Chapter 4
Knowledge Based Modules for Adaptive Distributed Control Systems

Andrea Ballarino, Alessandro Brusaferri, Amedeo Cesta, Guido Chizzoli, Ivan Cibrario Bertolotti, Luca Durante, Andrea Orlandini, Riccardo Rasconi, Stefano Spinelli and Adriano Valenzano

Abstract Modern automation systems are asked to provide a step change toward flexibility and reconfigurability to cope with increasing demand for fast changing and highly fragmented production—which is more and more characterising the manufacturing sector. This reflects in the transition from traditional hierarchical and centralised control architecture to adaptive distributed control systems, being the latter capable of exploiting also knowledge-based strategies toward collaborating behaviours. The chapter intends to investigate such topics, by outlining major challenges and proposing a possible approach toward their solution, founded on autonomous, self-declaring, knowledge-based and heterarchically collaborating control modules. The benefits of the proposed approach are discussed and demonstrated in the field of re-manufacturing of electronic components, with specific reference to a pilot plant for the integrated End-Of-Life management of mechatronic products.

4.1 Scientific and Industrial Motivations

Since several years, the manufacturing industry has been facing a number of technological and production challenges related to the increasing variability of mix and demand of products driven by a short product life cycle. Nevertheless, the growing industrial demand for increased levels of reconfigurability—having impact both in production process automation, supervision and data collection and aggregation systems—is not properly fulfilled, mainly due to the lack of widely accepted solutions

A. Ballarino (✉) · A. Brusaferri · G. Chizzoli · S. Spinelli
CNR-STIIMA, Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato, Milan, Italy
e-mail: andrea.ballarino@stiima.cnr.it

A. Cesta · A. Orlandini · R. Rasconi
CNR-ISTC, Istituto di Scienze e Tecnologie della Cognizione, Rome, Italy

I. C. Bertolotti · L. Durante · A. Valenzano
CNR-IEIIT, Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Turin, Italy

T. Tolio et al. (eds.), *Factories of the Future,*
https://doi.org/10.1007/978-3-319-94358-9_4

and integrated reference models. In fact, implemented solutions are today typically characterized by rigid hierarchical structures, organized into strictly coupled layers.

Centralized PLCs (Programmable Logic Controller) architectures are often deployed, integrating the overall control logic of the production line for coordinating the execution of single mechatronic devices and machines. Such monolithic approach represents one of the major obstacles to achieve short-time reconfigurations. In fact, without a proper modularization and standardization of the control application entities, adaptations of automation system behaviour—to properly support required changes of factory production assets—result to be strongly time consuming for control system engineers and system integrators. Furthermore, the lack of modularization and standardization critically impacts on readability, portability and integration of single control modules across different applications, thus preventing the capitalization and re-use of company specific know-how on production process.

Further to this, modifications or extensions within the programs of the production line PLC possibly leads to the introduction of code errors, thus requiring extensive validation of complete automation system before deployment. Nevertheless, such activity is fundamental to avoid unpredicted deviations of the process execution, which could cause damages to devices/machines or, more critically, to human operators. As a result, existing industrial control systems are often very inefficient in facing the ever increasing demand for flexibility, expandability, agility and reconfigurability, which are typical requirements of advanced manufacturing system solutions [1, 2].

The previous requirements are fundamental to ensure a modern approach to efficient manufacturing, whose major competitiveness pillars rely upon: high value added goods, knowledge intensive production processes, and efficient and safe operating environments. This extremely articulated context is a very promising and appealing opportunity for countries heavily involved in the production of capital goods, machine tools and sophisticated solutions for manufacturing systems. A key capability of the control system is to evolve over time in order to anticipate and persistently adapt the control logics, functions and architectures to the evolving production scenarios. Therefore, flexibility in control infrastructure shall be prosecuted at the level of single control unit to leverage the hardware reconfigurability by exposing a set of (automation) functions/tasks that the single machine or device can execute to support production objectives. This latter aspect could effectively help in transforming monolithic and hardcoded solutions toward service oriented approaches. Yet this transformation cannot be fully tackled without considering that the execution of single task or function on a machine could either address the interaction with physical actuators, or imply a synchronization problem between different devices. Therefore, a real-time software infrastructure and communication framework shall be developed to properly support reliability within industrial automation field.

This chapter addresses the challenges related to the conception and development of next generation control systems, with reference to their application in Reconfigurable Manufacturing Systems (RMSs), as a viable solution while addressing varying production conditions within the manufacturing industry. The proposed approach, named *Generic Evolutionary Control Knowledge-based mOdule* (GECKO), aims at

developing a flexible distributed control infrastructure, based on the interactive cooperation of control modules. GECKO enables each device (e.g. end-effector, complex machine equipment, integrated cell or even system) to evolve from a stand-alone, rigidly and hierarchically managed condition to an autonomous, self-declaring, hierarchically interacting and collaborating scenario. GECKO modules are conceived to detect and interpret the production environment characteristics and to adapt its capabilities on the basis of specific requirements by automatically accomplishing local and global objectives.

In consideration of aforementioned topics, the next sections will present the relevant state of the art (Sect. 4.2) and then the context and reference problem (Sect. 4.3). The overall approach toward an adaptive control infrastructure is first described (Sect. 4.4), and then discussed with particular reference to its realization via the GECKO solution (Sect. 4.5). A specific focus on Reconfigurable Transportation Systems (RTSs) will be kept as target application for proposed GECKO solution (Sect. 4.6), since RTSs can play a relevant role in embedding manufacturing systems with the capability of adapting their structure and functionalities to the evolving needs of production processes. Finally, the conclusions are presented in Sect. 4.7.

4.2 State of the Art

Traditional control systems based on hierarchical and centralized control structures hold good performance in terms of productivity over a limited and specific range of conditions. Nonetheless, such large monolithic software packages typically need relevant modifications of the control code to cope with any sort of (even minimal) system adaptation and reconfiguration. As a result, they are very inefficient to face the current requirements of flexibility, expansibility, agility and re-configurability required by advanced manufacturing system solutions.

When considering the existing scientific and industrial practices to evolve from current centralised approaches, three major topics shall be addressed, namely interoperable and pluggable (mostly defined as *plug and play*, P&P) control solutions, design of automatically reconfigurable control solutions as well as implementation of optimization strategies and learning mechanisms in the control.

In the field of P&P solutions, a number of EU projects [3–5] and scientific papers [6, 7] propose advanced solutions mostly related to specific devices (such as robots) or to specific enablers (such as the communication layers). The industrial manufacturing practice limits the usage of P&P concepts to very basic applications mostly related to the connection of devices preliminarily connected to a communication network, mainly disregarding the phases of automatic discovery and capability assessment, as well as of dynamic cooperation. Yet control entities—conceived to control mechatronic devices so that the former can be automatically run as soon as the device is physically plugged in the system dorsal—still represent an objective to be achieved. This basically implies addressing a number of challenging automation solutions under specific perspectives such as:

- standard architecture designed as a component-based software solution structured with standardized interconnected components, each one dealing with different tasks and goals;
- cooperation and interaction mechanisms between different control entities rely upon both standardized physical and logic interfaces—ensuring the possibility for modules to operate in multiple configurations and to be nested on different resources—and a common vocabulary, an open ontological classification of devices and their functionalities as well as a shared knowledge representation system [8, 9]—guaranteeing that the different components of the system use a consistent and shared representation of products, processes and resource data, as well as semantics and rules characterizing the information interrelations [10, 11];
- an open communication network enabling both the physical and logical connections of the control module to the plant, providing Timeliness, Security and Energy Efficiency properties.

Regarding the first topic, the conception and deployment of reconfigurable solutions have mostly concentrated on the mechanical aspects of the devices, leading to the realization of reconfigurability enablers targeting several applications, from machine tools and robots, to transportation systems and fixturing systems based on standard interfaces and flexible CNC [12]. Challenging industrial applications of these reconfigurability concepts [13] have been realized by machine tool builders (e.g. Panasonic, Zevac, Mori Seki), by providers of robotic solutions (e.g. Kuka, Mitsubishi, Robotnik) and by providers of modular equipment (e.g. Festo and Flexlink). These automation reconfigurability options basically consist in a set of predefined optional add-ons or a set of deterministic capabilities that are acquired since the beginning, thus endowing the resources with very high flexibility instead of reconfiguration enablers [14].

As already mentioned, the majority of automation applications is characterised by traditional centralized control architecture, with consequent sensible limitations in the achievement of agile adaptation to run-time production changes. Moreover, such limitations become even more critical when the complexity of the process to be controlled and the functionalities to be automated increase.

