Application of Multi Agent Systems for Leak Testing*

Hamood Ur Rehman[1,2], Jack C. Chaplin[1], Leszek Zarzycki[2], Mark Jones[2] and Svetan Ratchev [1]

***Abstract*— The manufacturing of customised products is a driver in the trend of incorporating intelligence in the system. This intelligence is required to enable the system to self-configure processes to meet the requirements of unique products. This work deals with the quality testing phase in a production system. A method of controlling testing functionality for leak test process is presented utilising a multi-agent approach. A means of controlling agent execution by expressions is discussed. Two types of expressions; operational and conditional expressions are conceptualised. An experimental use case is demonstrated to validate the approach.**

***Index Terms*— leak test, multi-agent, expression driven, intelligent manufacturing and industrial control.**

I. INTRODUCTION

Product customisation and fast-paced changes in product demand has forced manufacturers to reconsider their traditional manufacturing models. Data driven transformation in manufacturing is driven by new technologies such as machine learning (ML), reinforcement learning (RL) band cloud computing [1]. New paradigms such as agent based modelling, and service oriented architecture (SOA) are being adopted. The shift is mainly towards replacing centralised control to a more decentralised and distributed one to accommodate changing product and process requirements .

Reconfigurable Manufacturing Systems (RMS) [2],Bionic Manufacturing Systems [3],Holonic and Evolvable Production Systems [4] present the case for transition to a distributed environment for production operations. These developments contribute to dynamic adaptability and reconfigurability. High product customisation is the driver with intelligence embedded in the production system being the main enabler.

Data driven emerging technologies like cloud computing and agent systems supporting this paradigm shift ensure the required infrastructure for system interoperability, process and device integration, and associated data management is in place. This is necessary to transition the traditional manufacturing setups through a digital transformation to a fully digital manufacturing environment.

This digital transformation comes with its own challenges, one of which is lack of practical applications [5]. Industrial adaptation of these technologies will be accelerated if there exists multiple business and use-cases that demonstrate their feasibility and achieve their purpose. This insufficiency of methodologies and their underlying practical applications acts as a barrier to real-world realisation leaving most of the work in theoretical abstraction.

This article discusses the implementation of multi-agent technology in a real-world production system for a product testing application. For many high-value or highly regulated sectors it is necessary to check each and every part before sending it to the following work station to ensure 'no-faults-forwards'. High product customisation places additional requirements on testing as it needs to be adapted to the unique features of the part being tested. This is typically, done manually.

In this methodology, the multi-agent technology is used to drive testing based on expressions. Expressions are representative statements that are used to drive the functionality of the production system by influencing the agent system. This implementation case is generalised to be applicable to wide variety of manufacturing environments. The agents control and direct the functionality of the testing production system basing their execution on some expressions obtained through the user or by a machine learning pipeline. This storage component service integration provides control "as a service" to the additional scalablity characterstics inherent in multi-agent systems.

This paper is organised as follows: Section 2 gives the relevant background of the technology and, approaches for this methodology. Section 3 deals with the architecture developed for testing processes by integrating agents with process. Section 4 demonstrates the feasibility of the architecture by practical deployment to an industrial testing system and finally, section 5 concludes the paper while giving future research directions.

II. BACKGROUND

Intelligent production systems can reduce setup times, improve production planning, and decrease the frequency and severity of breakdowns, by gathering data and implementing responsive control strategies [6]. This is improved when these production systems have the capability of tracking and tracing objects, product control integrated with warehouse management systems, production monitoring and fast fault detection [6]. These intelligent production systems have led to the conceptualisation and development of distributed production intelligence and control leveraging their advantage of modularity, flexibility and reconfiguration. Their use has

*This work is carried out under DiManD Innovative Training Network (ITN) project funded by the European Union through the Marie Sktodowska-Curie Innovative Training Networks (H2020-MSCA-ITN-2018) under grant agreement number no. 814078

[1]Institute for Advanced Manufacturing, University of Nottingham, Nottingham NG8 1BB, UK, {Hamood.Rehman, Jack.Chaplin , Svetan.Ratchev}@nottingham.ac.uk

[2]TQC Ltd., Hooton St, Carlton Rd, Nottingham NG3 2NJ, UK {leszek.zarzycki, mark.jones}@tqc.co.uk

seen recent growth especially in applications where rapid response to unexpected changes is required [7], [8].

