Enabling the Digital Thread for Product Aware Human and Robot Collaboration - An Agent-oriented System Architecture

Joe David[1,2], Andrei Lobov[2], Eeva Järvenpää[1] and Minna Lanz[1]

Abstract— **Customized product requirements are driving the need for variety oriented assemblies even in collaborative environments between humans and robots. This calls for the need for robots and humans to be intelligent in order to be aware of and adapt to different product needs. To address this, this study presents a novel approach and architecture to realize a digital thread that allows human and robot agents access to the product model at system run-time. The approach entails modelling a knowledge-based engineering (KBE) software as an agent which actively participates with collaborative agents via communication mechanisms standardized by the IEEE Computer Society. The architecture is described by four concurrent views and is discussed for its advantages and design rationale.**

I. INTRODUCTION

Today's manufacturing scenario is driven by customized product requirements that necessitates a high part variety on the part of manufacturers [1]. The requirement for variety puts high demands on the flexibility of the production systems. As a result, such assembly systems must be designed and operated to handle such product variant assemblies [2]. Of late, to enable flexible production, collaborative environments between robots and humans are becoming more commonplace as a result of either necessity attributable to task complexity and diversity [3] or foreseen advantages of combining human dexterity with strength, endurance, repeatability and accuracy of robots [4]. A very recent interview conducted with automation engineers and shop-floor operators highlighted the need for Human-Robot collaboration to be flexible to handle product variety [5]. Specifically, they expressed the need for intelligent robots that are updated automatically and aware of the product type it should work with in the context of human and robot collaboration. They further opined that assistance in switching smoothly between these products would assist the assembly work.

While the idea of using collaborative environments for variety oriented assembly seems like an attractive prospect, the domain of human-robot collaboration itself is still evolving [3] and plagued with challenges that span a broad spectrum including (but not limited to) human safety [6], human and robot knowledge or information models [7] and bi-directional communication and intent recognition [8] that facilitates acknowledged role-taking. Therefore, new approaches are needed and must accommodate this added complexity [1] involving product variety [1], intent recognition [9] and communication mechanisms [10].

One way for humans and robots to be product aware in a collaborative environment is to create a digital thread that serves the agent (human and robot) with access to and reason with product model they are interacting with. The digital thread notionally is purposed with integrating information and knowledge from traditionally siloed enterprise systems (such as product data management systems (PDM)) across different phases of the product lifecycle. This would allow the agents to create and share relatable mental representations of the product and the goals such as collaborative product assembly. Such relatable mental representations contribute to comprehensible behaviours and completion of complex tasks [9]. This would in effect create a digital thread that would also enable to gather the knowledge attained during collaborative assembly back upstream to the designers embracing the *design for collaborative assembly* paradigm [11]. While creating these mental models, the view of humans and robots as multiple agents passively introduced earlier is paramount. Such a view is appropriate in that the robot and the human must be capable of exhibiting dynamic and autonomous behaviour, which is the central to the notion of agency [3].

To this end, we embarked on research that preserved the agent view whilst allowing for establishing a digital thread. The result is a framework for human-robot collaboration that integrates the product design environment with a human-robot workcell to address the aforementioned issues at system run-time. This is accomplished with the flexibility that comes with modern knowledge-based engineering tools. The novelty of the framework mainly lies in three aspects: (i) integration of the product design environment with a human-robot workcell, (ii) modelling a product design software as an autonomous agent and which actively participates in interactions between agents that interact with the product and (iii) the use of an agent-oriented software middleware (JADE) based on (FIPA) standards [12] to facilitate agent communication for the afore mentioned purpose. The architecture of the framework encompasses a novel *interaction model* and separately a novel *information model*, neither of which is the subject of this study and remains as future work. However, for the sake of completeness the *interaction model* is briefly introduced here. This paper focuses on reporting the overall architecture and underlying hardware and software infrastructure and its communication mechanisms using the 4+1 architecture view model [13]. The framework reported here lays the foundation for the deployment of the said

[1]Faculty of Engineering and Natural Sciences, Tampere University, 33720 Tampere, Finland. <firstname>.<lastname>@tuni.fi

[2]Faculty of Engineering, Norwegian University of Science and Technology, Verkstedteknisk, 213, Gløshaugen, Norway joesd@stud.ntnu.no, andrei.lobov@ntnu.no

interaction and information models.