The challenging aspect in the reconfigurability concept consists in its concurrent applicability to the control with regard both to logic and physical aspects. Logic reconfigurations of the control should stem from the need to modify the control functions and strategies in response to a production change (e.g. new product), events altering pieces of equipment (e.g. resource failures or abnormal behaviours) or revised production goals (e.g. minimization of energy consumption vs. idle times). On the other side, physical reconfigurations imply that the control physical device and the related control software interact with a modified control system, where new entities have been integrated or dismissed. This firstly requires a more complex reconfiguration process that goes beyond the control logic features and deals with physical interconnections, communications and bindings. Beside this, a new functional-oriented architecture enabling modular automation is needed to realize

the encapsulation and (re-) use of self-contained functions and/or automation tasks as services.

SOA approaches can help to fulfil these requirements [15]. SOA is meant to provide services through a set of black box components exposing interfaces based on communication services [16]. In the field of modular control systems, single modules can be represented by such black box components, aimed at accomplishing functions by encapsulating services. The OASIS reference model was published for SOA, containing the definition of a service as a "…mechanism to enable access to one or more capabilities…" [17]. Services exposed by an entity are meant to be used by other entities. On top of this, the European research and development project named SOCRADES[1] addressed a novel manufacturing paradigm deeply based on SOA, and particularly on web services [18]. According to SOCRADES framework the intelligence of the manufacturing system is implemented by an agent-based approach embedded in smart devices. These devices collaboratively act at the same hierarchical level by using web services as an interface for mutual communication [19]. The autonomous units therefore operate cooperatively, each being intelligent and proactive. In addition to the more general SOA approaches described, Mendes et al. [20] proposed an approach for the implementation of service-oriented control in process industry based on adoption of high level Petri nets.

The separation of control functionalities into services supporting process control and intelligence can be found in [21]. The proposed hierarchical structure is similar to the one described in ISA 106 [22]. In [23] a SOA implementation approach based on the adoption of function blocks [24] is introduced. Basic or atomic services are developed as encapsulated elements, to be invoked either directly by request-response messaging or indirectly, when contained in other complex high level services. Furthermore, both actuators and sensors signals require the access to field data. Such access is also implemented by using services. While mutual connectivity between all devices is required, services do not need to be allocated to particular hardware. Function block types deployed according to [24] are used as service types. Connections between function blocks realize the messaging mechanism, and therefore, the communication between corresponding services. Communication contains message types and parameters, where the former are achieved by event connections and the latter by data connections. Therefore, the introduced approach decomposes complex manufacturing tasks to the very basic level. Each basic task is implemented as an atomic service. Atomic services can be combined into aggregated ones to achieve more complex functionalities: the resulting set of basic and aggregated services can be orchestrated by a central coordinator or by autonomous choreography [23]. A high control level can invoke a service by messaging and the invoked service can invoke downstream subservices in a specific order.

When referring to the second of the mentioned topics (i.e. optimization strategies and learning mechanisms), the goal of the control optimization process traditionally pertains to a number of aspects related to the products quality to be achieved, the technologies to be exploited, along with the physical equipment usage over time.

[1] http://www.socrades.net.

Coherently with the traditional way of modelling the factory as a number of linked layers, these optimization paths are handled at the Production Management and Manufacturing Execution System (MES) levels by the scientific and industrial communities [25–31]. In this view, the system logic control is conceived with a very little awareness of the shop-floor and limited options to perform dynamic control adaptations. This is evident also when focussing on part routing problems in manufacturing transportation systems, where context recognition and optimization issues become crucial. Existing control solutions are based on centralized/hierarchical control structures that offer good performance but they require a big effort to implement, maintain or reconfigure control applications when a high-level of flexibility is needed. Thus, classical approaches usually do not meet requirements of manufacturing systems in terms of flexibility, expansibility or reconfigurability. Current R&D efforts focus on the development of novel control systems characterized by distributed intelligence, robustness and adaptation to the changes in the environment and exogenous factors [32]. The multi-agent paradigm aims at addressing control objectives by introducing modularity, decentralization, autonomy, scalability and re-usability as main features. Even if the definition of agent concept is neither unique nor shared [33], its most important properties are the autonomy, intelligence, adaptation and cooperation [34].

Although Artificial Intelligence (AI) based approaches have been considered as local enablers in some RMSs [35], the design of control models considering all the possible failures and changing situations remains a crucial problem. Moreover, relevant structural modifications in an agent configuration entail a re-design of the control strategies, which is hard to manage on the fly. In manufacturing, knowledge-based approaches exploiting ontologies have been applied to increase flexibility in modelling and planning of mechatronic devices [36], resources in collaborative environments [37], automation and control systems [38], and to manage information of distinct types [39]. In these cases, planning specifications, if considered, are fixed and neither automatically generated nor subsequently adapted. In robotics, ontologies have been more widely exploited in the knowledge framework OMRKF [40], KnowRob [41], ORO [42, 43]. Nevertheless, none of the previous approaches addresses the issue of dynamically adapting the planning models. Other AI approaches based on Answer Set Programming have been proposed [44, 45].

From the software infrastructure point of view, the use of a real-time operating system (RTOS) as an execution framework in control applications is now becoming more and more popular with respect to full-custom designs, due to its clear advantages in terms of software development time and cost. RTOS can be divided into two main categories according to their application programming interface (API) that directly affects application development and portability:

- operating systems providing a full-fledged POSIX API, sometimes tailored to the specific requirements of embedded systems, as specified by the IEEE Standard 1003.13 [46]. For instance, this is the case of Linux;
- operating systems providing their own proprietary, and usually simpler and more efficient, API. A typical example belonging to this category is the FreeRTOS open-source, real-time operating system [47].

Among RTOS solutions, the POSIX interface is to be preferred when aiming at portable software and firmware suitable for long-term reuse and maintainability, because it is backed up by an international standard. As an additional benefit, it is widespread and well-known to programmers. Focusing on Linux, this option is made even more appealing by the increasing maturity of real-time extensions for the mainstream kernel, like the preemption Real-Time patch (Preempt_RT),[2] Real-Time Application Interface patch (RTAI),[3] and Xen-based virtualization support.[4]

POSIX interface is also in line with International Standards of the International Electrotechnical Commission (IEC) IEC 61158 and IEC 61784 which enable interoperability between devices of different manufacturers. Three types of industrial networks are currently available: Fieldbuses, Real-Time Ethernet Networks (RTE) and wireless networks. Moreover, these networks may be used in a combined, resulting in hybrid systems [48–50]. In addressing the specific choice for each application, a key requirement to be considered is the capability to guarantee traffic separation, a critical feature needed to avoid interference between traffic flows with different timing requirements.

4.3 Problem Statement

A new approach to the design of automation system is required to address the aforementioned requirements and challenges, while exploiting advanced engineering practices for control modularization and distribution. Novel integrated platforms for automation system development, supporting distributed control system engineering, represent a fundamental enabling technology, where openness, interoperability and compliancy with international industrial standards shall be considered as fundamental features to be guaranteed.

The adoption of a distributed automation approach can help to increase the reconfigurability and robustness of the automation system. Traditional centralized and hierarchical architectures should be replaced by new modular control solutions that can better support a fast integration of new functional components and run-time adaptation of the system to the dynamic change of production demand and requirements. The modularization concept enables the organization of the control solution into software entities, encapsulated within standard interfaces, that can be directly connected to functional components of the system to be controlled, so that the re-usability of the applications can be enhanced.

The following core requirements must be tackled by a novel approach based on distributed automation:

- *Interoperable and pluggable control solution*. Compared to the existing solutions, control entities shall be conceived to control mechatronic devices so that the for-

[2] https://rt.wiki.kernel.org/index.php/Main_Page.

[3] https://www.rtai.org/.

[4] https://www.xenproject.org/.

mer can be automatically run as soon as the device is physically plugged in the system dorsal. This P&P feature enables the automatic recognition and configuration process of the control modules; moreover, the interoperability characteristics should allow applying this procedure to every resource type and/or tasks, thus ensuring the possibility to interact with all the resources of the shop-floor.

