Triggering Strategies for Automatic and Online Service Reconfiguration Reconfiguration

Nelson Rodrigues[1,2,3], Paulo Leitão[1,2], Eugénio Oliveira[2,3]
[1] Polytechnic Institute of Bragança, Campus Sta Apolónia, 5300-253 Bragança, Portugal
[2] LIACC - Artificial Intelligence and Computer Science Laboratory, Rua Campo Alegre 1021, 4169-007 Porto, Portugal
[3] Faculty of Engineering - University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto
{nrodrigues, pleitao}@ipb.pt, eco@fe.up.pt

Abstract — **The failure to meet production requirements, e.g., due to condition dynamic environment condition changes, operational performance deviations or missing opportunities to adapt, leads to a decrease of competitiveness. Technological advances are then required to promote flexible and adaptive systems with the ability to reconfigure their offered services in a cost effective manner. In spite of the current research efforts, there is still a lack of automated tools to support the service reconfiguration capability at run-time, being the understanding of when and how to reconfigure crucial to support an efficient reconfiguration process. This paper focuses on the problematic of when to reconfigure a system, with the proposed service-based multi-agent system performing the dynamic service reconfiguration based on triggering strategies. This allows to discover reconfiguration opportunities to maintain the system stable and competitive, but also to explore and promote new system configurations when needed. The envisaged triggering strategies, implemented through the intelligence mechanisms embedded in the agents' behaviors, consist in the event, periodic and trend approaches following the reactive, predictive and preventive behaviors. The preliminary experimental results validate the feasibility of such triggering strategies for service reconfiguration leading to more efficient and agile systems.**

Keywords – Service reconfiguration, Multi-Agent Systems, Triggering Strategies

I. INTRODUCTION

The growing of global markets and interest of customer satisfaction, challenge the manufacturing companies to manage the complexity of delivering high-quality customized products while facing the continuous demanding requirements. As a consequence, it can be noticed an economic investment in several research projects in manufacturing fields to endorse reconfiguration, flexibility and evolvable systems, such as IDEAS, PRIME and GRACE [1-3,8].

Several reconfigurable paradigms were proposed during the last decade, introducing flexible and agile characteristics to react promptly to unexpected events, system failures, quality deviations, etc., avoiding the loss of financial revenue and trust from the customer's point of view [2]. As an example, the Reconfigurable Manufacturing System (RMS) [3] is a well know paradigm that provides the ability to repeatedly change and reorganise the components of a system in a cost-effective way.

Following this research line, many academic and industrial works report reconfiguration as a result of flexible and high responsiveness actions aiming to react quickly to uncertainties and changes. For example, the SOA (Service-Oriented Architecture) paradigm allows the service reconfiguration at real-time addressing the flexibility to produce more easily customized products, e.g., often used to control hardware devices as robots and grippers [4]. However, for small systems, a centralized approach, i.e., executed by one central decisional node, is traditionally used to perform the service reconfiguration. On other hand, the distributed and decentralized reconfiguration system implies benefits such as flexibility, modularity, reconfigurability, etc., which have been pointed out as appropriate for larger companies. Multi-Agent Systems (MAS) [5] decentralizes the control functions over distributed autonomous and cooperative control nodes to deliver modularity, autonomy and adaptability, for reacting promptly to changing conditions [6], being suitable to address distributed service reconfiguration problem.

The question addressed in this paper relies on the determination of when is the best moment to reconfigure, based on the analysis of the environment observation and available set of services. The service reconfiguration triggers should not be restricted to faulty events, but also consider, in a dynamic and run-time manner, the discovery of new reconfiguration opportunities and the exploration of new system configurations. Studies to understand when to reconfigure and how to change the system configuration are crucial to support an efficient reconfiguration process, and more important its impact in terms of monetary revenue. Having this in mind, the challenge is to identify the best strategies about when to reconfigure the system that can lead to a better production efficiency. For this reason, this work proposes the exploration of triggering strategies at run-time, implemented by a society of autonomous agents that are actively promoting service adjustments in the system's configurations.