The next section elicits the requirements for the architecture development along with a few guiding principles. Section III presents the architecture of the framework using three concurrent views of the system. The use-case view is presented in Section IV that briefly describes how the developed architecture can prove useful. Section V discusses how we achieved the functional and non-functional requirements we set out with along with the design rationale of the system. A conclusion section in the end summarizes the study and presents future directions for research and implementation.

II. REQUIREMENTS AND PRINCIPLES

The architecture to be developed is to address the functional and non-functional requirements elicited below.

A. Functional Requirements

1) The robot and the operator should have dynamic runtime access to relevant information from the design environment with respect to the state of the assembly process.
2) Interactions:
   a) The operator should be able to communicate his/her intent with respect to the tasks to the robot.
   b) The interaction between the human and the robot must be non-invasive with minimal effort required from both sides and by doing so, communicate and collaborate effectively.
3) Both the robot and the operator should have cognizance of necessary information relevant to the assembly task in hand. For example, the sequence of the sub-assembly parts or dependencies between sub-assembly parts (for example, part B must be in place to assemble part C on top of part A).
4) Both the robot and the operator should have cognizance of each other's statuses with regards to the assembly task. For example, the robot is currently assembling part D (status: busy) as acknowledged by the operator or the operator has finished assembling part C (status: idle) as acknowledged by the robot.
5) A user interface should exist to manage all necessary information such as inputting IP addresses of the robot(s), logging information, calibration parameters of cameras, projector, etc.

B. Non-functional (Quality) Requirements

1) Scalability: The developed system must be a multi-agent system, i.e. scalable to include more than one of different kinds of agents.
2) Modularity: The architecture must enforce a modular design as opposed to a monolithic one. Components should be easily replaceable, wherever possible.
3) Usability: The system must be easy to use and intuitive and should not introduce unnecessary cognitive load on the operator.

C. Principles

1) The system's dependency on proprietary solutions is to be minimized.
2) The use of open-source software is encouraged
3) Use of standards where possible is encouraged

III. ARCHITECTURE

Using the 4+1 architecture view model [13], this section presents three of the five different concurrent views of the architecture to describe the system. The development view was seen as the least important from a research standpoint and has been omitted due to paucity of space while the scenarios are described in the next section. For readability, the names of elements (nodes, artefacts, etc.) of the framework are *italicized* while names of object-oriented programming constructs are in a `different font`.

A. Physical View

The physical view maps the software (artifact) to the hardware [13] and also depicts the physical connection between the hardware. It is useful in seeing the overall architecture of the system and is depicted in an UML deployment diagram in Fig. 1. It consists of seven main nodes: (i) the *Design Software Platform* node where the KBE application is installed, (ii) the *Robot* node, (iii) the *Ring Mouse* node, (iv) the *Projector* node, (v) the *Kinect* (sensor) node, (vi) the *Main Platform* node that hosts the JADE Main Container and to which the *Ring Mouse*, *Projector* and *Kinect* nodes connect to and (vii) the *Router* node that physically connects the *Design Software*, *Robot* and *Main Platform* nodes.

The *Design Software Platform* houses a knowledge-based engineering (KBE) software application, Siemens NX, which is also the Java Remote Method Invocation (RMI) server that exports the `NXSession` object that the *NXAgent* (RMI client), modelled as a JADE agent, uses as a proxy. JAVA RMI is a feature of JAVA that permits an object residing in one system (on the JAVA Virtual Machine - JVM) to access or invoke an object on another. Here the object is the `NXSession` object from the NXOpen API that exposes all the functionalities of NX to the JADE agent *NXAgent*. Next, the physical robot is connected to a robot controller that connects to the router via ethernet. The *Ring Mouse*, a part of the *Interaction Model*, is worn by the human operator and connects to the *Main Platform* via a wireless protocol such as Wi-Fi or Bluetooth depending on what the mouse supports. It is used to simulate mouse clicks on the *Interaction UI* artefact, which in principle is a purpose-built web-browser based user interface and part of the *interaction model*.