- *Automatically reconfigurable control solutions*. As mentioned, the reconfigurability concept should handle both changes in the control logic and in physical aspects. The first aspect is aimed at matching the need to adjust the control functionalities as a consequence of change in production goals. The second aspect should tackle situation where the (control) system modified, since new entities have been integrated or dismissed. This requires a more complex process that manages both known a priori reconfigurations—by implementing pre-determined portions of the control which are not utilized until the occurrence of the activation event—and brand new reconfigurations—where the physical integration of a new entity recalls for system recognition phase to determine the automatic nesting of the new portion of the control within the overall architecture, as well as the updating of the knowledge of existing control module about system manufacturing capabilities and status.
- *Optimization processes and learning mechanisms*. The realization of distributed control solutions based on cooperating modules enables the implementation of enriched and comprehensive optimization strategies at control level, aimed at adapting the control functions and strategies on the basis of the actual status of the controlled device(s) and the knowledge about the nominal behaviour. In order to ensure the accomplishment of productivity, quality and energy efficiency targets as well as matching the frequent changes of the production environment, the optimization component should incorporate local and global optimization goals to be balanced coherently with the production scenario. A further feature of the optimization process concerns the integration of learning mechanisms. The learning mechanism incorporated in the optimization component should be aimed at structuring a basic knowledge layer and providing learning functions to support the analysis and interpretation of information and adjust the control behaviour over time.

Nevertheless, this design methodology requires an advanced development environment, capable of supporting the design of the control code according to international standards—for compliancy and interoperability—and of implementing runtime information exchange among different modules of the distributed control systems—to achieve global goal by means of real time collaborative strategies.

4.4 Proposed Approach

The proposed approach, named *Generic Evolutionary Control Knowledge-based mOdule* (GECKO), consists in a control infrastructure based on an interactive coop-

eration of control modules to cope with the requirements defined in the previous section. The GECKO approach enables the single devices—from end-effector, complex machine equipment up to the integrated cell and system—to evolve from stand alone, rigidly and hierarchically managed components into autonomous, self-declaring, heterarchically interacting and collaborating components. GECKO is conceived to detect and interpret the production environment features and adapt its capabilities on the basis of the specific requirements by automatically accomplishing local and global objectives. Under this perspective, opposite to pre-determined production system architecture, GECKO presents an adapting behaviour resulting from an advanced reconfiguration mechanism together with a form of intelligence and knowledge enabling the recognition of the products, events and other entities operating in the shop-floor.

GECKO is designed as a complex control architecture consisting of a number of cooperating software components:

- The *Communication* component (Sect. 4.5.1) is responsible for exchanging information and signals with the shop-floor. This module is in charge to: (i) enforce interoperability with other entities (included GECKO modules), (ii) collect data from the sensor infrastructure, (iii) manipulate data in order to assess the production environment over time and, finally, (iv) publish both the internal status and the capabilities of the entity itself.
- The *Control* component (Sect. 4.5.2) activates the control functions based on the capabilities to be accomplished by also dynamically managing events that can be both exogenous (e.g. new product features or demand volumes) and endogenous (e.g. failures of machine tools).
- The *Capability Assessment* component (Sect. 4.5.3) aims at matching the context reproduction with the GECKO nominal capabilities and determining the more suitable GECKO features and behaviours to be exploited in a specific time.
- The *Context Recognition* component (Sect. 4.5.4) reproduces an abstraction of the production context relying on an internal knowledge base elicited from the information collected by the communication component and exploiting intelligent reasoning and learning techniques in order to dynamically recognize the actual production context as well as to increase its knowledge.
- The *Optimization* component (Sect. 4.5.5) will determine the local and global optimization strategies driving the control module together with an evolutionary mechanism enabling the GECKO entity to learn and improve its functionalities over time.

These components provide functionalities that are integrated through a software infrastructure enabling the persistent internal communication. This infrastructure constitutes the physical and logic foundations to enable the joint functioning of all the GECKO modules. The production environment would consequently result in a community of GECKO entities encapsulated in the pieces of equipment that communicate, cooperate and negotiate to achieve the production goals.

4.5 Developed Methodologies and Tools

While further describing the software solutions developed for the GECKO approach, the attention will be focused on Reconfigurable Transportation Systems (RTSs) as target application, in consideration of their modular nature in implementing alternative inbound logistic systems' configurations. The transportation modules of a RTS are designed as mechatronic devices equipped with standard interfaces that allow modules to connect each other and embed logic controllers for actuators/motors and sensors. A central problem of RTSs is the Online Part Routing Problem (OPRP) addressing the formalization and synchronization of RTS mechatronic modules to transport all the parts in the manufacturing system according to their destinations [51]. A solution for the OPRP must ensure that all the parts are properly worked, the routings must be collision-free and routings must be efficient. RTS therefore represent a suitable bench-mark for proposed GECKO solution.

Since openness, interoperability and international industrial standards compliancy are fundamental requirements to be guaranteed, the hardware architecture for the deployment of GECKO control module is based on embedded industrial PC systems.

With reference to the operating system, as a foundation for the design choices, three different approaches to real-time execution support have been evaluated. Standalone RTOS, Linux plus RT Patch, and Xen-based virtual machines have been compared with respect to the following metrics: performance and determinism of execution at the operating system level, performance and determinism of real-time network communication, coexistence with non real-time applications, language and execution environment support, and communication among software modules. Linux and RT Patch have been chosen as base execution framework for the GECKO control module.

The GECKO components, as designed for the RTS case, are represented in Fig. 4.1 and will be described in the following subsections.

4.5.1 Communication Component

The communication architecture for GECKO modules has to accomplish two different tasks: data exchange with sensors/actuators distributed on the shop floor and communication between GECKO modules. The first task represents a typical feature of the networks deployed at the lowest levels of factory automation systems and is characterized by the fast and timely transmission of limited amounts of data, requiring both real-time and deterministic behaviours of the underlying networks. The second task serves to the purpose of coordinating modules activity and is typically characterized by higher traffic volume and more relaxed timing constraints than in the previous case. Considering the aforementioned traffic types, the best choice for the GECKO modules network is represented by Real-time Ethernet networks. Traffic separation, a critical feature needed to avoid interference between traffic flows

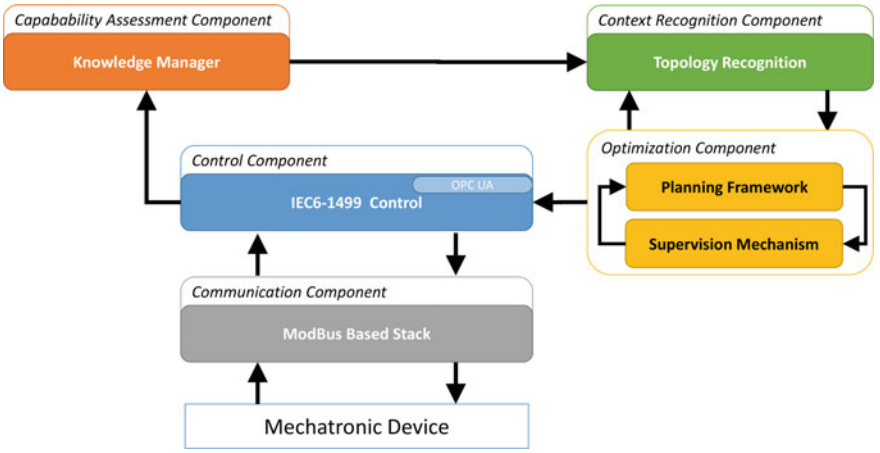

Fig. 4.1 GECKO module components

with different timing requirements, can be achieved in two ways: a single physical network, using a time-division multiple access (TDMA) method, and a physical traffic segregation that requires GECKO modules to be equipped with two network interfaces.

With reference to the communication protocol to access field I/O, Modbus was chosen, since the end-to-end round-trip delays proved to be compatible with execution times.

The implementation of the prototype (see Fig. 4.2) targeted the microcontrollers typical of low-cost embedded and industrial applications, namely the NXP LPC1768 and LPC2468, and required the configuration and integration of existing code[5] (light grey blocks in Fig. 4.2), specifically (i) a custom-built Modbus TCP node, leveraging the FreeRTOS RTOS; (ii) the lwIP TCP/IP protocol stack; (iii) an open-source Modbus TCP protocol stack (for slave nodes) and a commercial Modbus TCP protocol stack (for masters). Moreover, it was necessary to develop dedicated modules of code from scratch (white blocks in Fig. 4.2), better described in the following:

- Two different lwIP-specific adaptation layers between the network protocol stack interfaces provided by the open-source and commercial Modbus TCP libraries and the sockets-based lwIP interface.
- A suite of test programs to collect experimental data using the Modbus TCP protocol stacks on the master and slave sides.

