A Multi–Agent System for Industrial Fault Detection and Repair

Vincenzo Bevar[1], Stefania Costantini[2], Arianna Tocchio[2], Giovanni De Gasperis[2]

Abstract A Multi Agent System is described, capable of monitoring a telecommunication industrial test & measurement setup, designed as an application of the DALI agent language. The autonomy of the MAS is necessary to supervise the measurement apparatus during off-work time without human intervention, increasing the quality and efficacy of the overall test procedure. The MAS can decide whether to recover or repair the set of software process needed to achieve a correct test sequence without user intervention.

Key words: Fault tolerance, Automatic Test Systems, Logical Agents

1 Introduction

Nowadays, many kinds of applications need some degree of autonomy. There are application contexts that actually offer no alternative to autonomous software. Agents provide a tool for structuring an application in a way that supports its design metaphor in a direct way. Agents and multi-agent systems (MAS) have emerged as a powerful technology to face the complexity of a variety of ICT scenarios. Agents technology and methods have been thoroughly reviewed in many papers, among which we mention [1, 2, 3, 4]. There are now several industrial applications that demonstrate the advantage of using agents, e.g., in the manufacturing process [5], or even about planning and scheduling how to best cut wood in order to minimize loss of natural material [6]. The telecommunication sector has also seen a significant amount of effort on agent technology since the 1990's. The development of agent systems requires dedicated basic concept and languages: at the level of indi-

[1] Techolabs, R. & D. Strada Statale 17, L'Aquila, Italy, e-mail: vincenzo.bevar@technolabs.it
[2] Dipartimento Ingegneria e Scienze dell'Informazione e Matematica, Universitá degli Studi dell'Aquila, Via Vetoio 1, 67100 L'Aquila,
e-mail: (stefcost,tocchio,giovanni.degasperis)@univaq.it

vidual agents, representational elements such as observations, actions, beliefs, goal are required. Typical features of agents are reactivity and proactivity, not found in conventional controllers. At the MAS level, communication, coordination and social aspects such as joint goals and trust requirements need to be be expressed. Going further on the line of autonomous software, new applications need "intelligence" in the sense of the ability to exhibit, compose and adapt behaviors, and being able to learn the appropriate way of performing a task rather than being instructed in advance. Logic languages are evolving from static to "active", and in the last years have been enriched with new capabilities based on the "agency" metaphor. Due to their traditional "fast prototyping" character, to new efficient implementations and to the new concepts they are able to embody, they are good candidates for experimenting such advanced applications. A good survey about current state and future directions of logic languages in multiagent systems can be found in [7, 8] and in [9]. These papers emphasize how agent-oriented language may provide an affordable way of introducing the engineering of intelligent behaviors into software engineering and development practice. Among the industrial applications, process control appears to be potentially a natural application for agents, by virtue of controllers being in principle autonomous entities [10]. This kind of application implies measuring a system, so as to verify whether specific measurable values are within a pre-defined range, and acting on the system so as to keep specific observable values stay within the range characterizing an acceptable behavior of the system itself. Measuring a system implies selecting the correct checks to perform at each stage. Controlling the system implies being able to either modify or restore its operational parameters and behavior as required. Agents can replace a human operator in this kind of task. If the controlled system is composed of several parts, single agents can control the various parts, and can cooperate so as to enforce the system overall behavior. In this paper, we discuss the use of a logic active agent-oriented language for the development of an industrial application in the field of process control. The case study is focused on an Automatic Test System (ATS) developed by Technolabs (L'Aquila, Italy) and we will discuss how its features has been enhanced by the agent technology, so as to recover the ATS system execution from unexpected events, thus resulting in an Automatic Test-recovery Method (ATM). At the core of the MAS implementation the DALI agent-oriented language [11, 12, 13, 14, 15] has been adopted. DALI is a long term project developed at the Computer Science Department of the University of L'Aquila.

