of this document is to propose a framework for the diagnosis of industrial automated processes, by interpreting, from process data, the context and the fulfillment of procedures associated with tasks or operations. This capability could be used for a variety of things, such as: (i) verification of correct task performing with respect to a predefined ideal procedure, (ii) extraction of ideal procedure based on expert operator, (iii) novice operator training and guidance, etc.

A digital twin (DT) makes it possible to work with an emulated process, without interfering with the operation of the real one. All types of situations are reproduced without stopping production or endangering equipment or people. It can support development or configuration tasks, as well as training applications for non-expert employees.

OPC UA, which is based on the client-server principle and is vendor-independent, is the used industrial communication protocol. It enables transparent reading and writing of variables into controller memory, usually a programmable logic controller (PLC). Latest generation devices include the capability of an OPC UA server, as it is considered the Industry 4.0 standard.

After this background review and the definition of the contibution that is the core of this work, the proposed approach is detailed. A use case is presented based on a robotic cell that identifies and classifies drone housings, prior development against a DT of it. Finally, the results achieved and the conclusions reached are shown, proposing possible applications and future work.

II. Basis of the framework

ADAPT framework is based on ULISES metamodel and its runtime kernel [4]. The basis is a domain-independent metamodel inspired by the cognitive process that real domain experts or tutors (e.g. in educational scenarios) carry out while supervising or tutoring a learner during a process: experts first perceive facts or observations happening in the environment. Then, they interpret those observations in order to identify actions (i.e. they unconsciously assign meaning to their observations in the context of the domain) being carried out, and once they have processed this information, they make a diagnosis, which involves detecting and/or correcting mistakes. ULISES's runtime kernel, composed by Observation,

Interpretation and Diagnosis subsystems, defines a metamodel divided into three logical levels that generically represent the same process: observation, interpretation and diagnosis levels.

A. Observation level

This level contains the necessary elements for specifying the observable facts that are interesting from an educational or process diagnosis point of view. These elements are called observations and they represent events that take place during an interval of time in the Interactive System (IS). For the specific case of ADAPT, the IS to be integrated will be an OPC UA server. The objective of the observation level is to specify how the observation subsystem has to transform the data streams in the ISs into observations. If it was received information of markers that a worker is wearing, a right knee observation could be generated. Additionally, observations may have properties. For instance, the lifting right knee observation can have properties like lifting speed, movement on the vertical axis, etc. Observations are the primitives for the interpretation and diagnosis levels. Therefore, the observation subsystem's job is to fuse the inputs from the interactive system in order to transform them into observations and to update them synchronously for the two upper subsystems: interpretation and diagnosis subsystems.

B. Interpretation level

Interpretation level describes generically how to recognize ISs with such accuracy that the diagnosis subsystem is able to determine whether the actions are correct or incorrect and its reasons. To that aim, it is essential to know the context where the actions are happening. For that reason, the interpretation level defines two elements that are generated from observations using the constraint modelling paradigm [5]:

- Step: this represents an action that takes place over an interval of time and that will be diagnosed. A step defines how the ULISES runtime kernel will analyze observations to interpret when a step is being carried out. Considering that observations are durative, temporal relations between them are described. It is important to point up that a step must not be interpreted based on the correctness or incorrectness of an action. What this level must do is to interpret actions when they are carried out, regardless of their correction.
- Situation: this element represents the context where the actions are being executed. For example, if an IS holds a driving simulator and a lane change is wanted to be evaluated when overtaking, it is necessary to know if the driver is an overtaking situation. If the driver is leaving a lane because he is approaching an exit, that action would not be relevant in that context.

C. Diagnosis level

The objective of this level is to manage the elements that will allow an appropriate diagnosis of the activity to be generated. ULISES's diagnosis level allows different diagnostic components like constraint-based diagnostic modules to be integrated and communicate with external diagnostic systems like DETECTive [6]. ULISES provides the educational or process evalution components of the built system with real-time diagnostic results or final evaluation results as needed. The diagnostic results contain information at different levels: the correctness of situations, the correctness of every step within each situation and other information such satisfied and non-satisfied conditions within the steps.