[5] https://www.freemodbus.org/.

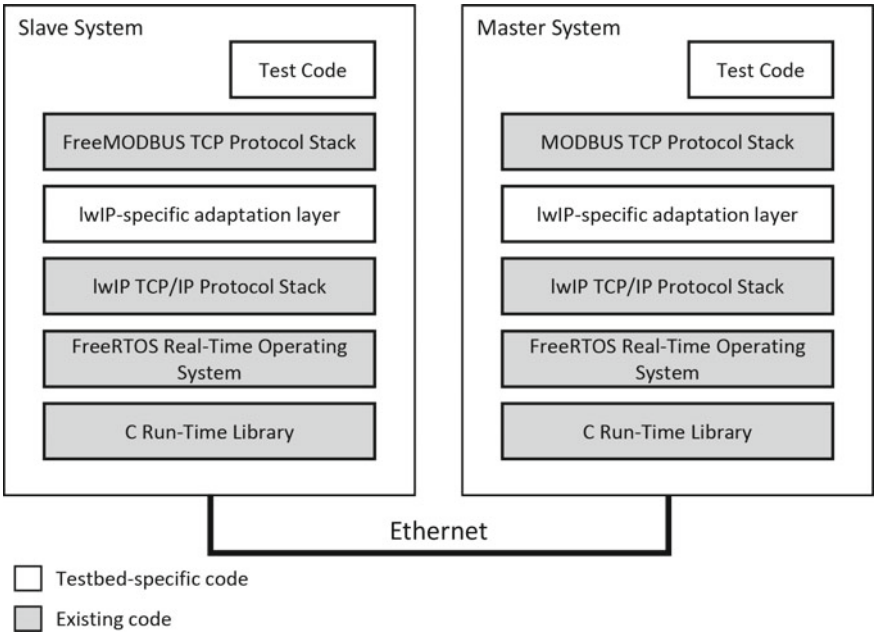

Fig. 4.2 GECKO communication and software infrastructure prototype

4.5.2 Control Component

IEC 61499 modelling standard has been adopted to develop a modular control system. According to the standard, the function block (FB) represents a unit of control software that is associated with a hardware component in the controlled system. It is therefore aimed at defining the control logic for each basic functional component of the system, meant as a self-contained module. Each function block exposes: (i) an event type input request corresponding to each task the module is capable of performing (i.e. automation function); (ii) a data type input for each configuration parameter related to the execution of module tasks; (iii) an event type output to acknowledge the execution of each module task (including un-nominal and failure conditions) and to request task execution towards downstream modules; (iv) a data type output for each variable representing the module internal state and tasks execution [52]. The execution of the control logic is regulated by the Execution Control Chart (ECC): ECC is basically a Moore automaton, consisting of states, event-conditioned transitions and actions, as well as a set of algorithms, associated with the ECC states, to be executed during specific operating conditions.

The development of IEC 61499 compliant control application takes place into a softPLC environment, named IsaGRAF,[6] natively supporting the IEC 61499 stan-

[6] http://www.isagraf.com/index.htm.

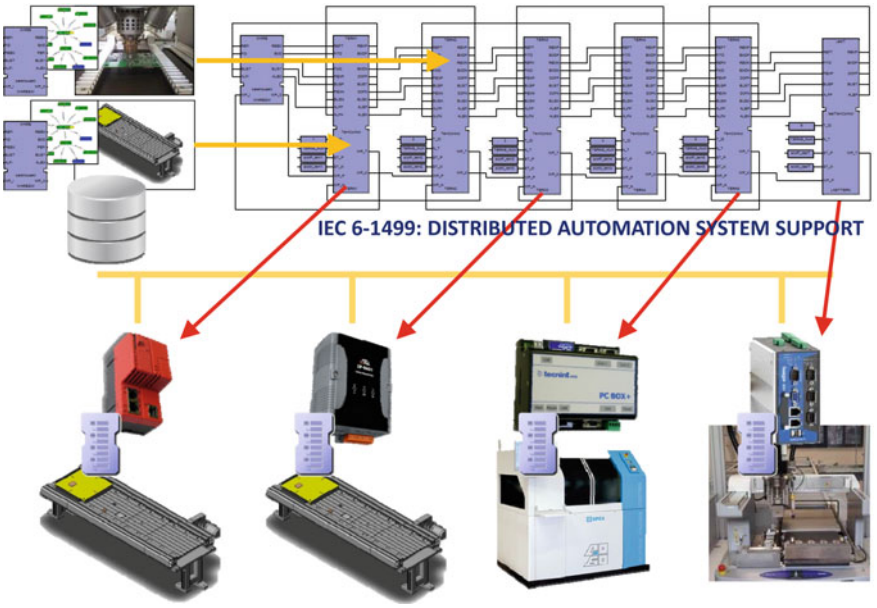

Fig. 4.3 IEC61499 based distributed control of an electronic component remanufacturing line consisting of conveyors and specialised machines

dard. As a result, a structured modular design of the overall system can be addressed through composition, i.e. by aggregating simpler mechatronic devices in a recursive way, from the level of *atomic* devices, such as sensors/actuators, till more complex equipment. In fact, by IEC 61499 reference model, a complex control application is defined by connecting FBs, as well as encapsulating basic function blocks into aggregated (composite) function blocks.

Figure 4.3 shows the distributed control engineering framework schema based on IEC 61499.

In terms of internal connections with other GECKO components, the control component includes an OPC-UA server, linked to the IEC 61499 run-time system, for exposing (as software services) the capabilities provided by the controller, thus supporting dynamic discovery, assessment, configuration and interoperability. The address space nodes are structured according to an object oriented approach based on OPC-UA. A remarkable element to be mentioned is that the IEC 61499 modelling of the control code results in a consequent identification and *clustering* of data/parameters/variable as inputs/outputs of the FB, being therefore propaedeutic to their handling according to object oriented data exchange standards. Furthermore, function driven control development is intuitively oriented and absolutely prone to the integration with Service Oriented Computing, therefore resulting in a promising hybrid paradigm.

4.5.3 Capability Assessment Component

The large amount of information linked to the capabilities exposed by various control components as well as to the technological process has led to the creation of the so called Knowledge Manager, a dedicated component capable of managing and assessing such information. The Knowledge Manager is based on a suited ontology that models the general knowledge of manufacturing environments, classifying relevant information in three distinct contexts (i.e. Global, Local and Internal) and considering a Taxonomy of Functions which classifies the set of functions the GECKO modules can perform according to their effects in the environment [53]. The Knowledge Manager exploits the ontology to build and manage a Knowledge Base (KB) that represents an abstract description of the structure and the capabilities of the GECKO modules.

In terms of implementation, the Web Ontology Language (OWL), a well-known technology for semantic data modelling, has been exploited to represent the KB. The ontology editor Protégé [54] has been used for KB design and testing.

4.5.4 Context Recognition Component

Based on contents stored in the capability assessment component, a Rule-based Inference Engine analyses the KB information and infers the advanced functional capabilities the overall control module is actually able to perform by composing the atomic capabilities exposed by the control component. The result of such an inference is a *planning* model generated from the KB to be used by the optimization component for actually control the behaviour of the mechatronic device [55]. The planning model contains an abstraction of the device, the environment's parameters in which it operates, and all the relevant constraints necessary to guarantee physical consistency. In the specific case of RTS, the context to be recognised is the topology of the plant, which shall be maintained and/or updated dependently on reconfigurations and/or failure of single modules. Thanks to the information exchanged with other neighbouring modules, every module dynamically re-builds (and constantly keeps updated) a local map of the shop-floor topology. Each module acquires a complete layout of the RTS including the connections of all the modules [56].

In terms of implementation, the Knowledge Processing Mechanism relies on Ontology and RDF APIs and Inference API provided by the Apache Jena Software Library.[7]

[7] http://jena.apache.org.

4.5.5 Optimization Component

The optimization component determines the global optimization strategies as response to the plant specific optimization problem, therefore driving the control module in its evolution.

In the context of RTS, the routing problem—as the main problem to be solved—is addressed within the optimization component as a multi-agent problem where a set of agents interact and share information in order to transport pallets (or parts) to their final destination within a RTS. Specifically, a distributed auction-based algorithm for part routing has been developed that coordinates the transport modules [51]. Such algorithm, denominated Planning Framework, establishes the assignment (i.e. routing) of the parts to the most suitable agent (i.e. module) as a result of a negotiation process by executing an auction-based mechanism with an associated multi-objective function [56].