2 Technolabs Automatic Test System

Technolabs is a R&D Lab (located in L'Aquila, Italy) specialized in design and development of telecommunication equipments for transport networks. Technolabs adopts industrial design and production processes that comply with consolidated factory standards, in order to guarantee good quality levels satisfying market requirements. To this aim, new testing techniques have been recently introduced in

order to improve the dependability of equipment which has to be connected to the carrier network. Below we describe the current testing methodology. We then identify how to improve this methodology by adopting the agent technology. Testing a HW/SW system requires at each stage the selection of a relevant test procedure (i.e., the Test Selection phase) and its execution (i.e., the Test Execution phase). The Automatic Test System (ATS) is a methodology developed by the Systems Integrations & Test area of Technolabs, aimed at handling both activities. In particular, ATS allows both phases to be performed without the intervention of a human technical supervision. The ATS architecture contains five main HW/SW components, as shown in Figure 1:

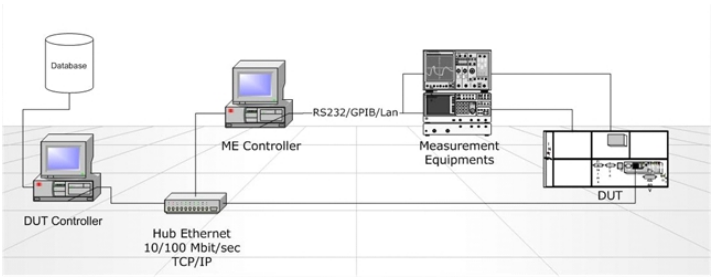

Fig. 1 The Automatic Test System Architecture. **DUT** device under test, DUT Controller, **ME** Measuring Equipments which collect and check observable values from the DUT, the ME Controller connected with the ME

The device which is currently under test *(DUT)* is the MSI FP (Multi-Services Integrator Flat Pack), which is a multi-service platform used in metropolitan and regional carrier networks for optimizing packet transportation across legacy networks. The DUT controller is composed by two components: (1) the Technolabs Local Craft Terminal (LCT), which offers the capability to remotely manage telecommunication equipments through a graphical user interface. It can be used for coping with faults, monitoring traffic and for the configuration of the equipment. (2) WinRunner, an HP software application which allows emulating in a graphical way the actions of a human operator as if (s)he were really present at the test bench. Using scripts written in a C-like language called TSL (Test Script Language), it can send command sequences to both the DUT Controller and to the ME Controller in order to perform measurements in the test session; it acts like a human operator in a fast, efficient and, above all, repeatable way. In the ATS system, WinRunner is used to configure the DUT (through the LCT application) and to synchronize the operation done on the DUT, sending, if necessary, suitable commands through the LAN to the ME Controller. The task of the *Measurement Equipment* is twofold: verify the system performance compliance to requirements and check the equipment state by detecting failures in the service quality. Measuring instruments that may be used for testing the DUT are an oscilloscope or an SDH analyzer, which allows one to evaluate signal quality of data transmitted and received and to detect and propagate alarm. The *ME Controller* is a software application developed by Technolabs

(Instrument Server) which allows the management and remote control of test and measurement equipments. In the ATS architecture, the ME Controller is remotely controlled by WinRunner. The ATS acts over both the DUT Controller and the ME Controller, setting up both the DUT and the ME, executing test suites, and storing tests results in files (database files) that can be examined off line also remotely through an Internet/Intranet connection.

3 Enhancing ATS by Agent Technology: Motivations

During the test of HW/SW systems, the most important activity is the test execution flow integrity and correctness. Often, a test lasts many hours and, for optimization reason of the resources occupation and costs reduction, it is executed during night-time, in order to have test results available the day after. Then, all test sessions are to be performed without human supervision. However, while testing hardware/software prototypes it may happen that a software process unexpectedly terminates during the night or the week-end, thus blocking any testing activity until the next working day. This of course strongly reduces the testing efficiency and increases costs. To improve reliability and enhancing features of the ATS, an agent-based solution has been chosen for two main reasons: (1) each component of the ATS system has to be supervised individually in autonomous way, reacting to every state change, detecting these changes and taking appropriate measures. An agent is able to perform this task. (2) The distributed nature of the ATS that implies the need for each agent to communicate the events occurred to the other agents. Then, the agents supervising single components have to form a MAS which has the overall objective of coordinating ATS activities and restoring its functionalities also in case of critical situations. Supervisor agents should be intelligent and proactive, so as to properly combine the abilities of checking and restoring ATS activities. Logical agents have been chosen in a first stage due to the fast-prototyping nature of the selected language, i.e., the logic-based language DALI (defined and implemented at the Computer Science Department of the University of L'Aquila) [16]. In the present experimentation, DALI has proven to be efficient, reliable and flexible.

