Heterogeneous Communication Middleware for Digital Twin Based Cyber Manufacturing Systems

Pascal Andre, Fawzi Azzi, Olivier Cardin

Heterogeneous Communication Middleware for Digital Twin based Cyber Manufacturing Systems

Pascal André, Fawzi Azzi, and Olivier Cardin

Abstract Communication systems and network are a backbone for manufacturing systems and IoT in Industry 4.0. In this context, the control software systems are made of scalable services that exchange through communication middlewares. Among them, the definition of the Digital Twin exhibits even more needs for heterogeneous communication, due to the various nature of the devices and their providers but also to legacy applications. The maintenance of such systems becomes tricky when the communication statements are widespreadly merged in the application code. In this paper, we investigate the separation of the communication concerns from other aspects of the application in order to improve the system evolution and make it adaptable and reconfigurable to different contexts (resources, workshop...). Various approaches are compared and a solution is exhibited with a first prototype.

Key words: Holonic Manufacturing Systems, Service, Communication, Digital twin, Heterogeneous protocols

1 Introduction

Since the event of Internet and web, communications become a crucial issue in automated systems. Communication systems and network are a backbone for manufacturing systems and IoT in Industry 4.0 and the advent of Edge computing. In [4], Cisco forecast the global IP traffic will more than triple to 3.3 zettabytes by 2021 which will be mainly machine to machine (M2M) communications. Consequently

Pascal André and Fawzi Azzi
LUNAM Université, Université de Nantes, LS2N UMR CNRS 6004 2, rue de la Houssinière F-44322 Nantes Cedex, France, e-mail: pascal.andre@univ-nantes.fr

Cardin, Olivier
LUNAM Université, IUT de Nantes – Université de Nantes, LS2N UMR CNRS 6004 2 avenue du Prof. Jean Rouxel – 44475 Carquefou Cedex, France, e-mail: olivier.cardin@univ-nantes.fr

system's interoperability remains a fundamental and still challenging quality of software and hardware systems. The current technologies are based on service oriented architectures (SOA) because the service paradigm is scalable from low level message communications to business processes interactions. The manufacturing software systems, and especially the digital twins, are based on distributed scalable services that exchange through communication middlewares which can be Remote Procedure Call (RPC) based solutions, Object Request Brokers ORB (ORB), Message Oriented Middlewares (MOM), the Enterprise Application Integration (EAI) framework or the recent Enterprise Service Bus (ESB) solution. Using one of the these middleware approaches provides adequate solutions for distributed systems especially in the case where we develop new manufacturing systems.

Due to the variety of the devices and their providers but also to legacy applications, the communication means are heterogeneous and one technology usually cannot suffice[1]. The *(software) maintenance of manufacturing systems* introduce perturbation when replacing failed or broken devices or resources by new ones (sometimes delivered by new providers), when facing technical debt of legacy application (changing controllers technology)...This evolution happens all along the system life, may introduce heterogeneity that impacts the communication middleware.

This maintenance becomes tricky when the communication statements are widespreadly merged in the application code and lead to hard refactoring tasks. In this paper, we investigate the separation of the communication concerns from other aspects of the application in order to improve the system evolution and make it adaptable and reconfigurable to different contexts (resources, workshop...). Various approaches are compared and a solution is exhibited with a first prototype. This work is a new contribution to a previous work [7] that focus on the communication aspect.

We detail the problem statement in section 2, and the domain of heterogeneous communication abstraction in section 3. In section 3, we propose a framework to serve as a communication layer between the manufacturing control system and the controlled workshop components including digital twins and simulation. Section 4 illustrates the *Sofal* product line approach on a communication part of a control system. A first prototype is presented in section 4. Applicability in the domain is discussed in section 5. In conclusion, we draw perspectives for manufacturing.

2 Background and Problem Statement

We assume the reader to be familiar with holonic manufacturing systems. If not we suggest to read the following references [11, 25, 11, 9] We also assume a *Service-based Component (SbC)* model [1] where a functionality is implemented by services provided by components. Provided services are not necessarily atomic calls and may have a complex behaviour in which other services might be needed (called) and messages can be exchanged. The problem statement issues from both literature review and practice from case studies.