Architectures such as ADACOR (Adaptive Holonic Control Architecture), PROSA (Product Re-source Order Staff Architecture), HCBA (Holonic Control Based Architecture), ORCA(Orchestration and Reconfiguration Control Architecture) and POLLUX are developed that use the improved flexibility, connectivity, adaptability and behaviour for distributed control applications [4]. Intelligent production systems can be best realised when coupled with dynamic operations, dynamic prioritisation and effective task migration [9]. Along with this, intelligent production control requires a consistent adaptive manufacturing system monitoring ensuring quality requirements are met and faults detected diagnosed [10].

Intelligent production systems are assumed to perform their tasks completely or partially independently without external assistance [11]. A production system is a set of processes grouped together to achieve an outcome taking an input and passing the output through each processing stage [12]. These process stages may be a single production operation step producing a finished product or a group of associated processes [6]. In this sense, intelligence in production systems needs not only to be distributed but also scalable to accommodate part variation (due to feature requirements), and process changes at each production step [9].

The main driver of intelligence in manufacturing is efficient production execution and productivity improvement [13]. In this respect it is linked to quality control and quality assurance [14]. In many sectors, quality testing is too labour intensive, too complex,has a low degree of automation and is mostly experience based [15]. This results in lower efficiency, a higher level of biases, and lower overall quality.

An intelligent testing production system has to support "four modernisations" that are interconnection, digitisation, information generation and intellectualisation [15]. This is important to ensure that the intelligent testing system understands the tasks, resources and other information in the platform to make correct inferences. This ability must be supported by a management system [16] that offers monitoring and traceability of all production stages. This is complimented with use of computing capability for on-board or cloud based data processing for real-time analysis [17].

Cloud based technologies enable virtualisation of resources and capabilities in a manufacturing environment [1]. Previous researches have focused on cloud concept in detail while stressing the need for converting resources to virtual by service oriented architecture and interoperablity adaptation [1]. Usually the cloud technologies follow a on-demand use of resources in a pay-as-you-go-model or deployed-as-you-need model [18]. Cloud services mainly lie into Platform-as-a-service(PaaS),Infrastructure as a service(IaaS), Software as a service(SaaS). Distributed intelligent production systems use these services for control and execution.

In an intelligent testing production system it is necessary that the system is able to perceive the system state and execute control based on that perception [19] so this necessitates automatic collection and transmission of data , data processing and decision making. Furthermore, an intelligent testing system needs to consider the level of automation, standardisation, high efficiency and accuracy required for 'no-faults-forwards' production.

A technology to realise intelligence in production systems is the use of a multi-agent approach [20]. Multi-Agent Systems (MAS) divides the production system and assigns autonomous and independent software agents where each agent achieves its goals by interacting with other agents [20]. MAS supports applications in scalable environments along with areas where components with conflicting objectives need to interact [21]. MAS breaks down these component functionalities of the production system into much simpler entities.

Agents have been applied in production systems from supply chains to manufacturing operation scheduling [22], [23], [24], [25]. Usually coupled with different technologies like Service Oriented Architecture (SOA) and semantic web ontologies, they present an effective management approach used to realise concepts such as Holonic Manufacturing Cells [26]. Cloud and edge integration, and zero-defect control paradigm have been drivers for intelligent production systems [27].Utilising the technologies for a more effective control is discussed in this research.

III. AGENT ARCHITECTURE FOR EXPRESSION BASED LEAK TESTING

The architecture for an agent-based deployment for a testing production system is comprised of multiple components, namely the multi-agent system, the testing production system and data storage component (local database or cloud-based service). The expression that drives the agent skill execution is hosted in the storage component. The expression is sent to the agent system at the start of each test operation. A brief of overview of the components and their interaction is given as follows:

A. Multi Agent System (MAS) for Skill Execution

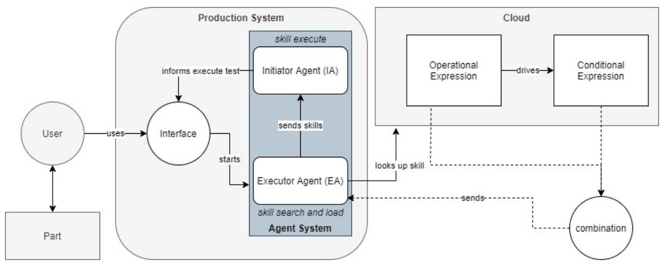


Fig. 1. MAS architecture for testing production system integrating expressions

A testing production system needs to execute skills in a certain order and with specific conditions to account for part, feature and conditional variations (changing priorities, testing requirements etc.). Agents can be used to automate control of this execution. The skill order can be defined in the operational expression, and conditional expression can be used to control behaviour of that skill.