The *Projector* node connects to the *Main Platform* via a standard HDMI or DisplayPort connection. The *Kinect* sensor node has an inbuilt RGB camera, an IR emitter and detectors that map the depth through time-of-flight (ToF) algorithms. Both the *Kinect* and the *projector* are mounted atop the worktable that the human operator works on. While the *projector* is used to project the user-interface of the *interaction model* onto the worktable, the *Kinect* sensor is used as an input to the *interaction model* to track user

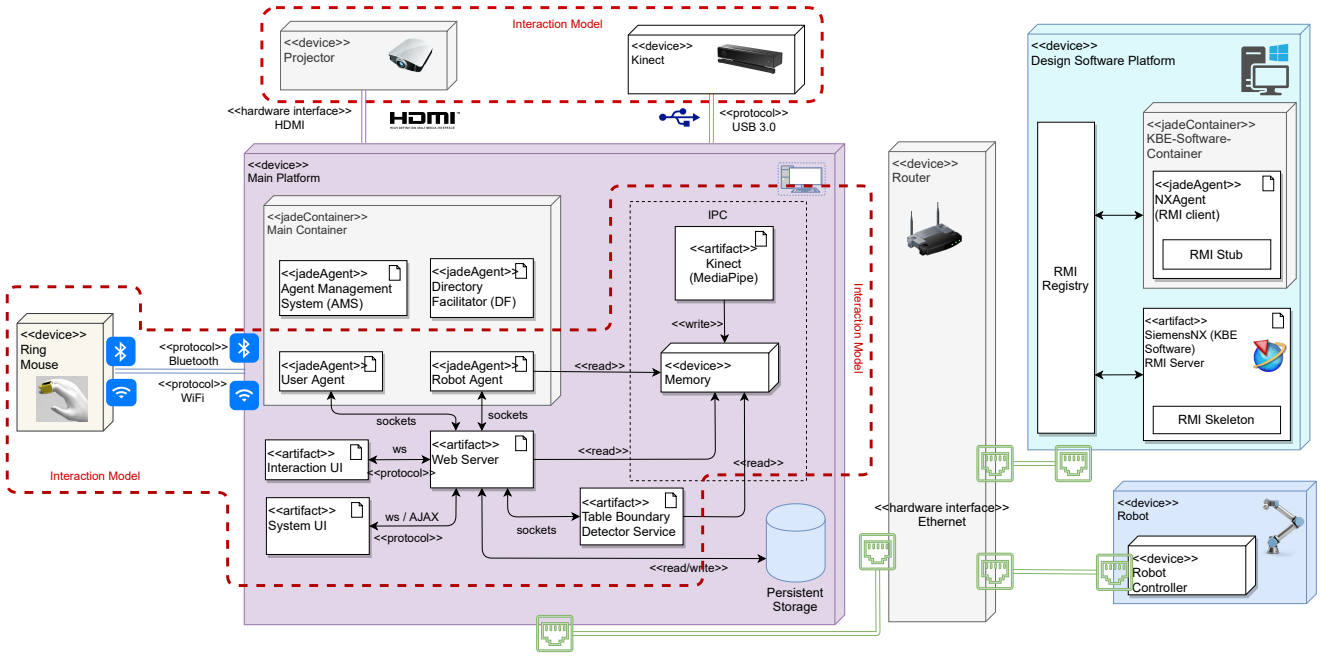

Fig. 1. Deployment Diagram representing the physical view of the system

behaviour and also for environment perception for the robot agent. The *Kinect* connects to the *Main Platform* via a USB that is mandatorily atleast version 3.0 owing the data transfer requirements of the Kinect RGB and depth sensors which can be upto 4Gbit/s.

The *Main Platform* contains the JADE *Main Container* that hosts the robot and the user agents. The *Main Container*, is the first container that exists on startup and hosts two special components; the *Agent Management System (AMS)* that is responsible for the functioning of the *Agent Platform* (not shown) such as creating and deleting agents and a *Directory Facilitator (DF)* that provides yellow pages services to agents in order to discover services provided by other agents. A detailed architecture of JADE is out of the scope of this study but the interested reader is encouraged to refer to the book by Bellifemine et. al. [14]. The *Web Server* also runs in the *Main Platform* and communicates with the JADE agents via Websockets. Websocket is a communication protocol that facilitates a full-duplex connection over a single TCP connection. This would be useful for bidirectional streaming data, the joint values of the robot for example. The *Kinect Stream* artifact is also implemented in the *Main Platform*, reads the RGB and depth frames from the Kinect Sensor and writes it to memory for interested artifacts to read from. This form of inter-process communication is (IPC) necessary as camera feeds can be quite large to transfer to interested parties real-time.

The *System UI* is a graphical user interface purposed to manage all necessary information such as IP addresses configuration, logging information, checking statuses for normal operation, displaying errors, calibration of cameras, projector configurations and operator logins. The database acts as an integrated persistent storage (Fig. 1).