D. ULISES runtime kernel

The ULISES runtime kernel is designed following multiagent arquitecture. Multiagent systems (MAS) have their origin in distributed artificial intelligence, but have been widely deployed in automation in the last decade [7] [8]. In terms of communication between agents, ULISES is based on the FIPA standard of interoperability [9]. It provides communication between any external service and applications and the three subsystems that the runtime kernel is composed of: observation, interpretation and diagnosis subsystems. Each subsystem of ULISES communicates with the others mainly through the standard subscribe, request and query protocols. Therefore, each agent can offer and ask for the information it needs just when it is required. Figure 1 shows the relation between ULISES's agents and the rest of the interactive intelligent system:

- Input-output agents: they allow the ULISES runtime kernel to communicate with user interfaces or other services, for example, with PATH [10]. This tool is both an authoring and monitoring system that provides domain experts with a way to create knowledge models (interpretation and diagnostic models) that are necessary for the framework.
- Observation agent: this encapsulates the observation subsystem and it is in charge of gathering OPC UA server's data via the input-output agents and transforming them into observations for the interpretation and diagnosis subsystems.
- Listener agents: they hold the communication between the IS. In ADAPT's case, OPC UA server(s) and the observation agent. It should be noted that there can be several agents in case of distributed ISs.
- Interpreter agent: this receives the observations that it is subscribed to, and then it generates the steps and situations that it detects by using the interpretation model.
- Diagnosis agent: this agent coordinates the diagnosis of the steps and situations received from the interpreter agent. The diagnostic process can be carried out internally (with specific diagnostic components integrated with the diagnosis subsystem via plugins) or externally (for example with DETECTive, in which case, wrapper plugins handle the communication between DETECTive and the diagnosis subsystem).

III. Methodology

The main objective of the observation level is to provide the necessary elements to describe how to observe the facts