Agents are given different roles to enable the system to function. For testing applications the Executor Agent (EA) can look-up Skills ‘S’ offered by an Initiator Agent (IA) based on an operational expression (see Table I). The Initiator Agent (IA) loads related conditional expression from storage component depending on information about the product gathered from EA (EA stores part related information). The IA executes the skills as per the expressions and confirms of the execution with EA. The integration with storage components ensures scalability, modularity and extensibility. For the purpose of testing simplicity and greater control IA is considered to be a group of agents where each each agent represents a skill it can execute (one skill per agent).

TABLE I
AGENT DESCRIPTION FOR TESTING SYSTEM

Agent	Description
Initiator Agent (IA)	Agents containing Skills that can be offered. IA is a group of agents offering one skill per agent.IA loads required conditional expressions for the skill execution
Executor Agent (EA)	Inquires skills from IA and executes the Skills based on operational expression.

The expressions, stored in a cloud-based or any other storage component oriented service, can be updated regularly to account for different requirements. This update can be carried out by the user or by a machine learning pipeline. As agents are linked to these expressions so they are also updated by the virtue of expressions giving an aspect of dynamicity during testing operation. A more descriptive respresentation of the architecture is shown in figure 1.

The architecture utilises agent technology to control execution based on expressions. This approach can be adaptable to accommodate different scenarios such as for part variation, feature variation and conditional variation. Figure 2 illustrates roles of agents in case of different scenarios and the manner in which they co-ordinate for expression development during execution operation.

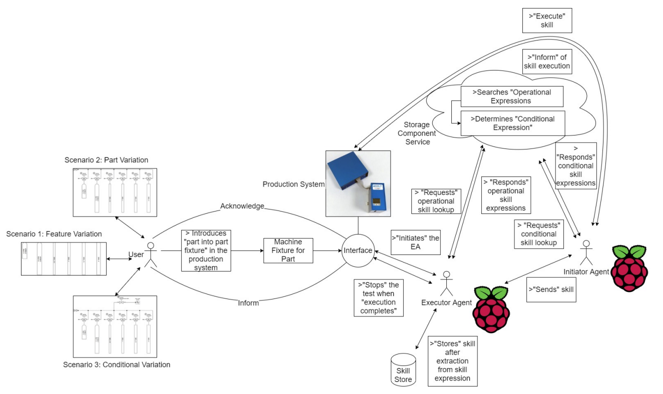

>"Execute" skill
>"Inform" of skill execution
>Searches "Operational Expressions
>Determines "Conditional Expression"
Storage Component Service
Scenario 2: Part Variation
Production System
> "Requests" operational skill lookup
> "Responds" operational skill expressions
> "Responds" conditional skill expressions
> "Requests" conditional skill lookup
Acknowledge
Scenario 1: Feature Variation
User
> Introduces "part into part fixture" in the production system
Machine Fixture for Part
Interface
>"Initiates" the EA
Initiator Agent
>"Sends" skill
>"Stops" the test when "execution completes"
Executor Agent
Inform
Scenario 3: Conditional Variation
Skill Store
>"Stores" skill after extraction from skill expression

Fig. 2. Role of Agents in Expression Development with changing scenarios; feature variation, part variation and conditional variation

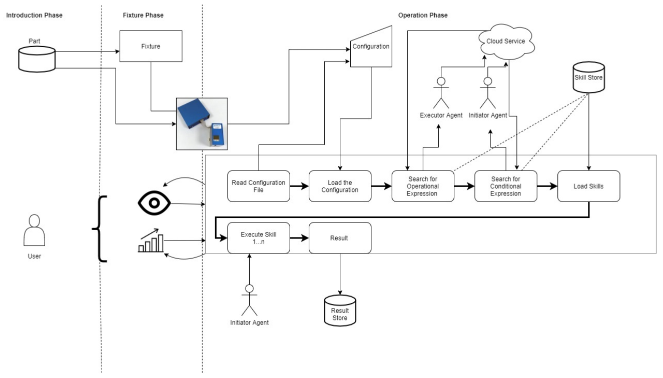


Fig. 3. Integration of testing process with expression development through storage component (cloud services)

These agents interact with the test process at different stages of execution, usually involving a different requirement to be fulfilled each time. These requirements as they are fulfilled can be viewed by the operator either by witnessing the change in configuration values or the result of test. Figure 3 presents this approach in a much descriptive form along with figure 4 detailing on the behavioural aspect of each actor during operation.