Further to this, a Supervision mechanism continuously checks the achievement of each task via a flexible and dynamic execution monitoring system. This system analyses the pallet routings to detect possible redundancies and deadlock situations. If necessary the Planning Framework is re-executed and the path updated. The same happens in case of unforeseen events, e.g. a RTS module failure [56].

The part routing problem has been implemented in Java SE 7 by developing a multi-agent framework within the JADE platform. Both Planning Framework and Supervision mechanisms use timeline-based technology and were framed by means of the EPSL architecture [57].

4.6 Testing and Validation of Results

4.6.1 Industrial Case

The De-Manufacturing Pilot Plant sited in the lab of CNR-STIIMA (ex CNR-ITIA) [58] is dedicated to the integrated End-Of-Life processing of mechatronic products (specifically, printed circuit board, or PCB) using modular technological solutions to process heterogeneous work pieces (i.e. the PCBs) while requiring limited hardware and software reconfigurations. The first goal of the plant is to repair the PCB via remanufacturing and, if this is not suitable or technically feasible, by recovering valuable components through disassembly. Eventually, the last option is to recover the raw material by shredding and separation.

The PCB enters the De-Manufacturing Pilot Plant by means of a robotized station and then it is mounted on a pallet and loaded on an automatic conveyor. Such conveyor transports PCBs to the workstations (see Fig. 4.4) that can execute the main operations: PCB circuit analysis (*Test* station) to identify possible failed components on the board; PCB rework (*Rework* station) to replace failed components; PCB or components that cannot be repaired for technical reason (e.g. overall circuit damage),

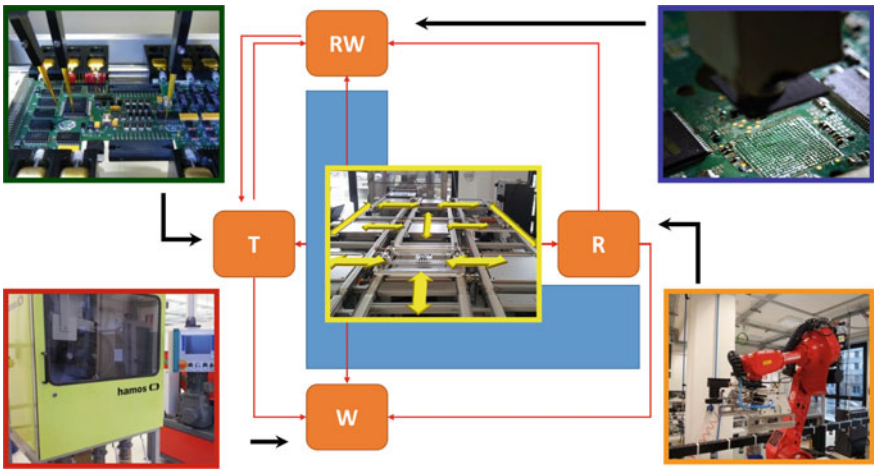

Fig. 4.4 De-manufacturing process architecture: *RW* Rework, *T* Circuit Test, *R* Robotic disassembly, *W* material recovery cell

or which economic value does not justify remanufacturing are sent to a further cell (*Disassembly* station), and possibly to a shredding cell (*Material Recovery* station) aimed at disassembling the PCB and recovering precious raw materials, respectively.

If the PCB entering the system has an economic value justifying remanufacturing, then it is first sent to the PCB analysis process for identifying eventual corrupted components or board damages. If no unrecoverable damage is identified, the PCB is sent back to the first robot station and exits the system as reusable product. If the analysis identifies a not repairable failure on the board, then the PCB is sent to the material recovery cell. If the analysis identifies a failure on one (or more) components, then the PCB is sent to the rework process for substituting the components before undergoing again the PCB analysis for checking conditions. If the test is passed, then the PCB is moved to the first station, otherwise the loop is repeated (with a configurable number of iterations). As soon as the maximum number of iterations is reached, the PCB is sent to the material recovery process. Before such stage, if the PCB mounts valuable components, it is sent to the rework station for chips desoldering. If the PCB has not an economic value justifying remanufacturing, it is moved directly to the material recovery cell. In case some components of the PCB are valuable for reuse, or have to be removed before shredding for safety reason (i.e. hazardous material content), they are sent to the rework station to disassemble such components before shredding.

The overall conveyor system is composed of a set of mechatronic modules to provide the maximum flexibility and scalability. All modules have the same dimensions, are composed of three intermediate positions for pallet hosting, each of them supporting two main (straight) transfer services and in some cases cross module transfer services. From a mechatronic point of view, the main sensing and actuating elements

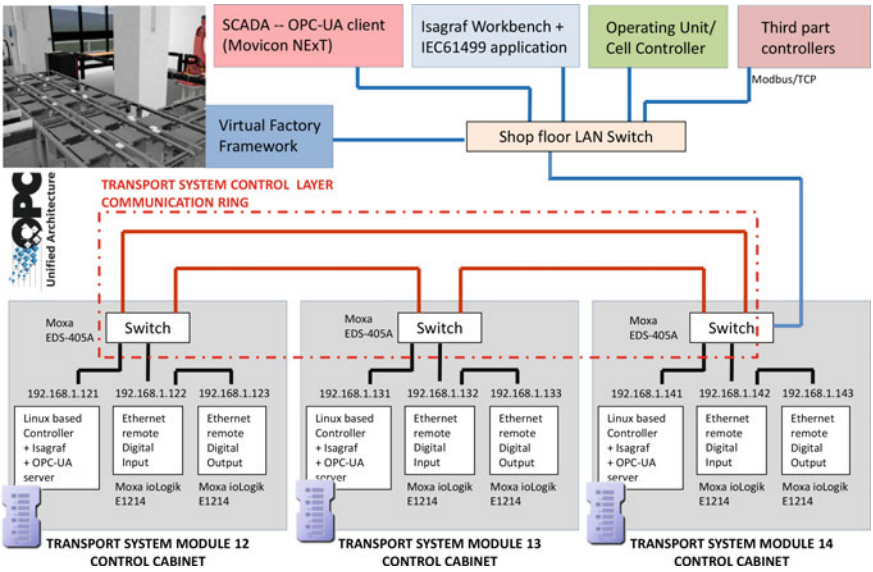

Fig. 4.5 RTS transportation module—Implemented control architecture

for each of the intermediate positions (and consequently for the whole module) are two-way motors—powering the belts for pallet transferring—indexing pins for pallet sequencing and proximity sensor for pallet detection. Each mechatronic module is mechanically, electronically and pneumatically self-contained, and designed so that connectors from one module to the neighbouring carry both power/air supply and Ethernet. Therefore, it is possible to dynamically design and deploy several configurations, depending on the particular needs of the transportation system and/or on the possible online events to occur during the production (e.g. integration of new machines in the line). As an example, a configuration can support forward and backward movements and one or more specific positions with right and left cross transfer capabilities.

4.6.2 Experiments, Results and Analysis

The GECKO control architecture has been validated within the De-Manufacturing plant described in Sect. 4.6.1 (hereinafter also referred to as *GECKO pilot plant*), by realizing a novel automation system for the RTS transportation modules. Following the GECKO approach, the control solution governing each module of the conveyor system has been re-designed to expose the dedicated services. Figure 4.5 summarizes the implemented control architecture for the RTS transportation modules of the de-manufacturing plant.

Since the most critical timings of a control system often concern I/O operations, the first tests have been performed by evaluating the real-time communication performance of the runtime by means of a custom Modbus TCP slave, implemented as described in Sect. 4.5. The tests have been carried out considering different increasing computational loads designed to reproduce the characteristics of the other software components expected to run on the system in the real plant. Moreover, existing knowledge about the internals of the RT Patch extension of the Linux kernel has been leveraged to bring the system towards its worst case behaviour. During the experiments, a custom Modbus TCP slave collected traffic statistics for later analysis, which evidenced a significant performance improvement by limiting the worst-case amount of activation and communication jitter [59].

More specifically, FreeRTOS scheduling and synchronization overheads have been evaluated by considering three main aspects, all of them typical of embedded distributed applications: (i) task scheduling overhead upon semaphore synchronization; (ii) task scheduling overhead upon message queue (mailbox) synchronization; (iii) worst-case interference from the FreeRTOS timer interrupt handler.