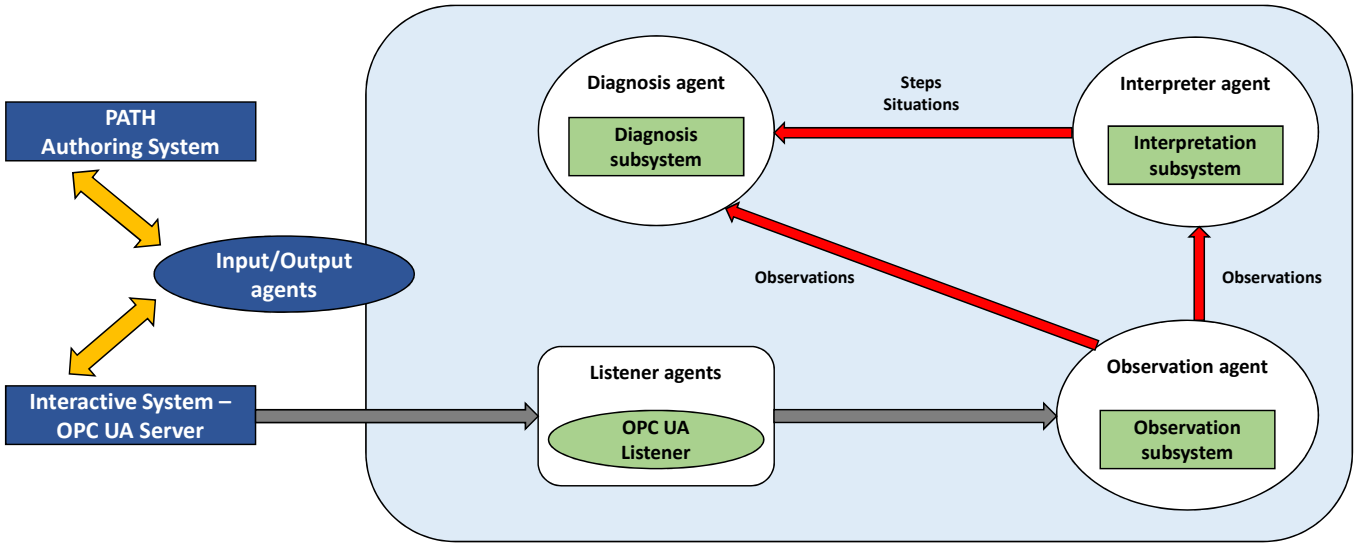


Fig. 1: Interaction between the ADAPT agents.

that are being collected by the OPC UA server and that are interesting in the process. Those facts will be the primitives for the higher levels, which will describe the activity of the process (that is interesting from an educational point of view) in real time. For example, if it is wanted to know if a part is holding a classification process in a specific period of time, it should be observed if the part has reached the camera and that the camera has detected it. As ULISES framework is a generic framework, the observation level cannot predetermine which are the characteristics of the IS. This is the reason why every system built with this framework needs an integration in the observation level. This section describes how this integration is carried out within ADAPT.

A. *Framework integration with OPC UA*

Observation subsystem's purpose is to gather streams that are coming from the OPC UA server and to turn them into observations. The two main elements that participate in this transformation process are the OPC UA listener and observer agents.

- OPC UA listener agent: this is the agent that communicates the framework with the OPC UA server. It publishes when a new cycle starts and ends and communicates the stream of data that has received from the OPC UA server. This agent contains an OPC UA client controller that stablishes connection with the external OPC UA server and creates subscriptions in the server based on the observations that have been predefined in the observation model. Once all data has been gathered, it is published for all the agents that are subscribed to this information.
- OPC UA observer agent: it is the responsible for converting the stream of data in actual observations. It receives the necessary data via a subscription to the OPC UA listener agent, and sends this data to the OPC UA observer. The observer implements an OPC UA client to generate observations in each cycle. When the evaluation and transformation of the streams of data is finished, observer agent notifies to the interpreter and diagnosis agents that observations have been updated. This observations are stored in a configuration file and are defined based on the interpretation and diagnosis that is intended to carry out for the process.

B. *Interpretation and diagnosis levels*

Once observations have been generated, the interpretation subsystem or interpreter discerns which steps and situations are being executed in the IS. Those steps and situations are modelled using a constraint based approach [4]. That is to say, user establishes constraints to define those two elements and in each cycle, those constraints are evaluated. If constraints are satisfied, steps and situations are created and this information is sent to the upper diagnosis subsystem.

Regarding the diagnosis subsystem, it processes the steps and situations that are detected by interpreter. For each step and situation, a solution is defined, which is diagnosed based on what has been defined in the diagnosis model. This model is also based on a constraint modelling technique. In a nutshell, a solution specifies how those steps and situations should be carried out. For example, if a simulation where a human takes part on is being executed, its activity could be evaluated in real time. Likewise, if a process is being watched, the correct functioning of each situation and step of the process can be specified. Therefore, this approach makes ADAPT a suitable solution for OPC UA based controllers and DTs. This is a commonly used standard for connectivity between PLCs (real or emulated) with industrial process DTs, in order to control software testing.

For the integration of devices and applications that are part of the framework by means of the OPC UA standard, the control device, usually a PLC, would act as a server, and

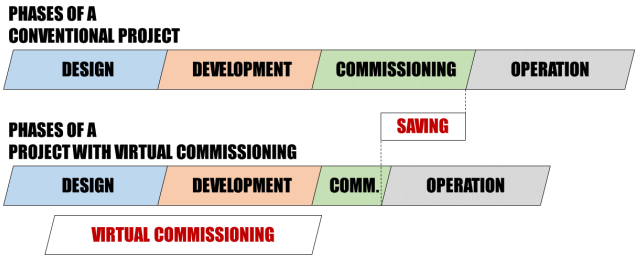


Fig. 2: Conventional project vs. project with DT-based VC.

the platform as a client, and there could be more elements, such as human-machine interface (HMI) panels or supervisory control and data acquisition (SCADA) systems. Another part can be a process emulated through its DT, so that the platform adapted to the process is tested and validated without affecting its operation. Once the platform is validated, it is integrated into the normal operation of the industrial process.