[1] Note that heterogeneity is the main motivation of EAI but it is a framework not a technology.

Figure 1 exhibits two examples of holonic based control implementation in manufacturing: case a) is related to the holonic control in tyre manufacturing industry and exhibits the connection with the virtual counterpart of the actual system [14], whereas case b) deals with the control of a flexible manufacturing system [10] with a Service-oriented Holonic Manufacturing System (SoHMS). In those architectures, we can spot that the main represented elements deals with the communications between the various elements. Indeed, even if this is usually considered as a "technological" issue, communication is an essential stake for distributed software.

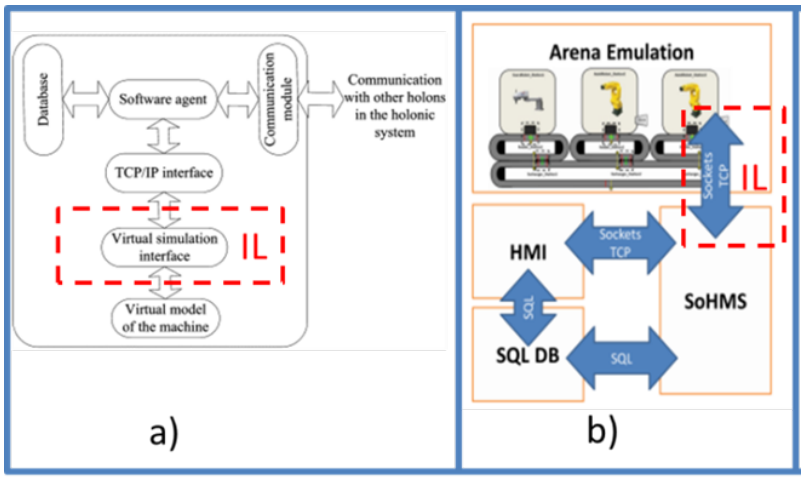

Fig. 1 Communication and interface layers in literature [14, 10]

In such architectures, three classes of communication can be differentiated:

Intra-holons communication When holons internally run on separate threads or processes that need to exchange information all along their life-cycle, it is necessary to standardize the way the communication is set up. This is the case for example when a holon is responsible the whole management of a resource. It needs to be able to both negotiate with other holons to establish the planning of this resource, and at the same time trigger and monitor the activity of the resource itself following this planning. Obviously, these tasks are not operated on the same time horizon, and are therefore executed in parallel. Solutions for this communication is often based on tables of shared variables or by queues and messages [16]. This communication mode is out-of-scope of this article.

Inter-holons communication Even if holons have independent behaviours, they are intended to collaborate and negotiate to reach the goals of the holarchy. Communication between holons is thus of particular importance in the software design. Because of the inherent asynchronous nature of the communications between holons, mes-

sage passing methodology, *i.e.* a methodology based on messages exchange between the entities of the architecture, is the most used.

To be more specific, multi-agent systems being the most used tool to implement HMS, the communication between holons is often based on the agent communication methodologies and protocols used in MAS, *i.e.* message including message or plan passing, information exchanges through a shared data repository, or high-level communication [21]. Many protocols were already developed, including the contract net protocol (CNP), probably the most well-known [23]: CNP provides a lot of basics for most of the communication needs of such control systems, including message announcement, message receiving management, bidding mechanism, contract awarding and contract execution [17], in coherence with the FIPA ACL definition [5].

This communication mode is usually used inside the control system in itself. However, this trend is meant to evolve with the evolution of embedded holons [24], where some holons could be physically separated from the control systems. In this kind of architecture, the need for supporting a multi-hardware and there-fore multi-protocol connection to various devices embedding the holons is important.

External communication The control systems is obviously also communicating with some other applications, including for example a user-machine interface, a database, some PLC, CNC, etc. This communication implies multiple protocols due to the software or hardware limitations. The list is obviously long, but it can be organized in three classes:

- Web or IoT-oriented protocols, such as UDP, TCP, MQTT or HTTP [20];
- Industrial local network protocols, e.g. Modbus TCP or Profibus [12];
- Messaging protocol standards OPC DA [6], OPC-UA or SOAP [19];
- TCP sockets [2, 22] or OTP [15].