The remain of this paper is organized as follows. Section II reviews the concept of service reconfigurability and presents some related work. Section III describes the proposed online service reconfiguration architecture, and Section IV details the triggering strategies used to implement the when to reconfigure module. Section V presents the preliminary experimental results by applying the triggering strategies. Finally, Section VI rounds up the paper with the conclusions and points out the future work.

II. RELATED WORK

Several academic and industrial research works report reconfiguration process as a result of flexible and high respon-

siveness actions aiming to quickly react to uncertainties and changes. In the Flexible Manufacturing Systems (FMS) context, some experiments considering self-adaptive infrastructures were made to dynamically identify changing contexts without any specific system's expert, aiming the improvement of the system overall performance along the production system life-cycle (see [7]). In this area, proactive adaptation is performed in a centralized manner, but the description of appropriate strategies to be implemented through the self-reconfiguration is still limited.

The growing interest in flexible and evolutionary systems that deal with uncertainty had promoted the creation of Evolutionary Production Systems or Evolvable Assembly System (EPS/EAS) paradigms as a mean to reach a flexible and evolvable approach in such dynamic environments. These paradigms provide a continuous reconfiguration environment and considers plug-ability as a critical issue. In this context, the IDEAS project [1] promotes the use of adaptable reconfigurable production systems, supported by MAS technology, capable of being on-line reconfigured without needing reprogramming efforts. The same principle of using MAS can be found in the GRACE project [8], where process and quality control were integrated envisioning the inclusion of self-adaptation and self-optimization mechanisms. The PRIME project proposes a self-organized architecture capable of reconfiguration capabilities for distributed manufacturing control systems. In a proactive way to support adaptation, the architecture makes possible to react to unexpected disturbances to keep the productivity and quality parameters [2].

Relating to the particular question of when to reconfigure, the control theory provides several triggering strategies to determine when it is possible to re-adapt the system. Both event- and self-triggered strategies are seen as the basic ones, although time-triggered strategies are also often mentioned [9]. These approaches are being applied, e.g., in wireless embedded control systems to reduce the frequency of transmission by intelligent decisions about when reducing the energy consumption.

These projects promote adaptive and flexible systems but the reconfiguration principles are mainly used to adjust some parameters to maintain the system's stability. Moreover, although these systems are considered able to be adapted and the flexibility to be preserved, the previous projects lack on the description of the methodologies that endorse the opportunities to adapt. Besides that, a proactive searching for different reconfiguration alternatives is also necessary, in order to find opportunities to improve the system performance.

However, and regardless the widespread literature on the field, most of the real reconfiguration solutions in manufacturing domain perform their adaptation procedures in reacting to failure events. A different perspective, and aligned with this work, considers predictive roles, giving attention to strategies and opportunities to adapt and reconfigure the system, without decreasing the performance due to unexpected disturbances. This identification is addressed in [10], where different types of changes in different moments might result in different costs that are worth to be evaluated for a proper reconfiguration policy. Besides the different reconfiguration criteria, e.g., adaptation time, the strategies to adapt in advance can be perceived as a predictive maintenance and thus much more effective. In this context, [11] presents a simulation-based approach that deals with the present gap of the analysis of the system reconfiguration in conjunction with maintenance operations. This study also describes when the reconfiguration process should be executed along with the proper time horizons for the execution of needed maintenance operations to increase the system throughput. The main drawback is that the adaptation of the system is made through a static rather than dynamic reconfiguration. A static reconfiguration involves stopping a running system, reconfigure it and then restarting the system. This methodology is not aligned with our needs, pointing to the need of automatic and online (dynamic) reconfiguration. Despite the promising perspective, there is a lack of implemented solutions that decide when to reconfigure the system in a proactive perspective through ongoing executions, without stopping the system. In fact, the literature does not offer expressive enough detail and results related with this kind of dynamic systems.

This lack of detail offers a research opportunity to further investigate techniques to decide when to reconfigure, and to perform re-configuration of service's features on the fly. A service-oriented paradigm supported by a MAS is proposed, to "intelligently" modify and make the services evolvable in accordance to the dynamic situation changes.