In both phases, the platform accesses the process and interaction data from the PLC through the aforementioned standard. From them, tasks and procedures are identified.

IV. Digital twins

The emergence of industrial process emulation tools opens up the possibility of supporting the development and validation of a framework of the above-mentioned characteristics without interfering with the operation of the real system. DTs attempt to imitate the performance of a system to do the same job and produce the same results. As opposed to simulation [11], it requires the real control system to work properly: a PLC or a robot.

The first and currently main application of these software tools in industrial automation is the control device software testing or virtual commissioning (VC), prior to the assembly of the equipment [12] and commissioning phase (see Figure 2). This is done by connecting the real (hardware-in-the-loop, HIL) or emulated PLC (software-in-the-loop, SIL) to the virtual model by means of OPC UA [13]. The use of a DT for VC brings benefits in terms of quality of the work developed [13], reduction of commissioning time [14] [15], less costs in this last phase [16], fewer risks and more flexibility.

But a DT can support all phases of the lifecycle of an automated system. Literature reflects use cases such as the design of a production line analyzing results of modifications without making them [17], the improvement and acceleration of the overall engineering process [18], or the continuous testing of programs to improve both model and control software [19]. All this is framed as integrated virtual commissioning (IVC).

For the development of this work, it is assumed that a DT of an automated process can support the development and validation of a diagnostic tool, without affecting process operation and prior to its integration with this, if necessary.

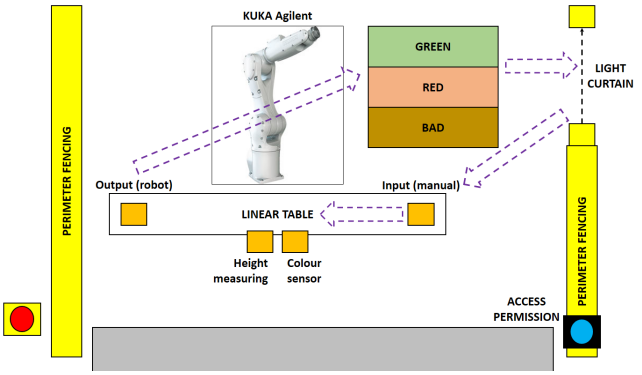


Fig. 3: Layout.

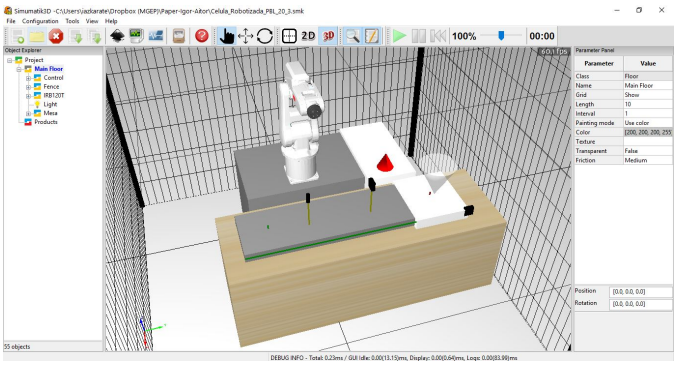


Fig. 4: Digital twin.

V. Use case

The use case presented below was intended to diagnose the activity in an automated system in which there is some interaction with the operator.

A. Process to be diagnosed

The system under study is a robotic cell (which layout is shown in Figure 3) placed in a laboratory of Mondragon Unibertsitatea (MU), and its DT (Figure 4) [20].