The probability to have multiple protocols in the same application is quite high, especially in the objective of plug-and-produce [18] or reconfigurable manufacturing systems [3] where various equipment and interfaces are plugged in and out all along the life-cycle of the control system. When integrating a Digital Twin in the architecture, this feature is quite mandatory, as the functionalities are split into several standalone middlewares enabling communications with multiple pieces of hardware and software.

3 An Heterogeneous Communications Handler

The previous section exhibit the need for a Multi-Protocol Communication Tool (MPCT) that would handle the communication in the interfaces of the HMS, either towards external applications, or even towards holons embedded in physical resources if the HMS is distributed on several physical assets. In this section we overview the different solutions and motivate a pluggable and adaptable solution that fits to the kind of problem we tackle.

Hohpe and Woolf propose a catalog of integration patterns to be used at the architectural or design level [13]. They consider four cases of integration styles: File transfer, Shared database, Remote procedure invocation and Messaging. Only the last style can apply to distributed systems with autonomous communicating *components* (*e.g.* application, agents). that have their own data and communicate through high level services (see assumptions in section 2).

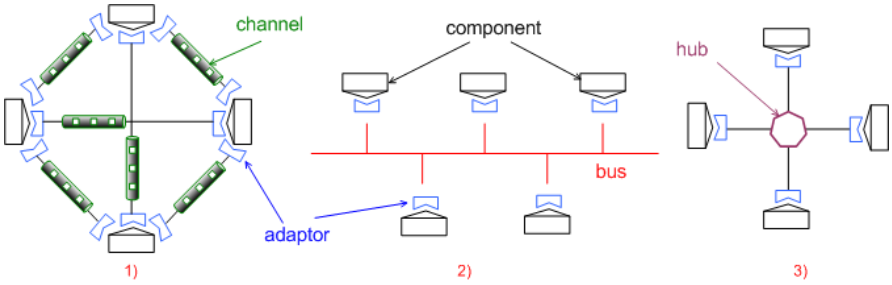

Fig. 2 Message Communication Architecture

Figure 2 shows three main architectural patterns to represent the topology of the message network in distributed systems with adaptors (or connectors) to handle heterogeneous communications.

1. **Point to point Architecture** (or messaging backbone) A message channel is built for each couple of components. This pattern supports heterogeneous (customized) communication protocols that can be synchronous or asynchronous. But it is hardly evolvable when the number of components increases.
2. **Bus Architecture** A bus convey the messages between the components, only the target components read the messages. Examples are Object Request Brokers ORB (*e.g.* CORBA, .NET) or Message Oriented Middleware MOM (*e.g.* AMQP, MQTT). Such protocols are scalable and handle efficiently broadcast but less heterogeneity. The messaging system of the bus is the pivot language for adaptation.
3. **Centralised Architecture** or *hub-and-spoke* A mediator component play the role of mediator (or federation) between the system components. Enterprise Application Integration (EAI) use the hub-and-spoke or bus architectures. Each link to the hub is a kind of point-to-point

Solution 2 is really interesting when the components play the same role with compatible protocols while solution 1 and 3 support heterogeneity. In practice, a mixed solution offer the good compromises for HMS based control systems.

Figure 3 illustrates the possible positioning of MPCT in those two cases. As the tool is meant to be able to handle multi-point communication using various standards, it can be used for all the interfaces of the HMS. As a matter of fact, it is required to integrate automatic configuration mechanisms of the tool in order to enable plug-and-produce mechanisms and dynamic configuration of HMS. Considering the interfaces, MPCT is meant to handle (1 to N) communications, i.e. one

generic interface on one side and N protocols on the other. Therefore, the generic and standardized interface has to be defined for the first side. Finally, an important task is to handle the fundamental differences between the protocols. For example, an event-based protocol and a protocol where pulling operations are necessary will be fundamentally different. Anyway, it is needed to aggregate these different behaviours in order to fit the standard interface.