III. Automatic and Online Service Reconfiguration

Traditional control systems are, in general, centralized applications that represent a weakness to cope efficiently the requirements imposed to manufacturing systems, namely in terms of flexibility, responsiveness and re-configurability. MAS systems are appropriate to overcome these requirements and complexity, by providing an alternative way to design decentralized, autonomous and cooperative control systems to response to the occurrence of disturbances and the dynamic reconfiguration on the fly. Having this in mind, consider a service-based MAS ecosystem representing a composed system, e.g., multiple workstation based manufacturing system, which is defined as follows:

Definition 1 *(Composed System)* $C_{system}=\{A^t, S^t\}$ *at a specific instant t is composed of a set of agents* $A^t=\{a_1,\ldots,a_n\}$ *and provides a set of services* $S^t=\{s_1,\ldots,s_n\}$. *Each individual agent is able to provide a specific set of services at a specific instant.*

The proposed approach goes beyond the state of the art in terms of service reconfiguration, in the sense that it considers the ability to dynamically reconfigure the services in a distributed, decentralized and on-the-fly manner, identifying and exploring opportunities to reconfigure those services promoting the system's performance improvement. Such ability is adopted from the self-organizing principles that address simple service modifications in a distributed manner to dynamically evolve the system. By definition, the dynamic reconfiguration consists in readjusting the elements or settings (of a system) at run-time. Based on the several reconfiguration definitions, such as [12-14], the proposed definition for automatic and online service reconfiguration is as follows:

Definition 2: "*Online manufacturing reconfiguration is the ability to dynamically, proactively and repeatedly adapt the hardware and software components, to provide better services in a cost-effective way".*

The dynamic and online service reconfiguration mechanism proposed in this work is embedded in each distributed and autonomous agent, which includes several modules as illustrated in Figure 1.

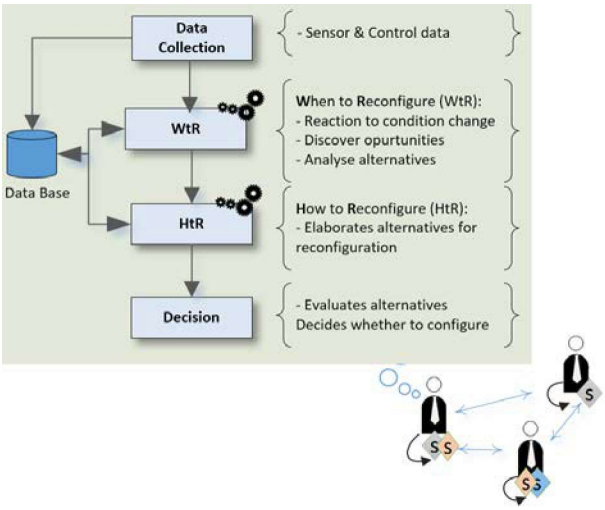

Figure 1. Service reconfiguration modules embedded in each agent (illustrated in the bottom figure) that belonging to the service-oriented MAS

The *Data collection* module is responsible for gathering and uploading the relevant data into a local database. The database can mimic the functionalities of the components by using data logging and manufacturing data acquisition systems, e.g., Supervisory Control and Data Acquisition (SCADA), to gather the manufacturing events, e.g., the insertion/removal/change of hardware or software components in the physical context. The *When to Reconfigure* module (WtR) is responsible for monitoring the agent performance and discover the needs and opportunities for reconfiguration, e.g., recognizing a decrease in the service utilization and consequently replacing this service by another more useful.

Several opportunities may arise during the discovery phase, e.g., plug-and-play of components, request of a new product type, service degradation and service failure, requiring a method to analyze and coordinate all these alternative triggers at run-time. Thus, according to this requirement, the control of stress and the learning from past experiments are crucial to impose a limit on the triggers proposals frequency, to avoid the chaos on the triggering proposal phase.