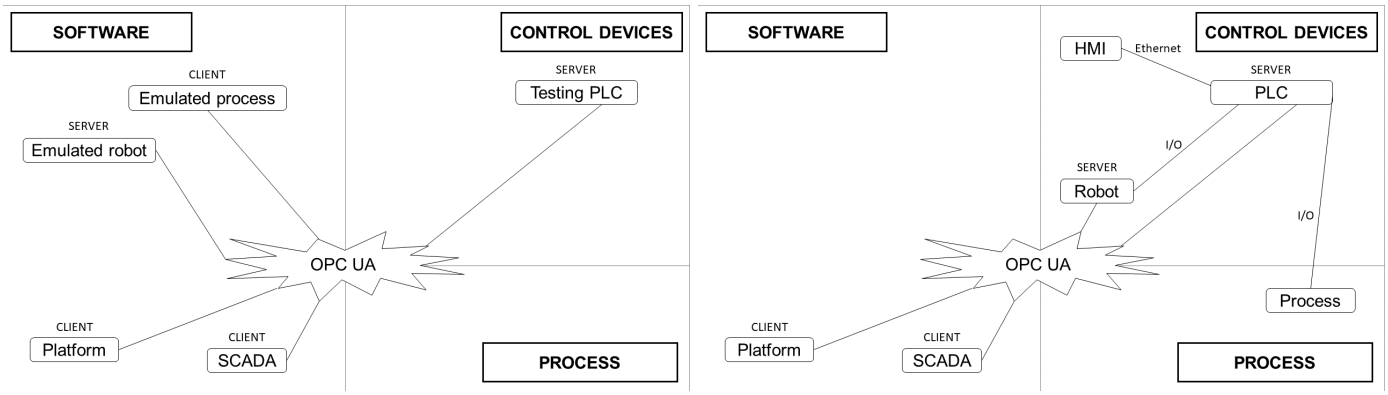
In this process, the identification, quality control and classification of drone housings is carried out. This robotic cell consists of the following elements:

- A table with a KUKA Agilent robot as a manipulator.
- A linear table controlled by a servo-variator for the transport of workpieces along four positions: manual supply, height measurement, chromatic sensor and output, where the robot picks up the housing for transfer to the corresponding area.
- A perimeter fencing and a light curtain, which makes it posible to safely carry out the manual input and output of workpieces.

B. Resources

In addition to the process described above, the following devices and software packages were available for its control:

- A Siemens S7-1500 series PLC.



(a) HIL configuration for platform setup.

(b) Configuration in the operating phase.

Fig. 5: Platform configuration.

- A Siemens HMI TP700 Comfort Panel.
- A PC with this software tools, among others:
  - Wonderware InTouch 2014 R2, a SCADA system.
  - KEPServerEX 6.2 connectivity platform with Wonderware OPC UA Client Service.

Furthermore, the entire control system already described was duplicated for the platform testing. A identical PLC was used, which allowed HIL testing. The aforementioned software tools were installed on a laptop, as well as the following ones:

- TIA Portal V15.1: development environment for Siemens control devices, for transferring the cell program to the PLC, emulating the HMI and monitoring variables.
- Simumatik3D® v1.0.3 (S3D), which is an emulation tool that enables users to test PLC and industrial robot programs in an easy way. A DT of the cell under study, previously developed, was available.
- Simumatik3D® OPC UA Server v0.1.6, as the robot does not incorporate a server, in order to emulate it in the DT.
- ABB's simulation and offline programming software, RobotStudio 2019.1.
- .NET Based OPC UA Client
- The framework, which includes the methodology and software to create intelligent interactive systems. PATH, e.g., is an authoring tool that provides the necessary knowledge models to be added to the framework.

C. Configuration

The configuration used for the integration and testing of the framework corresponding to the case study is shown in Figure 5a, as proposed in subsection III-A. The PLC (real) for testing and the robot (emulated) were configured as OPC UA servers, and the SCADA system, the DT of the process, and the platform as clients. The latter accessed the necessary data by reading variables from the PLC.