The internal delays of lwIP have been evaluated by inserting a number of high-resolution timestamping points into the protocol stack code, whose location has been determined by code analysis, and developing test code to generate the appropriate traffic patterns to be analysed. The experiments have been carried out using a frame size of 242 bytes, that is, close to the maximum frame size of Modbus TCP. The final part of the evaluation was by far the most complex one, because it involved, first of all, the implementation of a 2-node experimental testbed consisting of a Modbus TCP master/slave pair. Overall, the main outcome was the complete breakdown of Modbus TCP communication delays into simpler components, contributed by different software modules. For what concerns jitters, the most important discovery was to identify the role played by TCP acknowledgements, also depending on whether these acknowledgments are piggybacked onto other TCP segments or transmitted on their own.

An additional outcome has been a comprehensive comparison between the code execution performance of the two microcontrollers considered in the experiments. More specifically, besides the CPU clock speed, the two main contributing factors to the different performance of the two architectures are:

- Memory bus width. On the LPC1768 the static RAM is internal to the microcontroller and is connected to the CPU through a 32-bit bus. On the LPC2468 the RAM is dynamic for the most part, is external to the microcontroller, and is connected to it by means of a 16-bit bus originating from the on-chip External Memory Controller (EMC). For word-sized transfers, the most common kind of transfer, this implies a time scaling factor of 2 when clock frequencies are assumed to be the same in both cases, because two bus cycles are needed to access the RAM on the LPC2468, versus one on the LPC1768.
- DRAM/SRAM access. The access methods of static versus dynamic RAM are very different. SRAM access is simpler, because that kind of memory can consistently provide one 32-bit word of data every clock cycle. On the contrary, DRAM access

time is variable and depends on a multitude of factors, for instance, the length and likelihood of intervening refresh cycles, the current state of the memory controller state machine, and write buffer status. Due to the difficulty of calculating this factor analytically, its value is best derived from experimental data.

The next step aimed at verifying the real-time behaviour of the ISAGRAF IEC 61499. Functional tests have been conducted on a set of modules representative of the GECKO pilot plant, with runtime component hosted on the plant PLCs, so as to establish that this latter is not adversely affected by the concurrent execution of additional, interfering tasks on the same embedded control system. The output of such tests proved the operating system to be robust enough to be able to effectively isolate the IEC 61499 runtime component from the execution timing point of view.

The GECKO distributed control approach has been then tested both on the real plant and on a set of different simulated layouts to verify the effectiveness of the approach independently from the specific plant configuration. The design of experiment aims at (i) evaluating RTS throughput and parts lead time considering different layouts, (ii) comparing the features of the proposed control approach with respect to a well-defined benchmark case and (iii) assessing the benefits of the distributed approach when temporary and dynamic changes occur in the RTS. The observing horizon was set equal to a work shift (i.e. 8 h), while the rate of parts entering the system from the load station is kept constant. A complete and exhaustive description of such experiments can be found in [51].

Figure 4.6 shows the different RTS layouts used in simulated experiments. All layouts are composed of 20 transportation modules, where each module has a main movement direction and is equipped with cross-transfer devices (represented in the picture by the bold black segments) enabling the cross-transportation between adjacent transportation modules. Machines and I/O modules are represented, respectively, by red and green blocks.

The experiments have been carried out in two phases to (i) evaluate RTS throughput and parts lead time while considering different layouts (first phase), and (ii) assess the benefits of the distributed approach when changes occur in the RTS (second phase).

The experimental runs of the first phase proved the benefits associated with the distributed auction-based approach to support the design of RTS configurations. After the evaluation of the system throughput and the part lead time for the different RTS layouts, the system designer can select the most efficient configuration according to the production and technology requirements. Considering average throughput values and part lead time in each layout, the results of the experiments are reported in Table 4.1.

Figure 4.7 shows the trends of finished parts, i.e. the number of processed parts (y-axis) sampled every 30 min (x-axis). As shown in the figure, the RTS *Cross* configuration is the best topology among the tested ones, if the average throughput and part lead time are considered.

The experimental runs of the second phase targeted the capability of the distributed control approach to match temporary or permanent changes (e.g. module failures)

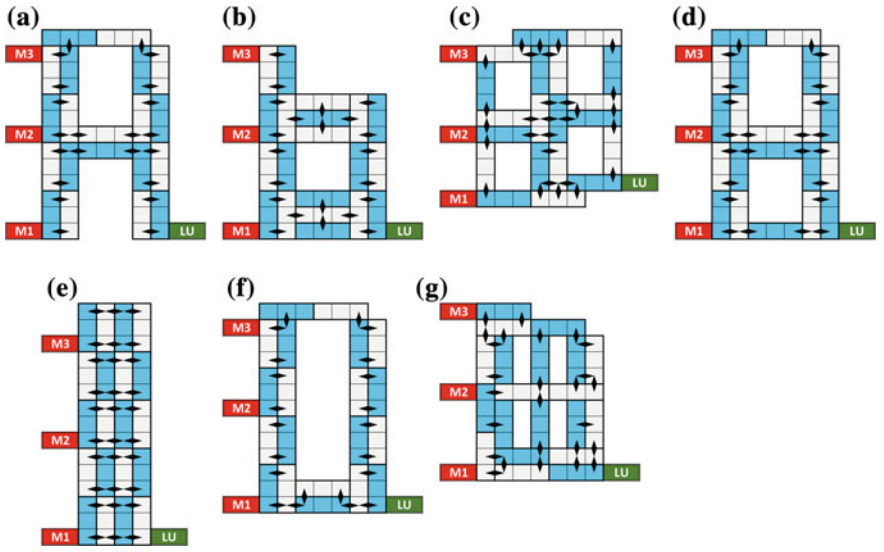

Fig. 4.6 Examples of RTS layouts considered for the experiments: **a** RTS *A*, **b** RTS *B*, **c** RTS *Cross*, **d** RTS *Eight*, **e** RTS *Full*, **f** RTS *Ring*, **g** RTS *Star*

Table 4.1 Average throughput values and part lead time of tested RTS layouts

Layout	Average throughput (part/h)	Part average lead time (s)
A	19	933
B	20	889
Cross	28	627
Eight	20	848
Full	21	840
Ring	20	840
Star	23	746

of the RTS layout. A failure leads to a temporary unavailability of the corresponding module, thus triggering the generation of a new RTS layout corresponding to the new reality of the system, and allowing to resume the production of new routing policies until the original layout is (possibly) restored. The interested reader can find further details in [51].

Additional tests have been carried out to evaluate the knowledge processing mechanism and the capability of a single GECKO agent to build a consistent abstraction of the production context and synthesize timeline-based models accordingly. Experiments have been designed to evaluate—across different configurations—the inference time needed to build the knowledge base of an agent as well as the capability of updating such a knowledge every time that a physical change (i.e. a reconfiguration of the transportation module) occurs in the module. All the different physical

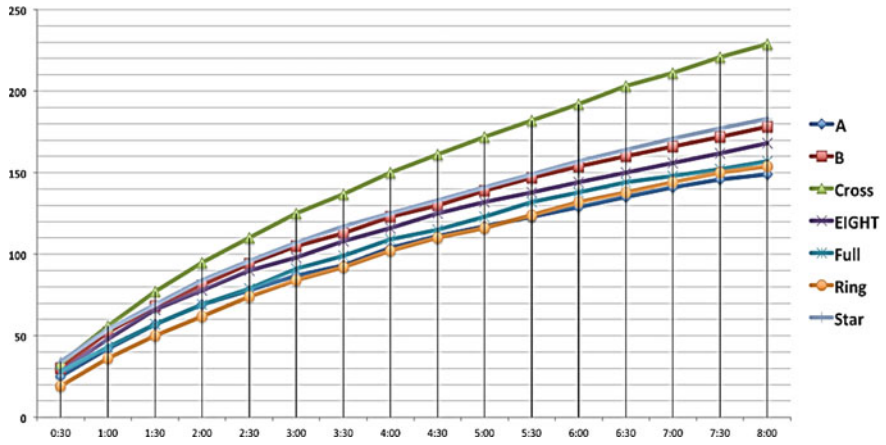

Fig. 4.7 Production volumes associated to examples of RTS layouts

configurations of a transportation module have been considered, from unidirectional module (with no cross-transfer device) to module with three cross-transfer devices. These configurations are referred to as *simple*, *single*, *double* and *full*, respectively. A further possible configuration entails a different number of connected neighbouring modules. Clearly, the more complex scenario is the superimposed one with the highest number of cross-transfers (full configuration) and neighbours. In order to pursue such complex scenario, reconfiguration scenarios have been addressed, based on occurrence on different external events, particularly an increasing number (from 1 to 3) of transportation module neighbours momentarily unable to exchange pallets, as well as on occurrence of internal failures to a cross-transfer device engine to a specific port.