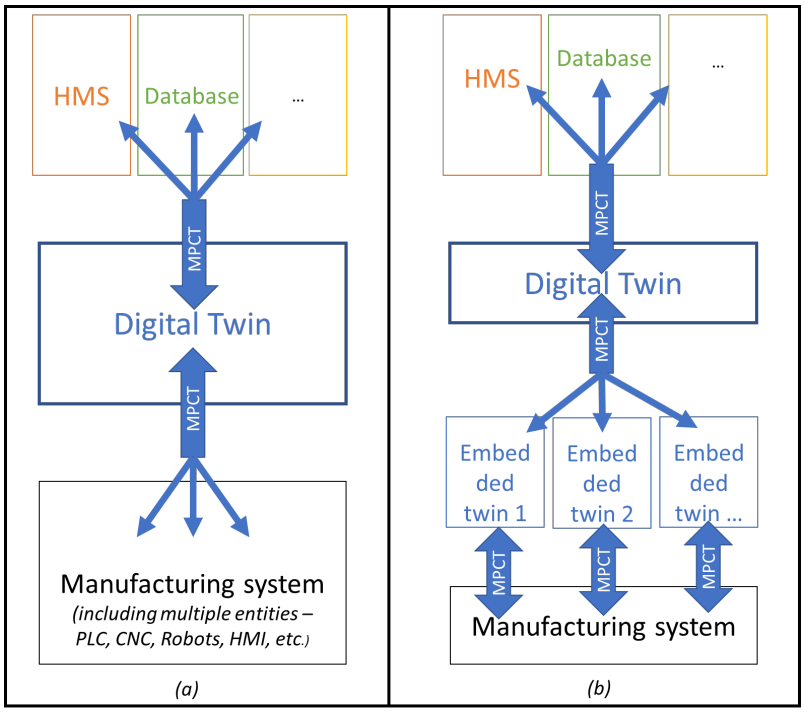

Fig. 3 Two possible implementations of MPCT

Finally, we propose an hybrid and non-symmetric communication system because all the components do no play the same role. In addition to the central manufacturing system we distinguish the persistent system, the information system linked to strategic management and the resources. There is no broadcast information. Figure 4 shows a mixed architectural pattern which is a compromise between simplicity, adaptability and heterogeneity.

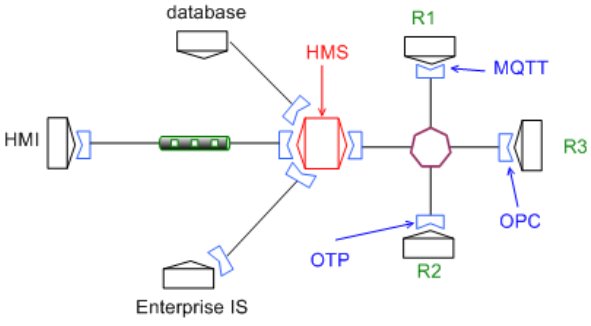

Fig. 4 Mixed HMS communication architecture

4 Case Study and Implementation

We illustrate the situation with the *SOFAL* production line implemented by Gamboa Quintanilla et al.[10]. As drawn by Figure 5, the legacy application is composed of three different parts: two applications (HMI, SOHMS) and the manufacturing workshop (or a simulator).

- The HMI (Human Machine Interaction) application enables the user to configure in real-time the whole layout (physical position of machines), products design, available services and required orders, to build various scenarios.
- The SOHMS (service oriented holonic manufacturing software) application represents the manufacturing engine and serves to control an assembly line: gets commands from HMI, defines plans for executing these commands according to the existing resources and finally sends decisions to the physical machines.
- SOHMS can be associated to the real manufacturing workshop or to an emulator *e.g.* Arena that simulate the workshop behaviour.

HMI and SOHMS are implemented in Java. All the communications are implemented at low level by TCP/IP connections using sockets.