B. Process View

The process view focuses on dynamic aspects of the system and describes its entailed processes and their communication at system run-time [13]. This section describes the process to setup the system before use and builds on the physical view to develop a deeper understanding of the inner working of the system. Further details of the dynamic communication processes between the agents not relevant to this study will be covered in the paper detailing the *interaction model*. A sequence diagram depicting the process is shown in Fig. 2. The following is the description of the sequences:

Setting up the Interaction UI:

1-1.6: First, the operator is to log in to the system with his/her credentials on the *System UI* which is verified by the back-end *Web Server*. A successful authentication leads to the landing page while an unsuccessful one notifies the operator and prompts for a re-entry.

2-2.2: In the *System UI*, the operator has the provision to open the *Interface UI* as a new window, which the operator then moves to the screen projected by the projector onto the worktable.

Setting up the KBE Software:

3: The user starts the custom developed digital thread application that provides the functionalities in the subsequent steps (2, 3 and 4) by means of menu buttons (Fig. 5.) for easiness of use.

4: The user starts the RMI registry next. The RMI registry is the namespace where NX places the `NXSession` object in the next step for lookup by the client.

5: Then user executes an NX Journal that binds the `NXSession` object to the RMI registry. (An NX Journal

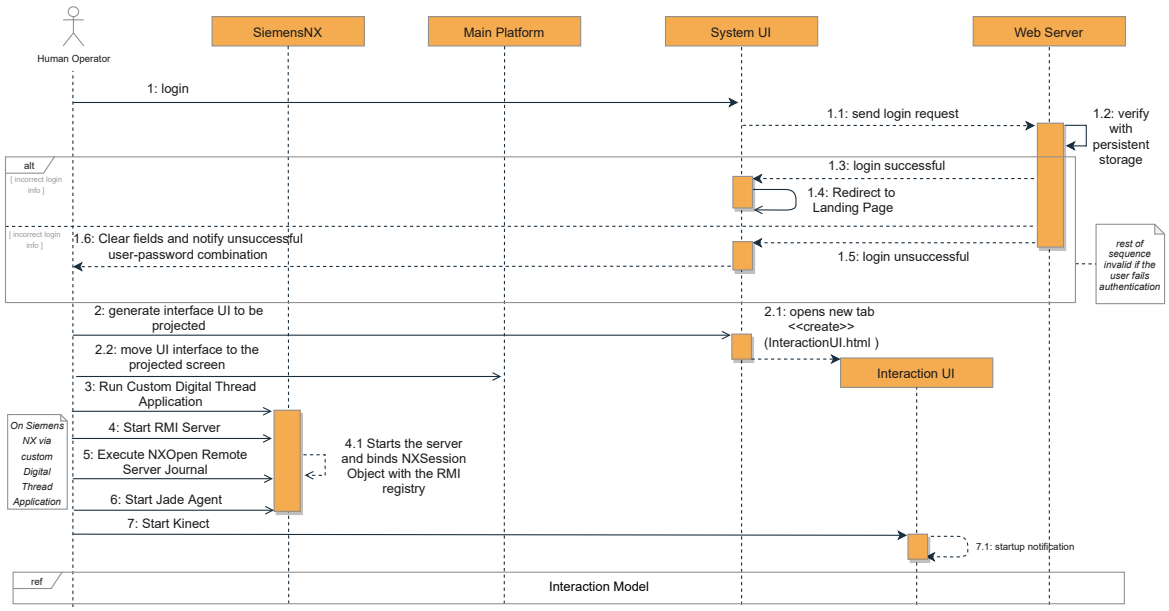

Fig. 2. Sequence Diagram representing the configuration Process

is a program that can be executed in the NX environment).
6: The user then starts the JADE agent (*NXAgent*) in a separate container (*KBE-Software-Container*) that looks up the `NXSession` object in the RMI registry. Thus the agent has access to the NX common object model via the NXOpen APIs.
7: The Kinect Sensor is turned on via the *Interaction UI*, which then displays a notification if it has successfully switched on for visual acknowledgement by the user. Behind the scenes, the operation runs a pre-compiled C++ binary of the *interaction model* that enables the human operator to use the interaction user interface.

C. Logical View

The logical view is the object-oriented decomposition of the architecture [13]. The class diagram in Fig. 3 represents the key object classes and their relationships using the principles of dependency, inheritance, composition and association.