The *How to Reconfigure* module (HtR) is responsible for elaborating the set of alternatives for the reconfiguration, after being identified an opportunity to reconfigure (by the WtR module)**.** Taking into account the self-organising principles, as suggested by Kota et al. [15], two major capabilities were adopted as a reconfiguration baseline, namely:

- Management entity, that means the ability to structurally add/change/remove entities, i.e., agent and service.
- Modification of properties, that means the ability to change the entity's behaviour, e.g., to optimize a tool.

On the way to perform the reconfiguration, these two main pillars are relevant for our study, affecting the agents and the services. Table I, illustrates the possible types of reconfiguration considered in this work.

TABLE I. POSSIBLE TYPES OF RECONFIGURATION

Level	Description	Implementation Effort
Service	Change the catalogue of offered services by using pluggability characteristics, e.g., offering a new drilling operation	High
Agent	Change the MAS structure, e.g. plugging a new resource station	High
Service properties	Modification of the service properties, e.g., modifying the quality measurements of a specific tool	Low

On one hand, individual agents can make behavioral changes, e.g., optimizing their services to maintain or improve the performance at the service level. These improvements represent a low impact on the system if we compare them with the structural modifications. The structural changes may create a bigger impact on the system as a consequence of the introduction or removal of agents and services.

The reconfiguration *Decision* module is responsible for deciding if the reconfiguration should be implemented and which alternative is selected, taking into consideration the control of nervousness and the learning from the past reconfiguration experiments (note that controlling the nervousness prevents the system from becoming chaotic). To be able to take better decisions in the future, each agent should observe the effects and impact of the reconfiguration in its own performance improvement.

IV. DISCOVERING OPPORTUNITIES TO RECONFIGURE

This section details the description of the WtR module, which automates the data analysis according to different triggering strategies, aiming to automatic recognize opportunities for the reconfiguration on-the-fly.

The reconfiguration analysis relies on three triggering strategies, namely, e*vent, periodic* and *trend*. As illustrated in Figure 2, in any of these cases, the WtR module receives data from the *Data Collection* module or retrieves directly from the database.

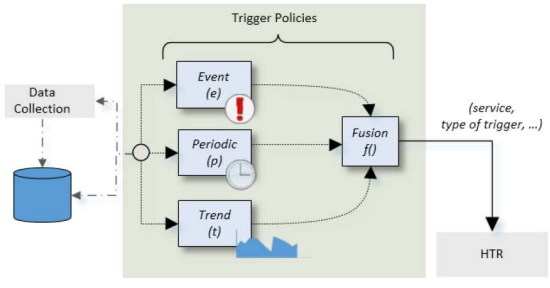

Figure 2. Internal architecture of the When to Reconfigure module

Each trigger policy applies a different strategy generating different triggers over the time. Some triggers pointing to re-

configuration may happen at the same time on different features, without any explicit interference between them. When this happens, the fusion module is responsible to merge the generated triggers and redirect the result to the HtR module. The behavior of each trigger policy can be dynamically adapted, by using appropriate learning algorithms, aiming to optimize the accuracy of the module, e.g., using the feedback about the reconfiguration suggestions (collected by the Data Collection module) to adjust the last threshold parameters' values.

Next subsections will detail the three triggering strategies.

A. Event Triggering

This strategy is activated by using an event-driven approach that detects events related to the system condition changes, e.g., a resource failure, the addition of a new resource or the removal of an existing resource.

This type of strategy permits a good reaction to face critical events, in analogy with the corrective maintenance benefits. Basically, since the strategy trigger is a direct one, the time spent on the decisions of the strategy trigger is irrelevant. For example, if a resource failure occurs, a reactive event is triggered and sent to the HtR module. An important point is the ability of such strategy to deal with unexpected events that occur in the system, which turns out to be an important feature in dynamic and unpredictable environments.

B. Periodic Triggering

This strategy is performed periodically based on a time interval, Δ, which may be dynamically adapted. In opposition to the event triggering, this strategy occurs on a scheduled basis, being very similar to the preventive maintenance. An example may be to change the car's oil, where the frequency of maintenance can be calculated according to statistical information about the components and the running process to schedule e.g., using the mean-time-between-failures (MTBF) parameter.