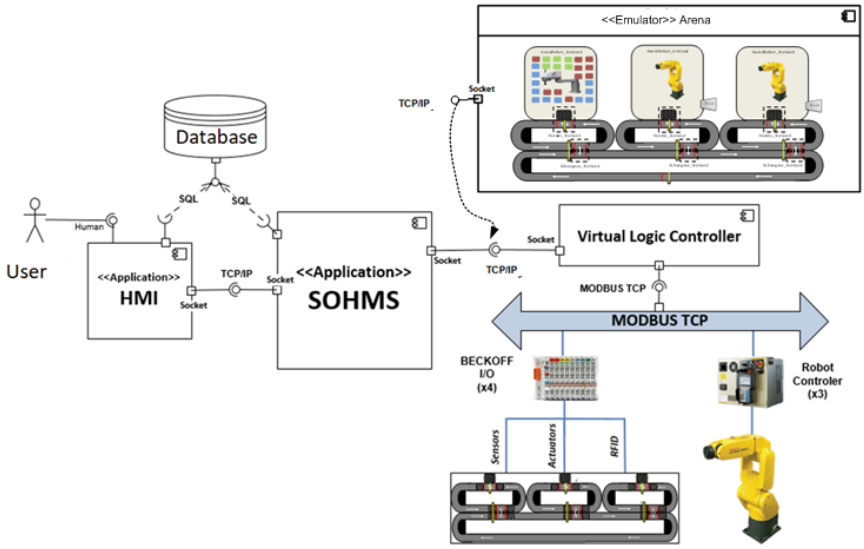

Fig. 5 SOFAL: a legacy SoHMS based manufacturing software [8]

During maintenance, the legacy application evolves due to updates of resource, order and products. Besides this we also changed the simulation tool and replaced `Arena` by `Flexsim`. As an example, a new protocol, MQTT (Message Queuing Telemetry Transport) is necessary for some resources.

Figure 6 shows the new *Sofal* architectural which is a trade-off between simplicity, adaptability and heterogeneity. The left part remains *as-is* because both the

persistent (JDBC) and controlling aspects (TCPSocket) are point-to-point communications with asynchronous message with mailbox queues. The right part will enable to add new resources with other protocols than `TCPSocket` and we develop the communication connectors to include these new resources.

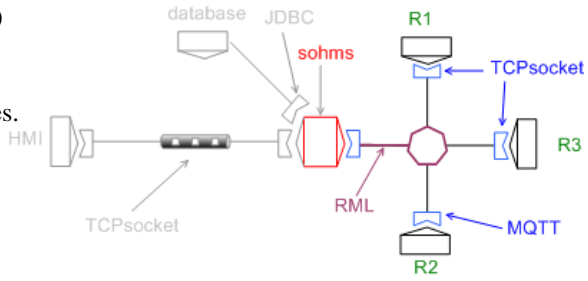

Fig. 6 New Sofal message communication Architecture

A resource message language (RML) defines the interface between SOHMS and the ressources. As an example, the messages are defined as follow:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement (name= "Message")
public class SocketMessage {
//ATTRIBUTS-----------------
   @XmlAttribute(name = "Sender")            public String sender;
   @XmlAttribute(name = "Receiver")          public String reciever ;
   @XmlAttribute(name = "Performative")      public String performative ;
   @XmlAttribute(name = "Encoding")          public String encoding;
   @XmlAttribute(name = "ContentOntology") public String contentOntology;
   @XmlAttribute(name = "Protocol")          public String protocol ;
   @XmlAttribute(name = "ConversationID")      public int conversationID ;
   @XmlElement( name = "Content")            public String content ;
```

Next listing shows the propagation of HMI actions on the SOHMS by a message sent on a socket:

```
SocketMessage requestOrder = new SocketMessage("HMI", "HMS", "TransportOFF",
                                    "XML", "Order", "RH modify", 2, "STOP");
// Launch OutboxSender to Send Messages
_communication.outBoxSender = new OutBoxSender(_communication._connexion,
                                        _communication.getOutput ());
_communication.outBoxSender.start ();
_communication.outBoxSender.sendMessage(requestOrder,false );
```

One can add as many protocols are needed and must create a handler for each. *e.g.* Protocol_RHmodify, given in the list of protocols of Figure 7. convert this messaging protocol into the resource specific protocol (in-/out) according to each resource communication requirements. RML could be extended to a general communication message language (CML) for all components linked to SOHMS (HMI, database, IoT...).

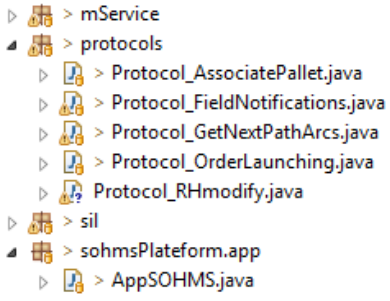

Fig. 7 Message protocols in SOHMS

Ongoing Prototype

We implemented a toy application called `MPComApp` to handle heterogeneous communications. This program reads a configuration file, to assign to each resource, a specific communication protocol. For instance, Resource R1 uses the socket communication protocol, while R2 uses the MQTT protocol. In this prototype, we used Java Swing components to interact with the application. Figure 8 and Figure 9 show respectively a TCP socket communication and a MQTT communication with the Mosquitto broker. In this last case, we used Windows console to launch Mosquitto.

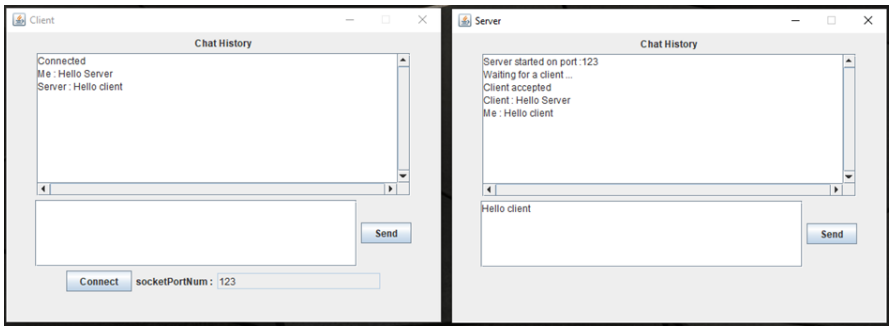

Fig. 8 Socket dialog of the `ComChoice` prototype

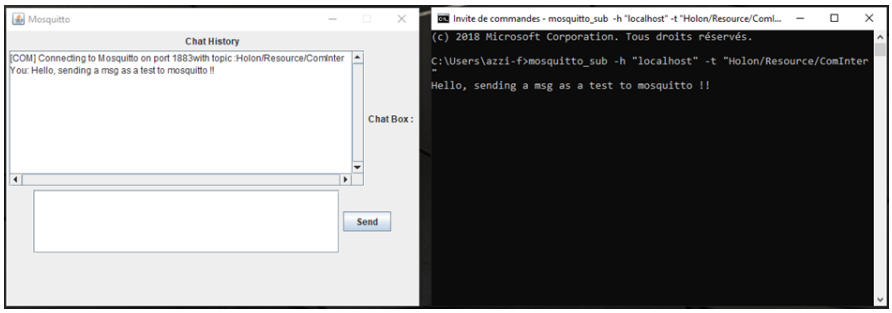

Fig. 9 MQTT dialog of the `ComChoice` prototype

Figure 10 show the class diagram of `MPComApp`. The SoftA instance plays the role of the communication hub of Figure 6. An handler instance of ComChoice is created for each resource. Based on the resource features, SoftA selects the configuration parameters.

By default, the resources play the server role in the TCP connection but two channels enable input/output bidirectional communications. In MQTT, each element is publisher on one topic and subscriber on another topic to enable bidirectional communications. Of course, if for a specific resource, a one-way direction is sufficient, the ComChoice handler will be notified by the configuration file.

This prototype shows the feasibility of heterogeneous communications. The next stage is to integrate it in the Sofal application.

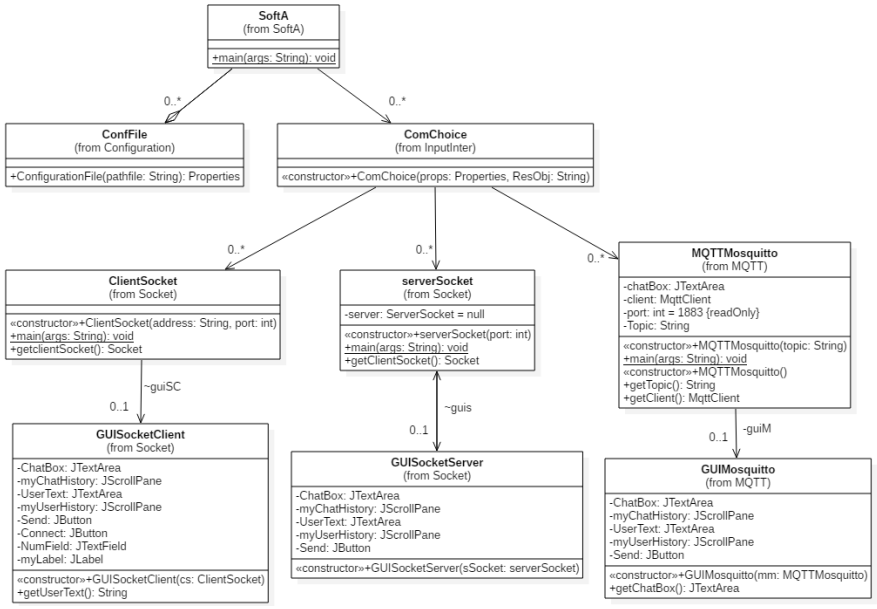

Fig. 10 Class Diagram of the `MPComApp` prototype

5 Discussion

This section points out relevant information on the applicability of the approach to different situations and discuss related references.

Developing cyber manufacturing control architectures is a difficult task for multiple reasons. One of them, if not the biggest one, is the communication system. Indeed, those control architectures are distributed by nature, and integrate a lot of different hardware and software to fulfil their functional specifications. Furthermore, the emergence of the Digital Twin in the architecture strengthen the need for communication because:

- The Digital Twin is intended to be placed as a middleware between the manufacturing system and the other applications of the control architecture. It plays there the role of the virtual replica of the actual system in the cyber world;