Figure 4.8 shows the results concerning the time needed to infer the knowledge base as a response to the new reconfigured scenario and the time needed to generate a corresponding timeline-based model. The experimental results prove the practical feasibility of the knowledge processing mechanism in increasingly complex configurations of a transportation module. The cost of such a reasoning mechanism has low impact on the performance of a module during its operations. Further details about the inference process and the experimental results can be found in [55].

4.7 Conclusions and Future Research

The proposed GECKO multi-agent distributed control approach [59] enables autonomous agents to dynamically recognize the working settings, identify their capabilities, and collaborate to control the production and support its reconfiguration [53, 55]. The proposed solution is capable of automatically dealing with structural

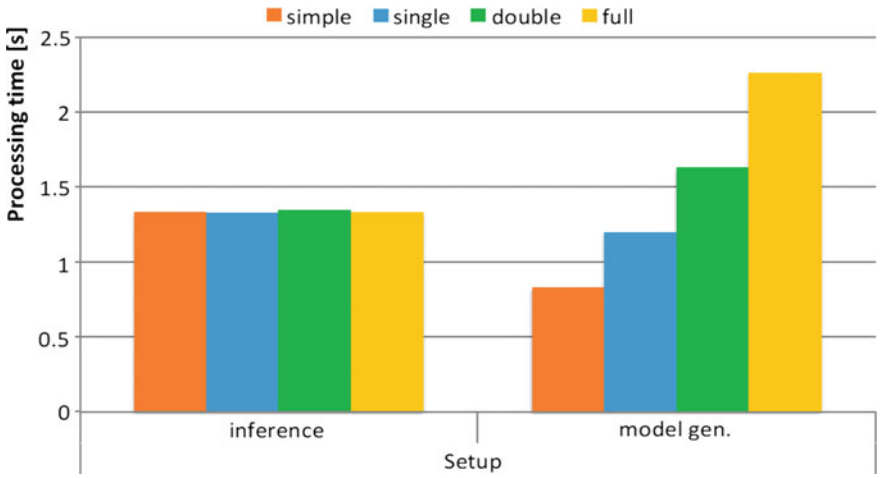

Fig. 4.8 Knowledge base inference time and timeline-based model generation time

changes in the topology of the plant as well as the capabilities of the transportation modules without affecting the production, so that it is possible to cope with changes in the demand and/or unforeseen events/failures at shop floor level. Indeed, unlike classical centralized control approaches, the GECKO distributed control approach can dynamically optimize the paths that pallets follow, by reasoning online on the actual state of the plant. It is worth highlighting that this reasoning is performed without suspending the production.

The adoption of the GECKO approach for design and development of novel solutions to automation systems has been discussed with reference to the specific case study of a de-manufacturing plant, showing how such a decentralized control approach and the integration of planning and knowledge reasoning technologies can be beneficial to improve the efficiency and the flexibility of complex systems like RTSs [56].

The integration and (re)use of the control solutions is enhanced by the encapsulation of the control functionalities into a network of interconnected function blocks. Moreover, considering the development of the overall control system, the activity of the control engineer benefits from the availability of control modules of mechatronic devices, as single components, already tested and validated.

Distributed control system methodologies and prototype tools presented in this work are further extended towards a higher TRL (Technology Readiness Level) within the framework of two Horizon 2020 projects. In particular, the Daedalus project[8] deals with the extension of the modular and distributed control system platform, orienting towards the Industry 4.0 CPS (Cyber Physical System) paradigm. Moreover, the MAYA project[9] addresses the integration of the distributed automa-

[8] http://daedalus.iec61499.eu/index.php/en/.

[9] http://www.maya-euproject.com/.

tion system interoperability framework to support dynamic fitting of factory digital models with data gathered from the real plants, so to improve the reliability of the forecasting and optimizations.

Acknowledgements This work has been funded by the Italian Ministry of Education, Universities and Research (MIUR) under the Flagship Project "Factories of the Future—Italy" (Progetto Bandiera "La Fabbrica del Futuro") [60], Sottoprogetto 1, research project "Knowledge based modules for adaptive distributed control systems" (GECKO). The authors greatly acknowledge the industrial partners of the GECKO project, namely FF Falconi Snc, Magneti Marelli Spa and Tecnint HTE for the technical support to the conception of requirements, development ideas and validation scenarios. Authors from ISTC-CNR would like to thank their colleagues Stefano Borgo and Alessandro Umbrico for their valuable work during the project.

References

1. Barenji RV, Barenji AV, Hashemipour M (2014) A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-013-5597-2
2. Heilala J, Voho P (2001) Modular reconfigurable flexible final assembly systems. Assem Autom 21(1). https://doi.org/10.1108/01445150110381646
3. Project SMErobot (2006) The European robot initiative for strengthening the competitiveness of SMEs in manufacturing. EU FP6 2006
4. Project COMET (2010) Plug-and-produce COmponents and METhods for adaptive control of industrial robots enabling cost effective, high precision manufacturing in factories of the future. EU FP7 2010
5. Project DYNEXPERTS (2010) Plug and produce components for optimum dynamic performance manufacturing systems. EU FP7 2010
6. Zimmermann U, Bischoff R, Grunwald G, Plank G, Reintsema D (2008) Communication, configuration, application—the three layer concept for plug-and-produce. In: Proceedings of the fifth international conference on informatics in control, automation and robotics, Funchal, Madeira, 11–15 May 2008. INSTICC Press, Setúbal, pp 255–262
7. Reinhart G, Krug S, Hüttner S, Mari Z, Riedelbauch F, Schlögel M (2010) Automatic configuration (plug & produce) of industrial ethernet networks. In: 9th IEEE/IAS international conference on industry applications, 8–10 Nov, Sao Paulo, Brazil
8. Guarino N (1998) Formal ontology and information systems. FAIA 46, IOS Press, pp 3–15
9. Borgo S, Leitao P (2007) Foundations for a core ontology of manufacturing. In: Sharman R, Kishore R, Ramesh R (eds) Ontologies: a handbook of principles, concepts and applications in information systems, volume 14 of Integrated series in information systems. Springer, pp 751–776
10. Kuijpers EA, Carotenuto L, Cornier C, Damen D, Grimbach A (2010) The Ulisse environment for collaboration on ISS experiment data and knowledge representation. In: Proceedings of 61st international astronautical congress (IAC'10), Prague, CZ
11. Kuijpers EA, Carotenuto L, Malapert JC, Markov-Vetter D, Melatti I, Orlandini A, Pinchuk R (2012) Collaboration on ISS experiment data and knowledge representation. In: Proceedings of 63rd international astronautical congress (IAC'12), Naples, Italy
12. Koren Y (1999) Reconfigurable machine tools. Patent number: 5943750
13. Koren Y, Heisel U, Jovane F, Moriwaki T, Pritschow G, Ulsoy G, Van Brussel H (1999) Reconfigurable manufacturing systems. CIRP Ann Manuf Technol 48(2):527–540
14. Terkaj W, Tolio T, Valente A (2009) Focused flexibility in production systems. In: ElMaraghy HA (ed) Changeable and reconfigurable manufacturing systems. Springer, pp 47–66