Once the platform was validated against the emulated process, it was integrated with the process in normal operation, like in Figure 5b. The robot was no longer emulated, and the signal exchange with the cell PLC was via digital I/O.

D. Interpretation and diagnosis phase

The robotic cell operated in two modes, which were represented in the interpretation and diagnosis models as automatic situation and manual situation. In this scenario, the diagnosis subsystem needed to ensure that within the automatic situation (process being run in automatic mode), (i) all the steps defined were carried out and (ii) all the steps were executed under a defined duration. The excessive duration of one of the steps may be due to the degradation of one of the components or mechanisms. In manual mode the focus of ADAPT was educational, diagnosing whether users were correctly following the established steps to complete a sequence, and in an appropriate time.

Note the following issues:

- The diagnosis of tasks that are over time or steps that are not executed in the correct order is based on Allen's grammar [21].
- The same step, carried out under different situations, can be right or wrong.

For the described purpose, tasks and interpretation models were defined by means of the authoring tool PATH, which was already integrated in the framework. The following steps and situations were defined and implemented, based on signals from the process and user interaction, to support the diagnosis.

1) Normal operation: work sequence.

- Situation: operating in automatic mode.
- Steps:
  - Request permission for access to the facility to place a workpiece.
  - Start the process.
  - Identify color and height of the workpiece.
  - Sort workpiece.

2) Manual mode operation: linear table position setting.

- Situation: linear table position adjustment being carried out.
- Steps:
  - With table in input position, select manual mode.

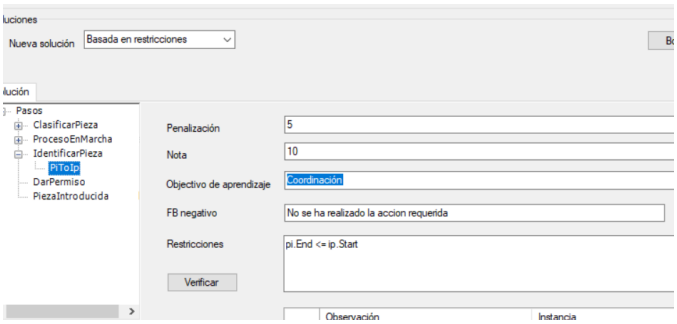


Fig. 6: Constraits by step.

- Move to predefined height testing position.
- Position adjustment (can be skipped).
- Validate position.
- Move to predefined colour testing position.
- Position adjustment (can be skipped).
- Validate position.
- Move to predefined output position.
- Position adjustment (can be skipped).
- Validate position.

The whole sequence could be repeated as many times as required. At the end of the sequence, it could be switched to automatic mode, if necessary. Note that if a return to automatic mode was made before the end of the procedure, it would be considered wrong.

The implementation, in the interpretation model, of the described steps and situations was carried out from the definition of restrictions associated with each step. These can be starting, finishing, general or action restrictions, among other types. Figure 6 shows the definition, in PATH, of a constraint associated with a step.

Every cycle of execution of the framework algorithm all rules were solved and the status of each step was determined, i.e. whether it started, finished, continued, etc. (see Figure 7).

VI. Results

An automated industrial process, both emulated by a DT and real, has been integrated into the framework, which has made it possible to diagnose steps and situations. In order to test the platform, the cell has operated alternating between automatic and manual modes. Randomly and periodically, long steps and incorrect sequences in terms of step order have been introduced. Table I summarizes types of situations tested, with several wrong and correct cases for each.

Figure 8a shows an error situation in the sequence of steps of the automatic mode situation, as well as an excess of duration of one of them. Figure 8b illustrates another scenario, that of a correct sequence of steps but with an excessive total duration of the manual mode operation.