In general, this strategy enables periodically the execution of a preventive service reconfiguration, reducing the probability of service failure or performance degradation. The triggering time interval should be dynamically adjusted to better fit the system dynamics, i.e. increasing or decreasing the Δ value, taking into the consideration the application of proper machine learning algorithms. In this context, Q-learning [16] is a suitable approach to address this challenge, since it provides a positive/negative reinforcement feedback that handles the system's dynamics, allowing to converge to an optimal value of Δ by mapping the values of each agent in a table state-action, as follows:

1. The state represents the agent's current knowledge situation. Specifically in our case, the State is formally defined by, State=<S^t, β> where S^t is composed by the set of services $S^t=\{s_1,\ldots,s_n\}$, at the instant *t*, β=<ξ, ρ> corresponds to the description of the production batch, ξ represents the number of orders, and ρ the product type.
2. Agents will take some pre-defined actions regarding the dynamic update of the Δ value, which will change according to the following set of actions: i) increase the time interval (↑Δ), ii) maintain the time interval (=Δ) or iii) decrease the time interval (↓Δ), formally, Action=⟨↑Δ, =Δ, ↓Δ ⟩.
3. The reward (R) is calculated by considering the action taken in a specific state and the reconfiguration triggering feedback. If the reconfiguration activity had led to a better configuration structure, the Q-value of state-action pair chosen will be increased. Otherwise, it will suffer a penalty for the chosen Δ value, given by equation (1), where *n* is the reward constant:

$$\mathrm{R(State, Action)} = \begin{cases} +\mathrm{n}, \text{ if reconfigure} \\ -\left(\frac{\mathrm{n}}{2} * \mathrm{penalty}\right), \text{ if not reconfigure} \end{cases} \quad (1)$$

In general, after a few number of interactions, the agent is able to select an optimal triggering frequency, in a given State. This strategy leads to a decrease of the event triggering strategy since the agent is executing the service reconfiguration more efficiently in a preventive way. Despite the advantages of this strategy, some opportunities to reconfigure due to a degradation of the service performance will be lost, since this strategy is driven by means of time periods and not by the constant monitoring of the services activities themselves. However, such challenge is covered by the trend triggering strategy.

C. Trend Triggering

This strategy aims to recognize a tendency or pattern in the degradation of a service performance, anticipating actions to improve its performance or to reconfigure by another more useful.

Having this in mind, the first hypothesis is related to use data mining algorithms to explore a large set of data in order to extract valuable information and build models able to automatically identify patterns and to predict the future state of the system. For this purpose, the agents, after creating accurate models, can classify/predict the upcoming events, operations, status of the machines, etc., to detect unusual values from the normal pattern, which constitutes opportunities to reconfigure. In this way, several algorithms can identify these opportunities, namely anomaly detection, cluster analysis-based and structural break.

The anomaly detection and cluster analysis-based methods may be useful to discover anomalies in patterns [17], e.g., the detection of the anomaly illustrated in Figure 3, as "Anomaly (outlier)". The anomaly detection method can be used for example with i) dataset already labeled as "normal" and "abnormal", where a supervised algorithm determines to what class a given instances belongs, and ii) unlabeled dataset, where several unsupervised algorithms identify the instances that are considerably different from the others, for example classify them as outliers.

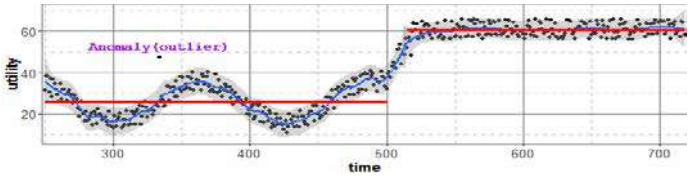

Figure 3. Example of a trend identifying an anomaly and the reaction to a product changeover detected by the structural break analysis

In general, the machines produce the same type of products during a certain time, and when a product type changeover happens, the effects could be observed by trends in the monitoring service data. This type of changeover results in trend deviations which may be understood as an abnormal behavior, thus anomaly detection algorithms are not useful here. The analyzed trends cannot be ignored since they lead to a crucial side-effect is the production changeovers. A simple trend analysis performed by the module still rely on a good approach, however, particularly in this case another method may be used, namely the structural break or structural changes analysis [18]. This is illustrated in Figure 3, where each horizontal red line corresponds to different products. This method activates the reconfiguration trigger to proceed with the necessary adjustments for the identified upcoming batch of products. This mechanism allows to reduce the ramp-up of new production batches.