Web WeThe *Web Server* composed of the *System UI* and the *Interaction UI* extends the `WebSocketServer` base class and overrides its key methods as shown. It depends on the built-in java `Robot` class to implement the *interaction model*. JADE provides an in-process interface (`Runtime`) that the `Webserver` uses to launch the JADE runtime during its initialization. This runtime is implemented by the `Runtime` class which uses a singleton pattern to provide a single instance of the class via the static `instance()` method. Hence, the dependencies between the `Webserver` and the `Runtime` class are as shown. This is used by the *Webserver* and Siemens NX to launch the main container and peripheral containers respectively. The agents are then initialized in these containers.

As for these agents, the core class is the `Agent` class from the JADE middleware which has child classes; the robot (`RobotAgent`), the human operator (`UserAgent`) and the design software agent (`NXAgent`). They exhibit JADE behaviours (`Behaviour`) that either happens once (`OneShotBehaviour`) or is repetitive (`CyclicBehaviour`). These behaviours belong to the class of simple behaviours (`SimpleBehaviour`) that a JADE agent may exhibit. Further, agents communicate with each other with ACL message objects (`ACLMessage`). Thus, both the message class and the behaviour class have associations with the agent class. At the object level agent class depends on objects of the behaviour and the `ACLMessage` class. The `NXAgent` class has additional fields and methods in comparison to the other agents. The `NXAgent` looks up the RMI Server to find the `NXServer` object that the `NXRemoteServerImpl` class has bound to the RMI registry. This then allows the `NXAgent` to execute the NXOpen API remotely at will.

IV. Scenarios

The scenarios describe three potential use-cases where the developed system can be used to extract run-time useful product information for collaborative product assembly

A. Pose Estimation

It is important to localize the product as the agents need to interact with them. Having access to the product model can train models that allow to detect them at run-time from perception of the environment through vision from a camera or point clouds from a depth sensor or a combination of both from an RGB-D sensor such as the Kinect sensor in the architecture presented in this paper.

B. Intent Recognition

Recognizing the intent of the robot in a collaborative environment between a human and a robot is crucial for operator