4 Automatic Test recovery Method

Below we present the Automatic Test recovery Method (ATM), implemented in DALI, which has the aim of recovering the ATS system execution from unexpected events, such as devices crashes. ATM automatically detects unexpected termination of HW/SW devices through continuous system monitoring, restarts the system and resumes the test procedure execution. It extends the basic architecture of an ATS, by adding a multi-agent system which automatically checks the test procedures and equipment execution status. The new architecture is shown in details in the com-

panion demonstrator paper. It includes this set of agents: *Executor*, *Master*, *Slaves*. A slave agent monitors a specific ATS software component and informs the master agent about crashes. It may also execute recovery when authorized by the master agent (by means of a special message). A master agent monitors the slave agents and receives messages from them about environment changes. On the basis of received messages, it decides whether it is possible to recover the hardware/software crash happened in the ATS.

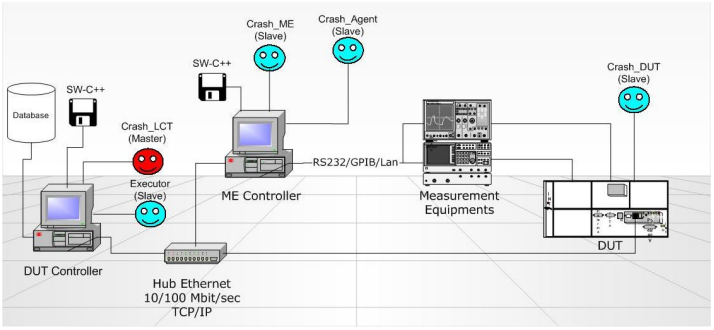

Fig. 2 The Automatic Test recovery Mechanism Architecture

4.1 Design

Two steps are needed for the involved agents to provide the required capabilities: Step 1: Perception phase; Step 2: Reaction and Action phase. Each involved agent perceives its related environment and devises consequent actions whenever some (in this case unwanted) change occurs.

Perception phase: the agent must control if the process corresponding to the LCT application is in an active state, in order to perceive the LCT state changes.

Reaction and Action phase: the agent can either enter a reaction/action phase or remain in the perception phase, depending on different execution scenarios: if LCT is in a crash state, then the agent reacts in order to restore its correct functioning by running the "restart the TNMS-CT application" command by which the agent emulates, through the WinRunner application, all the steps a human operator would perform in order to restart the LCT. If LCT is not in crash state, the agent remains in the perception phase, waiting for possible state changes.

The above-described MAS has been fully implemented, tested and experimented with very good results. Actually, the MAS is able to replace a human operator with high reliability. By exploiting advanced features of DALI [17] the involved agents can self-monitor themselves so as to keep their own behavior within the expected range. The full set of DALI programs that implements the multi-agent system can be

obtained from the authors, and can be run on the DALI interpreter [16]. More details are available in the companion demonstrator paper. The overall system behavior, measurement quality improvement and temporal charts, for lack of space will be presented at a later extended paper.

5 Related Work

In the telecommunication sector, TILAB (Telecom Italia Lab) developed and distributes JADE (Java Agent DEvelopment Framework, cf. http://jade.tilab.com/), a software framework fully implemented in Java language. JADE [18] simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA standard for communication and provides a set of tools that support the debugging and deployment phases. Telecom Italia LAB uses the JADE platform for several internal projects of interest for the Telecom Italia Business Units, but also other companies in the last years adopted JADE for their applications: BT Exact is developing, on top of the JADE platform and the LEAP add-on for mobile terminals, an application supporting the coordination and the activities of a mobile workforce, including the distributed scheduling of jobs, job management on the fly, travel and knowledge management, and location-based coordination. In the health care field, the JADE team collaborates with Swisstransplant, the Swiss National Transplant Coordination center for organ transplants, in order to develop an agent-based system for decision making support in organ transplant centers. Singular Software SA uses JADE in the context of the IST project "Intelligent Mobility Agent for Complex Geographic Environments". JADE, moreover, has been used in Acklin B.V. and Fraunhofer IITB projects. FactoryBrokerTM is a solution to Holonic Control System composed by mechatronics autonomous components that have relevant *responsibility*. Agent technology also in this case fits very well. Some groups with a strong business science orientation are incorporating DAI ideas in the area of financial services (ALLFIWIB and MASIFprojects). Moreover, we mention the works by Leckie et al. [19] and by Friedman-Hill [20]. In the first one the authors describe a prototype agent-based system for performance monitoring and fault diagnosis in a telecommunications network. In the second one Friedman-Hill introduces JESS, the Java Expert System Shell used to realize a system useful for monitoring all processes, instrumentation and data flows of the Kennedy Space Center's Launch processing System. In both cases, the adopted language is not a logical language, so no direct comparison with our approach is possible.