The second hypothesis to handle the trend triggering is concerned to activate the trigger for reconfiguration of services, exploring the execution of new services (i.e. change the catalogue of services that are offered by the agents). For this purpose, during the execution time, p, the agent is monitoring the number of services being requested, the number of services being executed ($\#Executed_p$) and the number of services that are rejected because they are not installed ($\#Rejected_p$). In this sense, the triggering activation condition for service reconfiguration could be based on the equation (2):

$$\delta_p = f\left(\frac{\#Rejectedp}{\#Rejectedp + \#Executedp}\right) \quad (2)$$

where δ_p represents the rate of rejected services during the periods p. This moving average considering the last p period, allows to focus the relevance of this calculation for the last events. In general, the trigger is activated when the condition $\delta_p \geq \theta$ is satisfied, where θ represents the threshold value. The determination of the θ value should be performed dynamically and at run-time, and is dependent on the application scenario and the system nervousness. Section V shows some experiments in detecting the optimal value for this parameter.

V. RESULTS

The proposed approach for service reconfiguration was validated based on Bench4Start benchmarking [19] as the test case to explore the reconfiguration strategies in production lines. This framework considers the AIP-PRIMECA flexible manufacturing system that comprises a set of 6 workstations (WS), interconnected through conveyors, being each WS offering a limited set of operations (i.e. services) (see [19] for more details). The system is able to produce a set of sub-products, namely the letters B, E, L, A, I, P and T, which combining can produce the final products BELT and AIP.

The proposed service-oriented MAS was implemented using the JADE framework [20], and the triggering strategies described in section IV were validated by considering the testing scenario represented in Table II:

TABLE II. TESTING SCENARIO CONFIGURATION

Parameters	Values
number of products	7
product type	B - E - L - T
max number of services offered per WS	3
WS with reconfiguration capabilities	WS4
Threshold variance (θ)	[0.2, 0.3, 0.4, 0.5, 0.6, 1]

The service-oriented MAS is initialized by comprising 6 agents to represent the WS disposed in the production system, and 28 agents to represent each product instance (i.e. 7xBELT=7x4=28). After the collaborative services' allocation performed among the agents aiming to reach the optimal schedule for each production order, the system starts to run normally towards the execution of the production batch (i.e. 7 products). The idea is to let the agents search for strategic opportunities to reconfigure their services in an automatic mode during their life-cycle execution, particularly the workstation 4 (WS4). In this work, the second hypothesis for the trend triggering was considered, being applied the equation (2) to generate the proper triggers if the condition $\delta_p \geq \theta$ is satisfied .

Figure 4 illustrates the results of the experiments in terms of the Cmax (representing the total amount of time necessary to complete production batch) of each simulation considering a different threshold value. Above each Cmax value depicted in Figure 4 is represented the number of reconfigurations performed during the experiment for that threshold value (e.g., for θ=0.2, the system was reconfigured 4 times). Note that the analysis of the different reconfiguration impact in the Cmax value should be made comparing to the Cmax value obtained for θ=1, which represents the simulation without performing any reconfiguration.