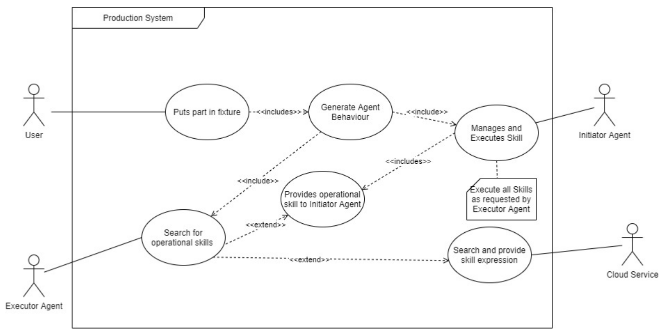

Production System
User
Puts part in fixture
<<includes>>
Generate Agent Behaviour
<<include>>
Manages and Executes Skill
Initiator Agent
<<include>>
<<includes>>
Provides operational skill to Initiator Agent
Execute all Skills as requested by Executor Agent
<<extend>>
Search for operational skills
Search and provide skill expression
<<extend>>
Executor Agent
Cloud Service

Fig. 4. Behaviour representation of actors in testing system

B. *Expression Concept for Production System*

Expressions drive the functionality of the production system by influencing the agent system. Expressions we conceptualise are of two types; “conditional” and “operational”. Conditional expressions govern the behaviour of the agent when executing a skill while on the other hand operational expressions are concerned with the behaviour of the production system itself. At its core the operational expressions deal with ‘what’ in a system and conditional expression define the ‘how’ aspect.

Operational expressions, involving the execution or performance of the testing operation, are accompanied by expressions formulated in a logic-based language that can be understood by agent systems. These expressions are responsible for the manner in which the testing production system carries out its execution for any part.

Let ‘T’ be an operational expression that constitutes of a set of skills to be executed. These skills are executed sequentially, parallel or in interleaved manner. The operational expression ‘T’ can therefore be considered as a set of skills and their associated conditions;

$$T = \{C_1S_1, C_2S_2, C_3S_3, ..., C_nS_n\} \quad (1)$$

The ‘T’ operational expressions is comprised of Skill ‘S’ under condition ‘C’ executed in a mentioned manner. The behaviours in which these skills under conditions are executed are represented in expressions given as follows;

$$T_{sequential} = C_iS_i.C_jS_j \quad (2)$$

$$T_{choice} = C_iS_i \bigoplus C_jS_j \quad (3)$$

$$T_{parallel} = C_iS_i \mid\mid C_jS_j \quad (4)$$

$$T_{interleaved} = C_iS_i \mid C_jS_j \quad (5)$$

These expressions can be formulated either to demonstrate sequential,choice, parallel or interleaved behaviour individually or in a combination. The interleaved behaviour represents a case of skill synchronisation where one skill waits till other skill is ready to be executed.

Operational and conditional expressions are responsible for agent skill execution for any testing operation carried out by the production system. An operational expression is accompanied by multiple conditional expressions for each skill. These coupled expressions can be loaded into the agent responsible for execution, as part is identified, for test to be carried out. These expressions can be housed in storage component on a scalable, extensible database service that is queried by the agent system to load best possible expressions for the identified parts. A depiction of the expression driven mechanism is shown in figure 5.