- The Digital Twin is not mandatorily monolithic, as embedded parts of the twin can for example be located close to the assets of the manufacturing system. Also, the aggregative nature of the manufacturing systems can be mimicked in the Digital Twin, which can then be composed of an aggregation of several twins, each representing their own specific part of the system.

The Multi-Protocol Communication Tool that is introduced here is promising in many ways, as it offers a generic, modular and configurable standalone application to handle any type of heterogeneous communication all along the architecture. The

objective is to simplify as much as possible those technical issues in order to make the developers focus on the most valuable tasks, including the decision making algorithms and negotiation patterns for example.

The communication standards considered for the moment are TCP sockets and MQTT. The aim is to provide both industrial informatics protocols (Profinet or OPC-UA for example) for low-level applications and classical protocols for high-level applications.

Apart from simplicity of implementation, the most important feature of such a tool is probably its evolvability. Indeed, it is impossible to imagine integrating all kinds of communication protocols in a stable version of the application. To cope with this issue, this application is based on a framework exhibiting, in the presented hybrid and non-symmetric communication system, the three main communication architectures (point to point, bus and centralized) in order to ease the evolution of the application towards new communication standards.

6 Conclusion

Software and services are pervasive in the next generation of manufacturing systems which are highly distributed for both the hardware and software parts. They require efficient but also customised communication middleware to handle heterogeneity. We propose a practical solution for multi-protocol communication that fits to small manufacturing systems. The solution is taking place in the evolution of the *Sofal* application.

An immediate perspective is to refactor the *SOHMS* framework of the *Sofal* application according to the new communication architecture. A mid-term perspective is to include the resource communication adaptors during the system code generation according to the principles given in [7]. A long-term perspective is to to check statically the service communications according to the approach introduced in [7].

References

1. André, P., Cardin, O.: Trusted Services for Cyber Manufacturing Systems, pp. 359–370. Springer International Publishing, Cham (2018)
2. Bal, M., Hashemipour, M.: Virtual factory approach for implementation of holonic control in industrial applications: A case study in die-casting industry. Robotics and Computer-Integrated Manufacturing **25**(3), 570–581 (2009)
3. Chalfoun, I., Kouiss, K., Bouton, N., Ray, P.: Specification of a reconfigurable and agile manufacturing system (rams). Int. J. Mech. Eng. Autom **1**(6), 387–394 (2014)
4. Cisco VNI: Cisco Visual Networking Index: Forecast and Trends, 2017–2022. Cisco White Papers (2019)
5. Colombo, A.W., Schoop, R., Neubert, R.: An agent-based intelligent control platform for industrial holonic manufacturing systems. IEEE Transactions on Industrial Electronics **53**(1), 322–337 (2006)

6. Covanich, W., McFarlane, D., Brusey, J., Farid, A.M.: Integrating a new machine into an existing manufacturing system by using holonic approach. In: 2007 5th IEEE International Conference on Industrial Informatics, vol. 2, p. 861–866 (2007)