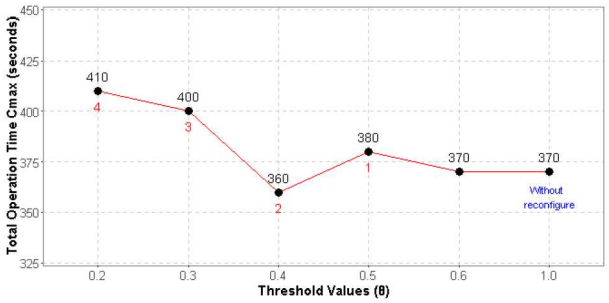

Figure 4. Results of applying different threshold values for the trend triggering using a production batch size of 7 products

The objective of these experiments is to decide the θ value that minimizes the Cmax time through the implementation of service reconfigurations. The analysis of Figure 4 clearly shows that each tested θ value leads to different Cmax values, meaning that the selection of the θ value will have different impact in the system performance. In fact, lower values of θ, meaning that the system is very nervous to constantly try to reconfigure their services, leads to the increase of the Cmax, reaching higher values than without applying any reconfiguration triggering strategy. This situation should be avoided in order to maintain the system stable and not in a chaotic state. On the other hand, higher values of θ, meaning that the system is very calm and usually never reaches the trigger for the service reconfiguration, lead to better results than those achieved for lower θ values but slightly worse than without applying any reconfiguration.

The best threshold value for this case study is for θ=0.4, where 2 service reconfigurations were performed, showing clearly that the system should find an intermediate point that balances between calmness and nervousness. Moreover, it is possible to conclude that reconfiguring several times may not always be beneficial in terms of Cmax, since the required time to implement the service reconfiguration (not only the computational time but essentially the time to physically change the service offered by the agent) will strongly affect the Cmax value. The challenge is to dynamically adjust the threshold value according to the application domain, the size of the production batch and the impact of the reconfiguration time in the processing times.

Additionally, Figure 5 illustrates the service utilization ratio for WS4 along the time for the different threshold values.

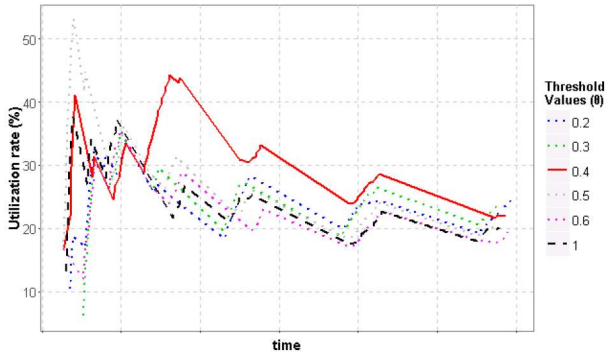

Figure 5. Resource utilization rate of WS 4 for the different threshold values

The analysis of these results shows a considerably higher service utilization rate for θ=0.4 (which is the θ value that leads to the minimum Cmax value, as observed in Figure 4) when compared with the other θ values. This means that the service reconfiguration considering a proper triggering mechanism represents a wiser and maximized utilization of the services.

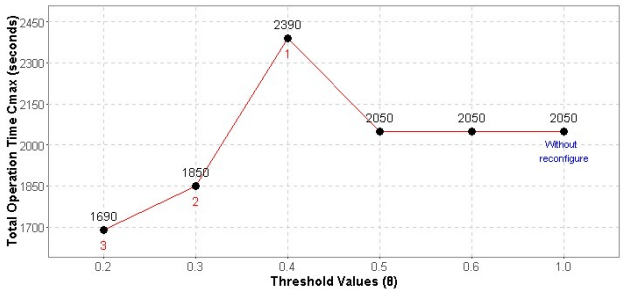

Figure 6. Results of applying different threshold values for the trend triggering for a higher production batch size of 14 products

As illustrated in Figure 6, the service reconfiguration is strongly dependent on the dimension of the production batch, having higher impact, namely in terms Cmax improvement, for bigger production batch sizes.

VI. CONCLUSIONS

This work proposes an approach for the dynamic, efficient and on-the-fly reconfiguration of services. The proposed approach actively explores and promotes different triggering strategies, embedded in smart agents, that does not only identify opportunities to change, but also to assist engineers and managers in exploring and deciding about different alternative configuration scenarios, to cope with potential disturbances or production changeover challenges or recommend service modifications.

Three different triggering strategies were designed, namely event, periodic and trend, and posteriorly tested using a flexible manufacturing system case study. The experimental results showed that the proper use of the dynamic service reconfiguration mechanism is strongly dependent of the threshold value, which should be dynamic and on-the-fly adjusted by using learning mechanisms.