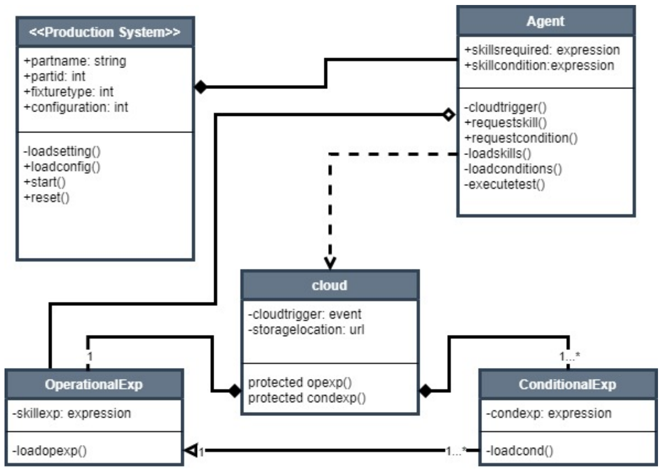


Fig. 5. Class Diagram for driving expressions from storage component (cloud service) for testing

IV. AGENT DEPLOYMENT FOR A LEAK TESTING USE CASE

To demonstrate the expression-based methodology, it is demonstrated with a use-case of an industrial dry-air leak testing production system (Micro Application Leak Test System). This is used to demonstrate the applicability of the approach along with providing the direction for deploying the methodology for other testing production systems. The industrial testing production system is connected to a Raspberry Pi. The Raspberry Pi acts as a gateway device to establish connection with cloud-based services. JADE (Java Agent Development Framework) is used for agent development and deployment.

Leak Testing is the process of determining leakage flow rate from a part under test. The leak testing production system constitutes of multiple processes. The part under test process is filled (over-pressure) with a testing medium or subjected to complete vacuum and allowed to stabilise under pressure. The pressure change is measured under leakage flow. This change in pressure or for simplicity proportional value of leakage flow rate is used as indicator to pass or fail on part. There are several variations for leak testing such as differential pressure measurement pressure decay, dosing leak test, chamber leak test, blockage testing, and coarse or gross leak test [28].

Testing is carried out at normal room conditions on a clean, dry, room temperature stabilised product. These conditions are maintained devoid of excess moisture due to temperature-sensitive nature of the dry-air leak testing consideration. Parameters for testing, calibration for attached instrumentation along with connection settings have been configured through the interface. This configuration has been set up to give a pass/fail criteria based on test pressure and differential pressure readings. JADE framework has been used to set up agents that control individual skills.

The cloud service, acting as a storage component, used for testing is by Google Cloud Platform (GCP). Cloud functions are event trigger functions that listen to agents and send back respective expressions when requested. The expressions are stored in data warehouse, that can be extended, to include more expressions.

Executor Agent (EA1) looks for the skills required for testing based from an “operational” expression. The “operational” expression lists down the skill required by part being tested as well as the sequence of execution for that part. This expression is stored in cloud based and loaded by EA1 based on part identification.

$$\begin{aligned} T_{EA1} = C_\phi\, basic.C_\phi\, cond.C_{ST1}\, stab.C_{FA1}\, fill. \\ C_\phi\, param.C_\phi\, iter.T_{EA1} \end{aligned} \quad (6)$$

The EA1 searches for the agents responsible for these skills. These agents then look for the “conditional” expressions that define the behaviour on how these skills may be executed. C_ϕ is a null condition that means no condition expression is present for executing the skill.So the expression may be simplified;

$$T_{EA1} = basic.cond.C_{ST1}\, stab.C_{FA1}\, fill.param.iter.T_{EA1} \quad (7)$$

TI1 agent is responsible for “basic” skill that starts the connection with the test system, CA1 executes “cond” skill that uses condition to change test parameters, ST1 changes the stabilisation time through “stab” skill, FA1 and PA1 induces change to fill time and any parameter (taken as argument) through “fill” and “param” skill respectively. IT1 is the agent responsible for “iter” skill that gives number of test iterations to be performed and executes the tests as per those iterations. Giving each agent responsibility of individual skill gives capability to define test for each part driven by “operational” expression. Here, TI1, CA1, ST1, PA1, FA1, and IT1 are a group of agents previously established as IA.

These agents once they have received request to execute a skill look for a conditional expression from cloud service. This expression is loaded and the skill behaviour modified as per expression. For the purpose of this test the fill time and stabilisation time was varied by each increment. The conditional expressions are;

$$C_{FA1} = i * fill \tag{8}$$

$$where \quad i = 1...i \; increments$$

$$C_{ST1} = stabilisation\; time + i * t \tag{9}$$

$$where \quad i = 1...i \; increments \quad and \quad t = constant$$

These expressions are loaded from cloud service and execution of the test carried out. The sequence diagram along with respective production system execution interface is given as figure 6 and figure 7. The state transition diagram for expression execution in leak testing is presented in figure 8.