7. El Amin Tebib, M., André, P., Cardin, O.: A model driven approach for automated generation of service-oriented holonic manufacturing systems. In: T. Borangiu, D. Trentesaux, A. Thomas, S. Cavalieri (eds.) Service Orientation in Holonic and Multi-Agent Manufacturing, pp. 183–196. Springer International Publishing, Cham (2019)
8. Gamboa Quintanilla, F.: Couplage des architectures holonique et orientée-services pour la conception de systèmes de production agiles. Ph.D. thesis, University of Nantes Angers-Le Mans-COMUE (2015). (in french)
9. Gamboa Quintanilla, F., Cardin, O., L'anton, A., Castagna, P.: A modeling framework for manufacturing services in service-oriented holonic manufacturing systems. Engineering Applications of Artificial Intelligence **55**, 26–36 (2016)
10. Gamboa Quintanilla, F., Cardin, O., L'Anton, A., Castagna, P.: Virtual Commissioning-Based Development and Implementation of a Service-Oriented Holonic Control for Retrofit Manufacturing Systems. In: T. Borangiu, D. Trentesaux, A. Thomas, D. McFarlane (eds.) Service Orientation in Holonic and Multi-Agent Manufacturing, no. 640 in Studies in Computational Intelligence, pp. 233–242. Springer (2016)
11. Giret, A., Botti, V.: Engineering holonic manufacturing systems. Computers in Industry **60**(6), 428 – 440 (2009), collaborative Engineering: from Concurrent Engineering to Enterprise Collaboration
12. Hoffman, A.J., Basson, A.H.: Iec 61131-3-based holonic control of a reconfigurable manufacturing subsystem. International Journal of Computer Integrated Manufacturing **29**(5), 520–534 (2016)
13. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
14. Jovanović, M., Zupan, S., Starbek, M., Prebil, I.: Virtual approach to holonic control of the tyre-manufacturing system. Journal of Manufacturing Systems **33**(1), 116–128 (2014)
15. Kruger, K., Basson, A.: Erlang-based control implementation for a holonic manufacturing cell. International Journal of Computer Integrated Manufacturing **30**(6), 641–652 (2017)
16. Masendeke, D.M., Basson, A.H.: Communication in a labview based holonic controller. In: International Conference on Competitive Manufacturing COMA'16 (2016)
17. Mcfarlane, D.C., Bussmann, S.: Developments in holonic production planning and control. Production Planning & Control **11**(6), 522–536 (2000)
18. Moraes, E.C., Lepikson, H.A., Colombo, A.W.: Developing Interfaces Based on Services to the Cloud Manufacturing: Plug and Produce, p. 821–831. Lecture Notes in Electrical Engineering. Springer Berlin Heidelberg (2015)
19. Morariu, C., Morariu, O., Borangiu, T.: Customer order management in service oriented holonic manufacturing. Computers in Industry **64**(8), 1061–1072 (2013)
20. Raileanu, S., Borangiu, T., Morariu, O., Iacob, I.: Edge computing in industrial iot framework for cloud-based manufacturing control. In: 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC), p. 261–266 (2018)
21. Shen, W., Norrie, D.H., Barthes, J.P.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. CRC Press (2003), google-Books-ID: JL3oHzCLBSAC
22. Silva, N., Sousa, P., Ramos, C.: A holonic manufacturing system implementation. Advanced Summer Institute (ASI'98) (1998)
23. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers **C–29**(12), 1104–1113 (1980)
24. Valckenaers, P.: Arti reference architecture – prosa revisited. In: T. Borangiu, D. Trentesaux, A. Thomas, S. Cavalieri (eds.) Service Orientation in Holonic and Multi-Agent Manufacturing, Studies in Computational Intelligence, p. 1–19. Springer International Publishing (2019)
25. Van Brussel, H.: Holonic Manufacturing Systems, pp. 654–659. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)