As already indicated, this diagnosis was based on measuring the duration of the steps and checking the correct order of them. Note that these two variables were defined without any programming effort. The generation of observations was programmed automatically and through the OPC UA server.

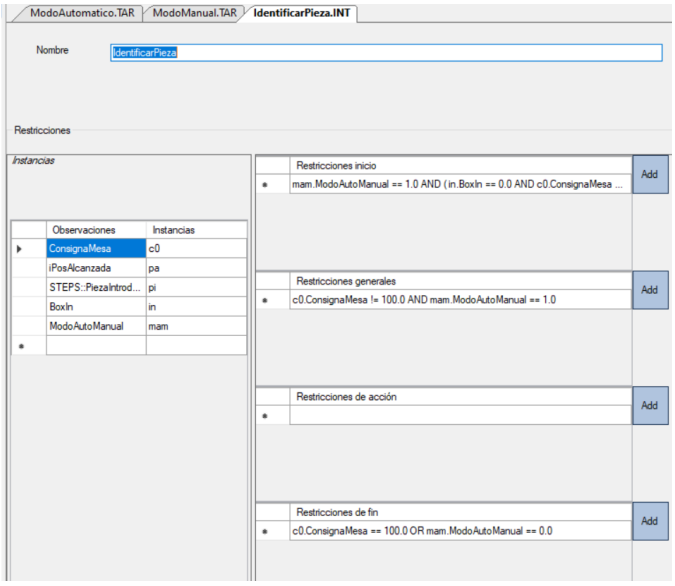


Fig. 7: Detection of whether a step is in progress.

TABLE I: Summary of results.

Tested situation	Result
Step order in sequence.	Diagnosed.
Step duration.	Diagnosed.
Sequence duration.	Diagnosed.

VII. Conclusions

A first approach to implement a framework for automatic context diagnosis of an industrial system has been presented. A use case based on a robotic cell has also been brought forward. The development of the work and its validation have been supported by a DT of the process, prior to integration with the real one.

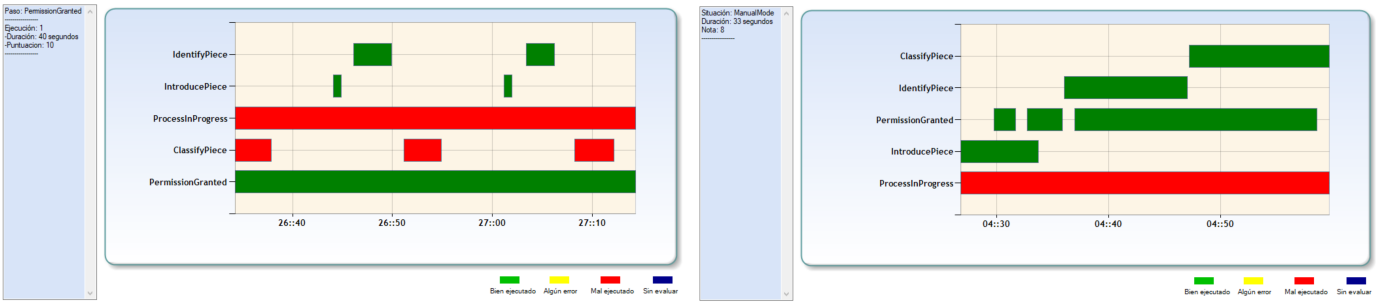
Emulation makes it possible to reproduce all scenarios of interest without affecting the operation and safety of real systems and users. It can be part of an exclusively virtual application, or support the configuration of one that will later work with a real process, as presented in the use case.

ADAPT can be used for diagnosing the correct execution of automatic and manual tasks, definition of procedures based on expert user interaction with the control system, and training of unskilled staff. Any real or emulated control device and any DT can be integrated, if they have OPC UA connectivity.

So far ADAPT has been tested mainly using process data comming from a PLC, however in future works process-operator interaction data from different HMI devices will also be included. This information will provide more complete view of the correctness of the task fulfillment, and applications such as operator training could be addressed.