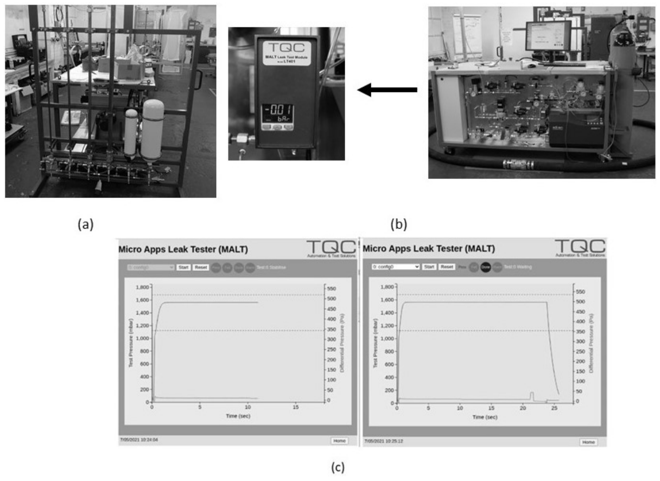
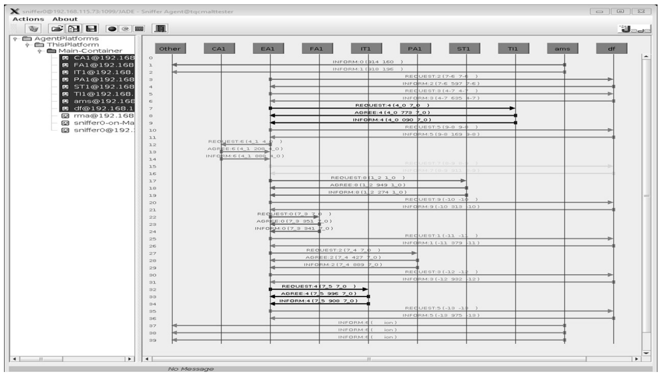


Fig. 6. The leak testing setup: (a) the cylinder volumes under test (b) MALT testing production system being a part of test bench for general leak testing (c) Interface for leak testing; agent system drives the execution by operational and conditional expressions

V. CONCLUSION AND FUTURE WORK

The research presents an application case for agent integration in leak testing production systems. This integration leads to control guided by expressions hosted in cloud services. The expressions proposed are of two types; operational and conditional expressions. Operational expressions define the



```
sniffer0@192.168.115.73:1099/JADE - Sniffer Agent@tqcmalttester
Actions About
AgentPlatforms
  ThisPlatform
    Main-Container
      CA1@192.168
      FA1@192.168
      IT1@192.168.
      PA1@192.168
      ST1@192.168
      TI1@192.168.
      ams@192.168
      df@192.168.1
      rma@192.168
      sniffer0-on-Ma
      sniffer0@192.

Other  CA1  EA1  FA1  IT1  PA1  ST1  TI1  ams  df

0
1  INFORM:0 (914 160 )
2  INFORM:1 (918 196 )
3  REQUEST:2 (7-6 7-6 )
4  INFORM:2 (7-6 597 7-6)
5  REQUEST:3 (4-7 4-7 )
6  INFORM:3 (4-7 635 4-7)
7  REQUEST:4 (4_0 7_0 )
8  AGREE:4 (4_0 773 7_0)
9  INFORM:4 (4_0 090 7_0)
10 REQUEST:5 (9-8 9-8 )
11 INFORM:5 (9-8 169 9-8)
12 REQUEST:6 (4_1 4_0 )
13 AGREE:6 (4_1 208 4_0)
14 INFORM:6 (4_1 888 4_0)
15 REQUEST:7 (8-9 8-9 )
16 INFORM:7 (8-9 911 8-9)
17 REQUEST:8 (1_2 1_0 )
18 AGREE:8 (1_2 949 1_0)
19 INFORM:8 (1_2 274 1_0)
20 REQUEST:9 (-10 -10 )
21 INFORM:9 (-10 313 -10)
22 REQUEST:0 (7_3 7_0 )
23 AGREE:0 (7_3 351 7_0)
24 INFORM:0 (7_3 341 7_0)
25 REQUEST:1 (-11 -11 )
26 INFORM:1 (-11 379 -11)
27 REQUEST:2 (7_4 7_0 )
28 AGREE:2 (7_4 427 7_0)
29 INFORM:2 (7_4 889 7_0)
30 REQUEST:3 (-12 -12 )
31 INFORM:3 (-12 932 -12)
32 REQUEST:4 (7_5 7_0 )
33 AGREE:4 (7_5 996 7_0)
34 INFORM:4 (7_5 908 7_0)
35 REQUEST:5 (-13 -13 )
36 INFORM:5 (-13 975 -13)
37 INFORM:6 ( ion)
38 INFORM:6 ( ion)
39 INFORM:6 ( ion)

No Message
```