Future work will be focused on testing the other designed triggering mechanisms, namely the periodic triggering, as well as more elaborated methods for the trend triggering. Regarding the Q-learning-based approach in periodic triggering, it is necessary to explore techniques to reduce the potential massive state space created. Afterwards, a more practical evaluation involving real industrial scenarios will be needed to validate the proposed dynamic and on-the-fly service reconfiguration solution.

REFERENCES

[1] M. Onori, N. Lohse, J. Barata, and C. Hanisch, "The IDEAS project: plug & produce at shop-floor level," Assem. Autom., vol. 32, pp. 124–134, 2012.

[2] "Plug and produce intelligent multiagent environment based on standard technology." Accessed on: 15 March 2016. Available from: http://www.prime-eu.com/.

[3] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable Manufacturing Systems," CIRP Ann. -Manuf. Technol., vol. 48, no. 2, pp. 527–540, 1999.

[4] J. M. Mendes, P. Leitão, A. W. Colombo, F. Restivo, "Service-oriented control architecture for reconfigurable production systems," 6th IEEE International Conference on Industrial Informatics, pp. 744–749,2008.

[5] J. Ferber, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison-Wesley Longman Publishing Co., Inc., 1999.

[6] M. Wooldridge, Introduction to Multiagent Systems, vol. 30. New York, NY, USA: John Wiley and; Sons, Inc., 2002.

[7] G. di Orio, G. Candido, J. Barata, S. Scholze, O. Kotte, and D. Stokic, "Self-Learning Production Systems (SLPS) - Optimization of manufacturing process parameters for the shoe industry," 11th IEEE International Conference on Industrial Informatics, pp. 386–391, 2013.

[8] P. Leitão and N. Rodrigues, "Multi-Agent System for On-demand Production Integrating Production and Quality Control," Holonic Multi-Agent Syst. Manuf., vol. 6867, pp. 84–93, 2011.

[9] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," IEEE Trans. Automat. Contr., vol. 52, no. 9, pp. 1680–1685, 2007.

[10] L. S. Belisario and H. Pierreval, "A conceptual framework for analyzing adaptable and reconfigurable manufacturing systems," Proceedings of 2013 International Conference on Industrial Engineering and Systems Management. pp. 1–7, 2013.

[11] J. Zhou, D. Djurdjanovic, J. Ivy, and J. Ni, "Integrated reconfiguration and age-based preventive maintenance decision making," IIE Trans., vol. 39, no. 12, pp. 1085–1102, Oct. 2007.

[12] R. M. Setchi and N. Lagos, "Reconfigurability and Reconfigurable Manufacturing Systems - State-of-the-art Review," 2nd IEEE Int. Conf. on Ind. Informatics, pp. 529–535, 2004.

[13] D. Bhatia, "Reconfigurable computing," Proc. Tenth Int. Conf. VLSI Des., pp. 356–359, 1997.

[14] A. M. Farid, “Measures of reconfigurability and its key characteristics in intelligent manufacturing systems,” J. Intell. Manuf., pp. 1–17, 2014.

[15] R. Kota, N. Gibbins, and N. R. Jennings, “Self-Organising Agent Organisations,” 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’09), 2009, pp. 797–804.

[16] C. J. C. H. Watkins and P. Dayan, “Q-learning,” Mach. Learn., vol. 8, no. 3–4, pp. 279–292, 1992.

[17] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Comput. Surv., vol. 41, no. September, pp. 1–58, 2009.

[18] D. N. Gujarati and D. C. Porter, Basic Econometrics. McGraw-Hill Irwin, 2009.

[19] D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, and J. Barbosa, “Benchmarking flexible job-shop scheduling and control systems,” Control Eng. Pract., vol. 21, no. 9, pp. 1204–1225, 2013.

[20] F. L. Bellifemine, G. Caire, and D. Greenwood, Developing Multi-Agent Systems with JADE, John Wiley & Sons, 2007.