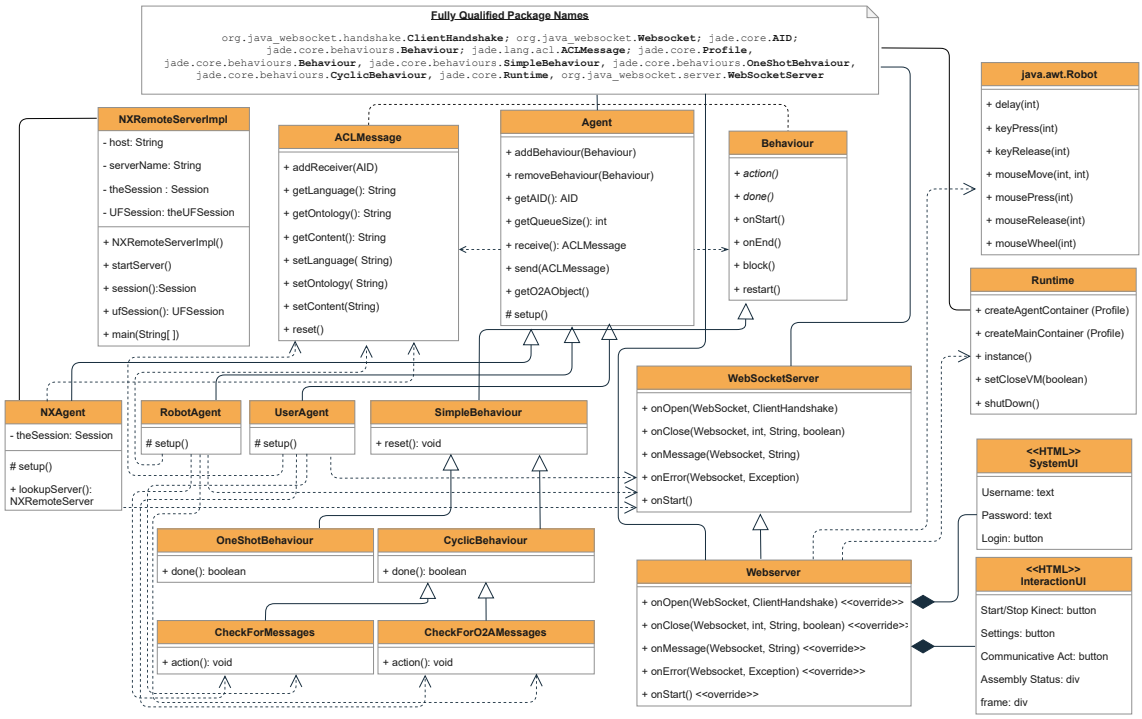

Fig. 3. Class diagram representing the logical view of the system

safety and cannot be inferred from robot motion alone. In such cases, product design information (CAD) can be an important enabler in projecting intentions on the product that has the operator's undivided attention during collaborative assembly. Required information can be computed by the agents dynamically at run-time. For example, the oriented bounding boxes of sub-assembly parts parts calculated from the CAD model of the product as shown in Fig.4 can be used to project robot intentions on its physical counterpart in the real world after having estimated its pose as described earlier.

C. Knowledge Representation and Reasoning

In our earlier work [15], access to the design model of the product was shown to be useful in reasoning about the assembly process to enable intelligent robotics. Specifically, the product model was annotated with semantic descriptions using the web ontology language (OWL) and the shapes constraint language (SHACL) was used to validate the robot's beliefs of the assembly process. The annotation was done automatically without human intervention using the programmatic flexibility of modern KBE systems. Such mechanisms can be extended further to validate other relevant run-time beliefs such as the sequence of assembly as an example

V. DISCUSSION

Now that the architecture of the system developed has been described using multiple concurrent views, in this section, a discussion on how it has accomplished the requirements elicited in Section II is presented.

Addressing the first requirement of integrating product design information also brings about the novelty of the research. Usually, such requirements are addressed in literature by developing ontologies specific to the domain or the use-case or by providing the agents with direct access to manually extracted information from the CAD model. Such approaches are difficult if not impossible to scale. While ontologies have, beyond doubt, proven to be useful in knowledge sharing and representation and will be a part of the information model of this system, there needs to be means to generate assertional knowledge constructs from different terminological constructs dynamically at run-time, even if its for the same resource. This is especially important in heterogeneous and dynamic environments where the existence of agents is non-deterministic (agents leave and join at will) and different vocabularies are used. This calls for the knowledge (product and product assembly) source to be online, dynamic and responsive to tailored need of agents. In our novel approach, we address this by modelling the design software as an agent that participates in agent interactions. This in effect translates to an agent that knows 'everything' about the product and can answer unseen queries about it. Thus, agents can communicate with the KBE software using the same agent communication mechanisms known to them using tailored interaction protocols and thus be able to handle different ontology vocabularies at system run-time. Further, an agent that is familiar with the NXOpen common object model API (that doesn't change), may choose to generate its beliefs of the product design data that it has no prior knowledge of, based off its CAD file which can be in any