Fig. 7. Sequence of Agents Execution of Leak Testing Process for 0.1 Litre volume

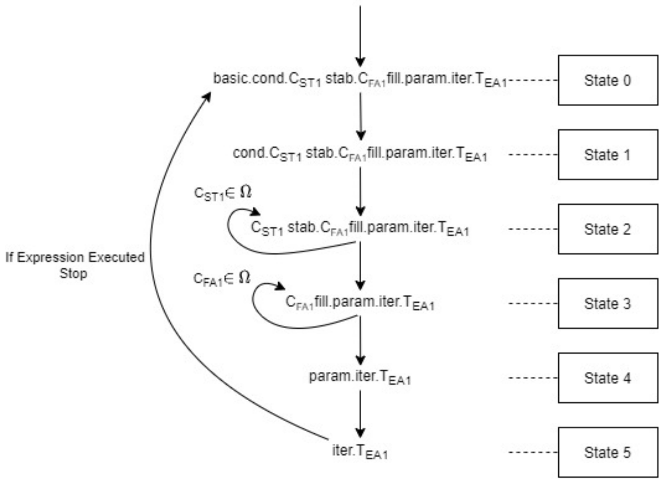

basic.cond.C_{ST1} stab.C_{FA1}fill.param.iter.T_{EA1}
State 0
cond.C_{ST1} stab.C_{FA1}fill.param.iter.T_{EA1}
State 1
$C_{ST1} \in \Omega$
C_{ST1} stab.C_{FA1}fill.param.iter.T_{EA1}
State 2
If Expression Executed Stop
$C_{FA1} \in \Omega$
C_{FA1}fill.param.iter.T_{EA1}
State 3
param.iter.T_{EA1}
State 4
iter.T_{EA1}
State 5

Fig. 8. State transition diagram for expression execution in leak testing. Here Ω is an handshake event that is hidden from the operator

skill requirement for the operation along with the sequence of execution of those skills, Conditional expressions detail the manner or behaviour of these skill execution. The agents can get these expressions from cloud service. This approach makes it suitable for customising operation of each part, along with scaling test skills as well as conditions as per requirements. As the expressions are derived from the cloud-based service so they can be modified, scaled or removed as per need.

Future work for this approach involves expanding the cloud based service with other service oriented architecture components. Machine Learning can be integrated to provide expressions for self-configurations and optimisation. Agent functionality will be expanded to factor in negotiation among agents and competing skill dependencies.

Currently the approach directs the skill selection and behaviour execution trough pre-loaded expressions made available to agents. This will be expanded upon to develop some strategy and methodology to modify the expressions as per data gathered and state observed. Rules for expressions will be researched. Logging and monitoring functionality will be explored.

References

[1] X. V. Wang and X. W. Xu, "An interoperable solution for cloud manufacturing," *Robotics and computer-integrated manufacturing*, vol. 29, no. 4, pp. 232–247, 2013.

[2] J. L. Chirn and D. C. McFarlane, "A holonic component-based approach to reconfigurable manufacturing control architecture," in *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, 2000.

[3] P. Leitão, J. Barbosa, and D. Trentesaux, "Bio-inspired multi-agent systems for reconfigurable manufacturing systems," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 5, pp. 934–944, 2012.

[4] O. Cardin, W. Derigent, and D. Trentesaux, "Evolution of holonic control architectures towards industry 4.0: A short overview," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1243–1248, 2018.

[5] A. R. Biswas and R. Giaffreda, "IoT and cloud convergence: Opportunities and challenges," in *2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, 2014.

[6] H. Theuer, N. Gronau, and S. Lass, "The impact of autonomy on lean manufacturing systems," in *Advances in Sustainable and Competitive Manufacturing Systems*. Springer, 2013, pp. 1413–1423.

[7] E. Bajic and F. Chaxel, "Holonic manufacturing with intelligent objects," in *International Conference on Information Technology for Balanced Automation Systems*. Springer, 2002, pp. 297–304.