References

[1] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret, “Context-aware cloud robotics for material handling in cognitive industrial internet of



(a) Error in sequence, and excessive duration of one step.

(b) Correct sequence but excessive total duration.

Fig. 8: Diagnosis reports.

things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2272–2281, 2018.

[2] A. N. Lee and J. L. Martinez Lastra, "Enhancement of industrial monitoring systems by utilizing context awareness," in *2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2013, pp. 277–284.

[3] A. Kuutti, A. Dvoryanchikova, A. Lobov, J. L. M. Lastra, and T. Vantera, "A device configuration management tool for context-aware system," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 10–15.

[4] A. Aguirre, A. Lozano-Rodero, L. M. Matey, M. Villamañe, and B. Ferrero, "A novel approach to diagnosing motor skills," *IEEE Transactions on Learning Technologies*, vol. 7, no. 4, pp. 304–318, Oct 2014.

[5] S. Ohlsson, "Constraint-based student modelling," *Journal of Interactive Learning Research*, vol. 3, no. 4, p. 429, 1992.

[6] B. Ferrero, "Detective: un entorno genérico e integrable para diagnóstico de actividades de aprendizaje," *Fundamento, diseño y evaluación [Tesis Doctoral]: Univ. of the Basque Country UPV/EHU*, 2004.

[7] D. SUN, *Synchronization and Control of Multiagent Systems*, 1st ed. CRC Press, Taylor & Francis Group, 11 2010.

[8] P. Leitão, P. Vrba, and T. Strasser, "Multi-agent systems as automation platform for intelligent energy systems," 11 2013, pp. 66–71.

[9] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with a fipa-compliant agent framework," *Software: Practice and Experience*, vol. 31, no. 2, pp. 103–128, 2001.

[10] A. Aguirre, A. Lozano-Rodero, M. Villamañe, B. Ferrero, and L. M. Matey, "Olympus: An intelligent interactive learning platform for procedural tasks." in *GRAPP/IVAPP*, 2012, pp. 543–550.

[11] I. McGregor, "The relationship between simulation and emulation," in *Proceedings of the Winter Simulation Conference*, vol. 2, Dec 2002, pp. 1683–1688 vol.2.

[12] P. Hoffmann, R. Schumann, T. Maksoud, and G. Premier, "Virtual commissioning of manufacturing systems - a review and new approaches for simplification," 06 2010, pp. 175–181.

[13] M. Schamp, S. Hoedt, A. Claeys, E. Aghezzaf, and J. Cottyn, "Impact of a virtual twin on commissioning time and quality." *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1047 – 1052, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896318315970

[14] M. Ayani, M. Ganebäck, and A. H.C. Ng, "Digital twin: Applying emulation for machine reconditioning," *Procedia CIRP*, vol. 72, pp. 243–248, 01 2018.

[15] S. T. Mortensen and O. Madsen, "A virtual commissioning learning platform," *Procedia Manufacturing*, vol. 23, pp. 93–98, 2018.

[16] N. Shahim and C. Møller, "Economic justification of virtual commissioning in automation industry," in *Proceedings of the 2016 Winter Simulation Conference*, ser. WSC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 2430–2441. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042094.3042396

[17] L. Damiani, M. Demartini, P. Giribone, M. Maggiani, R. Revetria, and F. Tonelli, "Simulation and digital twin based design of a production line: A case study," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 2, 2018.

[18] J. Rossmann, O. Stern, and R. Wischnewski, "Systematics with a corresponding software tool for the integrated virtual commissioning of production lines from planning over to simulation to operation," pp. 707–716, 01 2007.

[19] M. Oppelt and L. Urbas, "Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering," *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2564–2570, 2014.

[20] I. Azkarate Fernández, M. Ayani Eguía, and L. Eciolaza Echeverría, "Virtual commissioning of a robotic cell: an educational case study," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 820–825.

[21] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.