Among the many interesting applications of agents, we mention: co-operative supervision systems for energy management and distribution at Atlas-Elektronik in the context of the ARCHON project; dynamic cargo allocation for forwarding agencies at Univ. Erlangen-Nuernberg and at DFKI in cooperation with Daimler-Benz; loading dock scenarios at DFKI; cooperative traffic management(KIK-Teamware); traffic management at FZI Karlsruhe; group appointment scheduling at DFKI (KIK-Teamware and AKA-Mod) and at ECRC.

6 Concluding Remarks

The DALI language has proved to be a competitive tool for building intelligent agents able to work in a real contexts. The application presented and discussed in this paper and in the companion demo paper allows us to argue that logical agents can face complex problems where reactivity and pro-activity must be sapiently interleaved. In many industrial applications, logical agents are not yet used because their capability to reason and to pro-act autonomously is seen as a threaten for the control of applications. This problem, however, has been greatly alleviated by approaches to logical agents verification and self-verification, such as those of [21], [22], [23], [17] and others, proposing both static and run-time verification mechanisms. The possibility of experimenting logical autonomous agents in a critical context such as the Automatic Test System in Technolabs has been very important for demonstrating that logical reasoning potentiality (joined with reactive, pro-active and social behavior) can be applied with success in replacing critical human tasks. DALI agents, in particular, have sustained this task of crash control and system test exhibiting a high reliability and dependability. It can be observed that DALI has been usefully adopted in other complex applications, such as the one described in [24]. The present work has taken profit also from the experience of [25], where DALI had been used for developing a complex MAS aimed at supervising software systems.

We thank Prof. Henry Muccini for helping the team during the startup of the project and for insightful discussion afterwards, and Dr. Manuele Colarossi for embedded software development and testing involved in the Thesis work for his BSc degree in Computer Science.

References

1. H. S. Nwana , *Software Agents: An Overview*. In: *Knowledge Engineering Review*, Vol. 11, No 3, pp.1-40, Sept 1996 Cambridge University Press.
2. M. Wooldridge, N. R. Jennings, *Intelligent Agents: Theory and practice*, In: *The Knowledge Engineering Review*, 10(2):115-152, 1995.
3. K. P. Sycara, *Multiagent Systems*, In: *AI Magazine* vol. 19(2), pp. 79-92, 1998.
4. M. Fisher, R. Bordini, B. Hirsch and P. Torroni, *Computational logics and agents: A road map of current technologies and future trends* In: Computational Intelligence Vol: 23, No:1, pp. 61-91, 2007
5. Y. Chen, L.-J. Zhang and Q. Wang *Intelligent scheduling algorithm and application in modernizing manufacturing services*, In: Proceedings - 2011 IEEE International Conference on Services Computing, pp 568-575, 2011
6. E. Elghoneimy, W.A. Gruver, *Intelligent decision support and agent-based techniques applied to wood manufacturing* In: Advances in Intelligent and Soft Computing Vol: 91 pp 85-88, 2011
7. P. TORRONI, *Computational Logic in Multi-Agent Systems: recent advances and future directions*, In: *Annals of Mathematics and of Artificial Intelligence*, vol. 42, pp. 293-305, 2004.
8. Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., Torroni, P., *Agents, Multi-Agent Systems and Declarative Programming: What, When, Where, Why, Who, How?*, 25 Years GULP, Vol. 6125, Springer, 204-230, 2010.
9. Dal Pal, A., Torroni, P.: *25 Years of Applications of Logic Programming in Italy*, In: 25 Years GULP. Vol. 6125, Springer, pp. 300-328, 2010.