[8] P. Leitão and F. Restivo, "Towards autonomy, self-organisation and learning in holonic manufacturing," in *International Central and Eastern European Conference on Multi-Agent Systems*. Springer, 2003, pp. 544–553.

[9] H.-S. Park and N.-H. Tran, "Autonomy for smart manufacturing," *Journal of the Korean Society for Precision Engineering*, vol. 31, no. 4, pp. 287–295, 2014.

[10] J. W. Rozenblit and W. Jacak, "Towards design and control of high autonomy manufacturing systems," in *Proceedings of the Third Annual Conference of AI, Simulation, and Planning in High Autonomy Systems' Integrating Perception, Planning and Action'*. IEEE Computer Society, 1992, pp. 38–39.

[11] P. Antsaklis, "Setting the stage: Some autonomous thoughts on autonomy, introduction to the panel discussion: Autonomy in engineering systems: What is it and why is it important," in *Proceedings of the ISIC/CIRA/ISAS*, vol. 98, 1998, pp. 520–521.

[12] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," *Computers & Chemical Engineering*, vol. 47, pp. 145–156, 2012.

[13] A. Kusiak, "Fundamentals of smart manufacturing: A multi-thread perspective," pp. 214–220, 1 2019.

[14] N. Radziwill, "Designing a Quality 4.0 Strategy and Selecting High-Impact Initiatives," in *ASQ Quality 4.0 Summit, Disruption, Innovation and Change.Dallas: ASQ*, 2018.

[15] Z. Kui, C. Yaru, Z. Hailong, A. Shengbiao, and H. Jie, "Construction of intelligent testing system for electronic components," in *Journal of Physics: Conference Series*, vol. 1693, no. 1. IOP Publishing, 2020, p. 012215.

[16] A. Gunasekaran, "An integrated product development-quality management system for manufacturing," *The TQM Magazine*, 1998.

[17] D. Wu, D. W. Rosen, and D. Schaefer, "Cloud-based design and manufacturing: status and promise," in *Cloud-based design and manufacturing (CBDM)*. Springer, 2014, pp. 1–24.

[18] F. Ning, W. Zhou, F. Zhang, Q. Yin, and X. Ni, "The architecture of cloud maufacturing and its key technologies research," in *CCIS2011 - Proceedings: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 2011.

[19] P. Helo, M. Suorsa, Y. Hao, and P. Anussornnitisarn, "Toward a cloud-based manufacturing execution system for distributed manufacturing," *Computers in Industry*, vol. 65, no. 4, pp. 646–656, 2014.

[20] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

[21] V. Botti, A. Omicini, S. Mariani, and V. Julian, *Multi-agent systems*. MDPI-Multidisciplinary Digital Publishing Institute, 2019.

[22] C. Yang and J. Sun, "Research on negotiation of manufacturing enterprise supply chain based on multi-agent," *Journal of Internet Technology*, vol. 20, no. 2, pp. 389–398, 2019.

[23] C. Yang, R. Yang, T. Xu, and Y. Li, "Negotiation model and tactics of manufacturing enterprise supply chain based on multi-agent," *Advances in Mechanical Engineering*, vol. 10, no. 7, p. 1687814018783625, 2018.

[24] G. Guizzi, R. Revetria, G. Vanacore, and S. Vespoli, "On the open job-shop scheduling problem: a decentralized multi-agent approach for the manufacturing system performance optimization," *Procedia CIRP*, vol. 79, pp. 192–197, 2019.

[25] Y. Liu, L. Wang, Y. Wang, X. V. Wang, and L. Zhang, "Multi-agent-based scheduling in cloud manufacturing with dynamic task arrivals," *Procedia CIRP*, vol. 72, pp. 953–960, 2018.

[26] A. Thomas, T. Borangiu, and D. Trentesaux, "Holonic and multi-agent technologies for service and computing oriented manufacturing," *Journal of Intelligent Manufacturing*, vol. 28, no. 7, pp. 1501–1502, 2017.

[27] A. Sarkar and D. Šormaz, "Multi-agent system for cloud manufacturing process planning," *Procedia manufacturing*, vol. 17, pp. 435–443, 2018.

[28] D. J. Mount, "Trends in leak testing: advancing the state-of-the-art," *Quality*, vol. 54, no. 2, pp. S8–S8, 2015.