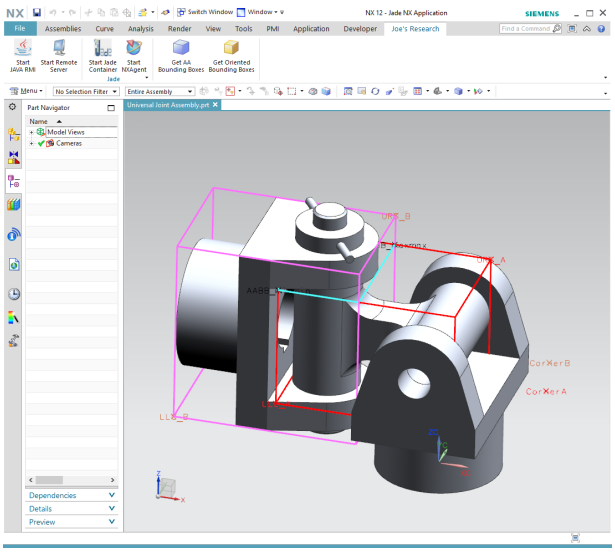

Fig. 4. Bounding boxes computed for an example Universal Joint CAD model

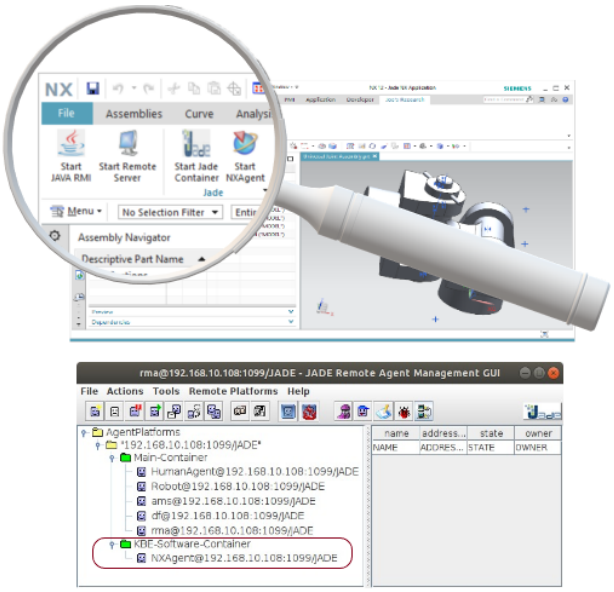

Fig. 5. Custom Application in NX to support launching of JADE *NXAgent* (top) and *NXAgent* as viewed from the JADE RMA GUI (bottom)

of the whole host of formats that is recognized by the NX software. The idea that agents can associate real world behaviour with CAD models is powerful and in theory would enable in training agents with product models and use such pre-trained models on unknown product models.

As for the functional requirements two to four, the architecture entails a mixed-reality interface via an *interaction model* that integrates with the JADE ecosystem to instantiate knowledge constructs with explicit semantics via an underlying *information model* to facilitate interactions between the human and robot agents. With the *interaction model*, in one direction, the operator is allowed to explicitly express his/her intentions with respect to the assembly process and on the other, the robot expresses desires that transform into intentions once acknowledged and agreed upon by the human operator exhibiting a master-slave relationship. This is expected to better intent recognition and minimize turn-taking behaviour. The information model on the other hand underpins the interaction model and uses the web-ontology language (OWL) to represent the product, process and resource along with other agent beliefs that aids the collaborative interaction. Further details not relevant to this study remains the subject of two papers that focuses separately on the interaction model and the information model currently in the works. Nevertheless, the presented architecture addresses these requirements.

Once the need for communication via concrete interfaces was established, a standards-based solution was sought. Consequently, an agent framework (JADE) that was an implementation of standards developed by IEEE Computer Society (originally by Foundation for Intelligent Physical Agents) was integrated to the interaction model. The FIPA standards were developed by over 60 members from more than 20 countries that involved both the academic and the industrial community [12]. The benefit is that these standards enforce the use of 'best practices' and using such an implementation means that they come ready 'out of the box' that is easy to develop and use or for reuse to integrate with new components (based on the same standard ofcourse). This is of special importance when such systems are scaled to operate in heterogeneous and distributed environments, typical of HRC use-cases. Further, JADE enforces a peer-to-peer architecture rather than, for example, the popular ROS middleware that enforces publish/subscribe semantics that works against the core agent to agent communication model. In JADE, each agent is identified by a global unique identifier and is allowed to join and leave at will and discover agents through yellow/white page services.

The choice of JADE also addressed the scalability quality requirement as JADE runs on the Java Virtual Machine which is platform-independent at both source and binary levels, and can work with any number of agents. This is infact how it is implemented, the *Main Platform* is a Linux platform while the design software agent (*NXAgent*) is hosted on a Windows Platform due to NX Requirements. This means that the system developed can be reused with a Linux installation that is free (and open-source) and if needed, along with any KBE Software of choice.

Further, ease of use is another quality requirement addressed here. The JADE framework is tightly coupled with the NX software in a purpose-built custom NXOpen Digital Thread Application. All the user needs to do is to key in the IP address of the main computer and start the NXAgent with the click of a button in NX. Fig. 5 shows the custom application in NX (in Windows) (top) and the NXAgent as viewed from the remote monitoring agent (RMA) from the JADE framework (bottom).

With the use of JADE middleware, communication be-

Fig. 6. Physical setup with integrated product design environment

tween agents was established. However, there was the need to integrate sensors for guidance for human agents and environment perception for the machine counterparts. Such an integration had to be real-time with latencies as minimum as possible and shared among as many agents that require it. A run-of-the-mill architecture would use a middleware like ROS to publish on a message topic that other agents subscribe to. Since this architecture adopted an agent-based approach, integrating ROS only for this purpose would not only be difficult (owing to the JADE concurrency model) but also mean that the agent-based mechanism of communication would be redundant and work against the peer-to-peer agent topology. To address this issue, the architecture adopts inter-process communication mechanisms, specifically shared-memory. We use a POSIX specification compliant approach (UNIX, LINUX, Mac OSx) to facilitate portability of the developed architecture. This meant that data from the Kinect Stream could be shared between any number of agents. A test with one agent recorded that image frames could be read from shared memory at over 1000 frames per second for the RGB stream (1920x1080px x 3 channels x 4 bytes/ch/px). This is currently an overkill as the Kinect Stream outputs at only 30fps. However, such mechanisms that support minimal latencies are important should we use streams from a faster sensor in safety applications, a use-case in the pipeline. This should in principle be faster that a ROS implementation but a performance comparison is out of scope of this study.