10. Åstrom, K.J., Wittenmark, B.: *Computer-Controlled Systems.Theory and Design*, Prentice Hall Internal Inc. (1990)
11. S. Costantini. *Towards active logic programming*, In A. Brogi and P. Hill, editors, *Proc. of 2nd International Workshop on component-based Software Development in Computational Logic (COCL'99)*, PLI'99, 1999.
12. Costantini, S., Tocchio, A.: *A logic programming language for multi-agent systems*, In: Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf.,JELIA 2002. LNAI 2424, Springer-Verlag, Berlin (2002)
13. Costantini, S., Tocchio, A.: The DALI logic programming agent-oriented language, In: Logics in Artificial Intelligence, Proc. of the 9th European Conference, Jelia 2004. LNAI 3229, Springer-Verlag, Berlin (2004)
14. S. Costantini, A. Tocchio and A. Verticchio. *A Game-theoretic operational semantics for the DALI Communication Architecture*. In: M. Baldoni, F. De Paoli, A. Martelli and A. Omicini, (eds.). Proceedings of WOA04, Pitagora Editrice (2004) Bologna, ISBN: 88-371-1533-4 , Also available on-line, at the URL: http://woa04.unito.it/Pages/atti.html. Accessed January 8th 2012
15. Costantini, S., Tocchio, A.: *About declarative semantics of logic-based agent languages*, In Baldoni, M., Torroni, P., eds.: Declarative Agent Languages and Technologies. LNAI 3229. Springer-Verlag, Berlin (2006) Post-Proc. of DALT 2005.
16. Costantini, S., D'Alessandro, S., Lanti, D., Tocchio, A. with the contribution of many undergraduate and graduate students of Computer Science, L'Aquila.: DALI web site, download of the interpreter (2010)
http://www.di.univaq.it/stefcost/Sito-Web-DALI/WEB-DALI/index.php. Accessed January 8th, 2012
For beta-test versions of the interpreter (latest advancements) please ask the authors.
17. Costantini, S., Dell'Acqua, P., Pereira, L.M., Tsintza, P.: *Runtime verification of agent properties*, In: Proc. of the Int. Conf. on Applications of Declarative Programming and Knowledge Management (INAP09). (2009)
18. Bellifemine, F., L., Caire, G., Greenwood, D. *Developing Multi-Agent Systems with JADE*, John Wiley & Sons, Hoboken, NJ, USA, 2007
19. Leckie, C., Senjen, R., Ward, B., Zhao, M., *Communication and coordination for intelligent fault diagnosis agents*, In:Proceedings Eighth IFIP/IEEE International Workshop for Distributed Systems Operations and Management, pp. 21-23, 1997.
20. Friedman-Hill, E., *Jess in Action: Java Rule-Based Systems*, In: Action series, Manning Publications, 2002.
21. Kacprzak, M., Lomuscio, A., Penczek, W.: *Verification of multiagent systems via unbounded model checking*, In: Proc. of the Third Int. Joint Conf. on Autonomous Agents and Multiagent Systems, AAMAS '04, ACM Press, New York, NY pp. 638-645 (2004)
22. Fisher, M.: *Model checking AgentSpeak*, In: Proceedings of the Second Int. Joint Conf. on Autonomous Agents and Multiagent Systems AAMAS03. LNCS 3862, ACM Press 409-416 (2003)
23. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: *Verifiable agent interaction in abductive logic programming: The sciff framework*, ACM Trans. Comput. Logic **9** 29:1–29:43 (2008)
24. Costantini, S., Mostarda, L., Tocchio, A., Tsintza, P.: *Dalica agents applied to a cultural heritage scenario*, IEEE Intelligent Systems, Special Issue on Ambient Intelligence **23**(8) (2008)
25. Castaldi, M., Costantini, S., Gentile, S., Tocchio, A.: *A logic-based infrastructure for reconfiguring applications*, In Leite, J.A., Omicini, A., Sterling, L., Torroni, P., eds.: Declarative Agent Languages and Technologies, revised selected papers presented at DALT 2003. LNAI 2990, Springer-Verlag, Berlin, Hot Topics Sub-series. (2004)