15. Bloch H, Fay A, Hoernicke M (2016) Analysis of service-oriented architecture approaches suitable for modular process automation. In: 2016 IEEE 21st international conference on emerging technologies and factory automation (ETFA), Berlin, pp 1–8
16. Henneke D, Elattar M, Jasperneite J (2015) Communication patterns for cyber-physical systems. In: 20 th IEEE international conference on emerging technologies and factory automation
17. MacKenzie CM, Laskey K, McCabe F, Brown PF, Metz R (2015) Reference model for service oriented architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html. Accessed 15 Oct 2015
18. European IST FP6 project, SOCRADES—service-oriented cross-layer infrastructure for distributed smart embedded devices. www.socrades.net
19. Cannata A, Gerosa M, Taisch M (2008) SOCRADES: a framework for developing intelligent systems in manufacturing. In: 2008 IEEE international conference on industrial engineering and engineering management (IEEM), pp 1904–1908
20. Mendes JM, Leitao P, Colombo AW, Restivo F (2008) Service oriented process control using High-Level Petri Nets. In: 2008 6th IEEE international conference on industrial informatics (INDIN), pp 750–755
21. Karnouskos S, Colombo AW, Bangemann T, Manninen K, Camp R, Tilly M, Sikora M, Jammes F, Delsing J, Eliasson J, Nappey P, Hu J, Graf M (2014) The IMC-AESOP architecture for cloud-based industrial cyber-physical systems. In: Colombo AW, Bangemann T, Karnouskos S, Delsing J, Stluka P, Harrison R, Jammes F, Lastra JLM (eds) Industrial cloud-based cyber-physical systems: the IMC-AESOP approach. Springer International Publishing, pp 49–88
22. Procedure Automation for Continuous Process Operations, ISA-TR 106.00.01
23. Dai WW, Christensen JH, Vyatkin V, Dubinin V (2014) Function block implementation of service oriented architecture: case study. In: 12th IEEE international conference on industrial informatics (INDIN), pp 112–117
24. Function blocks—Part 1: Architecture, IEC 61499-1 (2012)
25. Lotter B, Wiendahl H-P (2009) Changeable and reconfigurable assembly systems. In: Elmaraghy H (ed) Changeable and reconfigurable manufacturing systems. Springer, London, pp 127–142
26. Nonaka Y, Erdıs G, Kis T, Nakano T, Váncza J (2012) Scheduling with alternative routings in CNC workshops. CIRP Ann Manuf Technol 61:449–454
27. Azab A, ElMaraghy HA (2007) Mathematical modeling for reconfigurable process planning. CIRP Ann Manuf Technol 56:467–472
28. Nyhuis P, von Cieminski G, Fischer A (2005) Applying simulation and analytical models for logistic performance prediction. Ann CIRP 54:417–420
29. Tolio T, Ceglarek D, ElMaraghy HA, Fischer A, Hu SJ, Laperriere L, Newman ST, Vancza J (2010) SPECIES—co-evolution of products, processes and production systems. CIRP Ann Manuf Technol 59(2):672–693
30. Hu J, Camelio J (2006) Modeling and control of compliant assembly systems. CIRP Ann Manuf Technol 55:19–22
31. Terkaj W, Tolio T, Valente A (2009) Design of focused flexibility manufacturing systems (FFMSs). In: Tolio T (ed) Design of flexible production systems—methodologies and tools. Berlin, Heidelberg, pp 1–18
32. Quiao, B, Zhu J (2000) Agent-based intelligent manufacturing system for the 21st century. In: Proceedings of the international forum for graduates and young re-searchers of EYPO, The World Exposition in Germany, Hannover
33. Ferber J (1999) Multi-agent systems: an introduction to distributed artificial intelligence, vol 1. Addison-Wesley, Reading
34. Heragu S, Graves R, Kim B et al (2002) Intelligent agent-based framework for manufacturing systems control. IEEE Trans Syst Man Cybern Part A Syst Hum 32(5):560–573. https://doi.org/10.1109/TSMCA.2002.804788
35. Crawford LS, Do MB, Ruml W, Hindi H, Eldershaw C, Zhou R, Kuhn L, Fromherz MP, Biegelsen D, de Kleer J et al (2013) Online reconfigurable machines. AI Mag 34(3):73–88

36. Balakirsky S (2015) Ontology based action planning and verification for agile manufacturing. Robot Comput Integr Manuf 33(0):21–28. Special Issue on Knowledge Driven Robotics and Manufacturing
37. Solano L, Rosado P, Romero F (2013) Knowledge representation for product and processes development planning in collaborative environments. Int J Comput Integr Manuf 27(8):787–801
38. Terkaj W, Schneider GF, Pauwels P (2017) Reusing domain ontologies in linked building data: the case of building automation and control. In: Proceedings of the 8th workshop formal ontologies meet industry, joint ontology workshops 2017, CEUR workshop proceedings, vol 2050
39. Hristoskova A, Aguero E C, Veloso M, De Turck F (2013) Heterogeneous context-aware robots providing a personalized building tour. Int J Adv Robot Syst
40. Suh I H, Lim G H, Hwang W, Suh H, Choi J H, Park Y T (2007) Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence. In: IEEE/RSJ international conference on intelligent robots and systems. IROS 2007, pp 429–436
41. Tenorth M, Beetz M (2015) Representations for robot knowledge in the KnowRob framework. Artif Intell. http://dx.doi.org/10.1016/j.artint.2015.05.010
42. Lemaignan S, Ros R, Mosenlechner L, Alami R, Beetz M (2010) ORO, a knowledge management platform for cognitive architectures in robotics. In: 2010 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 3548–3553
43. Hartanto R, Hertzberg J (2008) Fusing DL reasoning with HTN planning. In: Dengel A, Berns K, Breuel T, Bomarius F, Roth-Berghofer T (eds) KI 2008 advances in artificial intelligence, volume 5243 of Lecture notes in computer science. Springer, pp 62–69
44. Yang F, Khandelwal P, Leonetti M, Stone P (2014) Planning in answer set programming while learning action costs for mobile robots. In: AAAI Spring 2014 symposium on knowledge representation and reasoning in robotics (AAAI-SSS)
45. Terkaj W, Urgo M, Andolfatto D (2017) Answer set programming for modeling and reasoning on modular and reconfigurable transportation systems. In: Proceedings of the 2017 federated conference on computer science and information systems
46. IEEE, IEEE Std 1003.13 (2003) Edition, standard for information technology—standardized application environment profile (AEP)—POSIX realtime and embedded application support
47. Barry R (2010) Using the FreeRTOS real time kernel. Raleigh, North Carolina
48. Decotignie JD (2005) Ethernet-based real-time and industrial communications. Proc IEEE 93(6). https://doi.org/10.1109/jproc.2005.849721
49. Dzung D, Naedele M, Von Hoff TP, Crevatin M (2005) Security for industrial communication systems. Proc IEEE 93(6). https://doi.org/10.1109/jproc.2005.849714
50. Moyne JR, Tilbury DM (2007) The emergence of industrial control networks for manufacturing control, diagnostics, and safety data. Proc IEEE 95(1). https://doi.org/10.1109/jproc.2006.887325
51. Carpanzano E, Cesta A, Orlandini A, Rasconi R, Suriano M, Umbrico A, Valente A (2016) Design and Implementation of a distributed part-routing algorithm for re-configurable transportation systems. Int J Comput Integr Manuf 29(12):1317–1334. https://doi.org/10.1080/0951192X.2015.1067911
52. Brusaferri A, Ballarino A, Cavadini FA, Manzocchi D, Mazzolini M (2014) CPS-based hierarchical and self-similar automation architecture for the control and verification of reconfigurable manufacturing systems. In: Proceedings of the 2014 IEEE emerging technology and factory automation (ETFA), Barcelona, pp 1–8
53. Borgo S, Cesta A, Orlandini A, Umbrico A (2015) An ontology-based domain representation for plan-based controllers in a reconfigurable manufacturing system. In: Proceedings of 28th Florida artificial intelligence research society conference (FLAIRS-15)
54. Musen MA (2011) The protégé project: a look back and a look forward. AI Matters 1(4):4–12
55. Borgo S, Cesta A, Orlandini A, Umbrico A (2016) A planning-based architecture for a reconfigurable manufacturing system. In: Proceedings of 26th international conference on automated planning and scheduling

56. Carpanzano E, Cesta A, Orlandini A, Rasconi R, Valente A (2014) Intelligent dynamic part routing policies in Plug&Produce reconfigurable transportation systems. CIRP Ann Manuf Technol 63(1):425–428
57. Umbrico A, Orlandini A, Cialdea Mayer M (2015) Enriching a temporal planner with resources and a hierarchy-based heuristic. In: Gavanelli M, Lamma E, Riguzzi F (eds) AI*IA 2015 advances in artificial intelligence. Lecture notes in computer science, vol 9336. Springer, Cham
58. Tolio T, Copani G, Terkaj W (2019) Key research priorities for factories of the future—Part II: Pilot plants and funding mechanisms. In: Tolio T, Copani G, Terkaj W (eds) Factories of the future. Springer
59. Borgo S, Cesta A, Orlandini A, Rasconi R, Suriano M, Umbrico A (2014) Towards a cooperative knowledge-based control architecture for a reconfigurable manufacturing plant. In: Proceedings of 19th IEEE international conference on emerging technologies and factory automation (ETFA). IEEE
60. Terkaj W, Tolio T (2019) The Italian flagship project: factories of the future. In: Tolio T, Copani G, Terkaj W (eds) Factories of the future. Springer