VI. CONCLUSION

This study presents a novel reusable agent-oriented framework architecture for dynamic product-aware human and robot collaboration. A physical setup using the framework is shown in Fig. 6. The framework is scalable, portable and based on standards. The framework makes use of free and open-source software everywhere except the KBE Software (JADE, *Web Server*, Ubuntu and libraries implementing the *interaction model*). The framework is intended to be used as an abstract foundation to be specialized to instantiate information models in JADE agents for human and robot collaboration analogous to how JADE is an abstract framework to implement agent-oriented solutions.

Future direction of this study include developing one such information model where agents use product design knowledge in one direction to allow for a collaborative workflow between the human and the robot and in the other, record the experience for the designer, thus facilitating a closed-loop digital thread. Two non-critical elements (functional requirement five) are also the subject of the next iteration as well; the System UI and its integration with the Persistent Storage.

REFERENCES

[1] A. Bilberg and A. A. Malik, "Digital twin driven human–robot collaborative assembly," *CIRP Annals*, vol. 68, no. 1, pp. 499–502, 2019.

[2] S. Hu, J. Ko, L. Weyand, H. ElMaraghy, T. Lien, Y. Koren, H. Bley, G. Chryssolouris, N. Nasr, and M. Shpitalni, "Assembly system design and operations for product variety," *CIRP Annals*, vol. 60, no. 2, pp. 715–733, 2011.

[3] Z. Kemény, J. Váncza, L. Wang, and X. V. Wang, "Human–robot collaboration in manufacturing: a multi-agent view," in *Advanced Human-Robot Collaboration in Manufacturing*. Springer, 2021, pp. 3–41.

[4] L. Wang, R. Gao, J. Váncza, J. Krüger, X. V. Wang, S. Makris, and G. Chryssolouris, "Symbiotic human-robot collaborative assembly," *CIRP Annals*, vol. 68, no. 2, pp. 701–726, 2019.

[5] P. Gustavsson and A. Syberfeldt, "The industry's perspective of suitable tasks for human-robot collaboration in assembly manufacturing," *IOP Conference Series: Materials Science and Engineering*, vol. 1063, no. 1, p. 012010, feb 2021.

[6] A. Hietanen, R.-J. Halme, J. Latokartano, R. Pieters, M. Lanz, and J.-K. Kämäräinen, "Depth-sensor-projector safety model for human-robot collaboration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Robotic Co-workers*, vol. 4, 2018.

[7] A. Olivares-alarcos, S. Foix, S. Borgo, and G. Aleny, "Knowledge Representation for Collaborative Robotics and Adaptation," in *IEEE international conference on robotics and automation (ICRA)*, 2021.

[8] M. Lee Chang, R. A. Gutierrez, P. Khante, E. Schaertl Short, and A. Lockerd Thomaz, "Effects of integrated intent recognition and communication on human-robot collaboration," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3381–3386.

[9] B. Hayes and B. Scassellati, "Challenges in shared-environment human-robot collaboration," *learning*, vol. 8, p. 9.

[10] L. Wang, R. Gao, J. Váncza, J. Krüger, X. Wang, S. Makris, and G. Chryssolouris, "Symbiotic human-robot collaborative assembly," *CIRP Annals*, vol. 68, no. 2, pp. 701–726, 2019.

[11] L. Gualtieri, G. P. Monizza, E. Rauch, R. Vidoni, and D. T. Matt, "From design for assembly to design for collaborative assembly - product design principles for enhancing safety, ergonomics and efficiency in human-robot collaboration," *Procedia CIRP*, vol. 91, pp. 546–552, 2020, enhancing design through the 4th Industrial Revolution Thinking.

[12] S. Poslad, "Specifying protocols for multi-agent systems interaction," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 4, p. 15–es, Nov. 2007.

[13] P. Kruchten, "The 4+1 view model of architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.

[14] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007, vol. 7.

[15] J. David, E. Järvenpää, and A. Lobov, "Digital threads via knowledge-based engineering systems," in *2021 30th Conference of Open Innovations Association (FRUCT)*, vol. 30, 2021.