Collaborative, distributed simulations of agri-food supply chains. Analysis on how linking theory and practice by using multi-agent structures

Alejandro Fernandez[1] , Jorge E. Hernandez Hormazabal [2,3], Shaofeng Liu[4], Hervé Panetto[5], Matías Nahuel Pankow[1] and Esteban Sanchez[1],

1 LIFIA, CIC/Faculty of Informatics, National University of La Plata; Argentina
2 Management School, The University of Liverpool, UK
3 Universidad de La Frontera, Temuco, Chile
4 Plymouth Business School, Plymouth University, UK
5 CRAN Université de Lorraine, CNRS, France
{Alejandro.Fernandez}@lifia.info.unlp.edu.ar

Abstract. Simulations help to understand and predict the behaviour of complex phenomena's, likewise distributed socio-technical systems or how stakeholders interacts in complex domains. Such domains are normally based on networked based interaction, where information, product and decision flows comes in to play, especially under the well-known supply chains structures. Although tools exist to simulate supply chains, they do not adequately support multiple stakeholders to collaboratively create and explore a variety of decision-making scenarios. Hence, in order to provide a preliminary understanding on how these interaction affects stakeholders decision-making, this research presents an study, analysis and proposal development of robust platform to collaboratively build and simulate communication among supply chain. Since realistic supply chain behaviours are complex, a multi-agent approach was selected in order to represent such complexities in a standardised manner. The platform provides agent behaviours for common agent patterns. It provides extension hotspots to implement more specific agent behaviour for expert users (that requires programming). Therefore, as key contribution, technical aspects of the platform are presented, and also the role of multi-level supply chain scenario simulation is discussed and analysed, especially under de context of digital supply chain transformation in the agri-food context. Finally, we discuss lessons learned from early tests with the reference implementation of the platform.

Keywords: digital transformation, supply chain, simulations

1 Introduction

In the last two decades, companies of all sizes have realized the importance of collaboration with suppliers and customers [1]. Supply chain collaboration can be defined as "long-term relationships where participants generally cooperate, share information, and work together to plan and even modify their business practices to improve joint performance" [2]. Despite the existence of abundant literature (as a recent literature review on supply chain collaboration [3] shows) organizations still have problems understanding how collaboration can impact their performance, and they are consequently reluctant to explore it.

ANONYMOUS is multidisciplinary, collaborative project whose goal is help agri-food value chains deal with risk and uncertainty. Researchers that participate in the project understand the importance of value chain collaboration. They regularly meet with different actors of the value chain, in an attempt to understand to what extent collaboration takes place, and to foster collaboration. In doing so, they must cross multiple knowledge boundaries. There are knowledge boundaries among organizations in the value chain, and there are knowledge boundaries among researchers and practitioners. Researchers understand they need to act as boundary spanners [4] (helping cross knowledge boundaries) and for that they need support.

Simulations have long been used as a means to understand and predict complex phenomena. The complexity can be understood as the variety of nodes and alternatives that exist in order to reach a solution, but, and more importantly, how protocols, hardware and software platforms are able to interact in order to provide the right recommendation, in the right time with the right quality. Thus, these kind of complex interactions are mostly found in supply chain domains, where resources, information, products and decision flows are required to commit to the end-customer requirement. Therefore, by the use of these simulations, what-if analysis are considered to help stakeholders and to understand the benefits and issues from co-operative environment rather than playing a pure transaction role with others [5][6]. In the line of this, but specially to provide a deep understanding and analysis about the relevance of such collaborations, this research work is committed to propose a platform that simulate communications and interaction based behaviours amongst supply chain stakeholders, which are represented by using agents. For this purpose, and based on the current analysis from [7], the main agri-food challenges and complexities are considered, this within the purpose of driving the supply chain agent-based structure. These aspects are: (a) complexity of interactions across agricultural value chain; (b) understands decision-makers challenge to build the solution; (c) Quantification of factors to generate desired solution; and (d) understand the whole-of-chain practical problem, especially when social, environmental and technological are the key drivers. Hence, this platform, which also consider the work from [6] as starting point, is oriented ease the creation, deployment, exercising and analysis of distributed supply chains within an agri-food view. This is achieved via the simplicity, standardization, scalability, collaboration of its main components. We argue that it offers adequate to support the task of boundary spanners looking to disseminate the benefits of collaboration in value chains.

2 Approach Overview

Multiple platforms exist to create multi-agent systems. In the context of agri-food, as [8] analysed, agent based model has been dominated by the following characteristics: single echelon supply chains; cases in high and middle income countries; unprocessed food products; use of empirical data; decisions related to production planning and investment; and the use of black box validation. Moreover, from technical point view validation, JADE [9], for example, is a well-known framework and supporting tools that aims at supporting the creation of multi-agent, peer-to-peer systems. It is domain independent, which is JADE, makes no assumptions regarding the behaviour of agents and the rules for the interaction. This makes JADE powerful, and consequently complex. A supply chain is a particular case of multi-agent system. Conversations follow certain specific conversation patterns (e.g., call for proposal, proposal, accept proposal). Existing agent platforms are rich and powerful but consequently difficult to use. Ad-hoc simulation tools are simpler to use, but limited to the features provided by its creators. We aim at something midway from both extremes. Instead of providing simulation authors with a full-fledge agent systems such as JADE, we have chosen to hide those fixed patterns behind an object-oriented framework. The framework captures common aspects of all supply-chain-simulations and implements then in a robust design. The platform, that we named **Sim-a-chain** currently focuses on demand-offer negotiations, similar to what occurs in agri-food domains.

Thus, The Foundation for Intelligent Physical Agents (FIPA) proposed a series of standards for the definition of agent systems, namely FIPA's Contract Net Interaction Protocol Specification and ACL Message Structure Specification [1]. The standards cover aspects such as inter-agent communication, agent management across and within agent platforms, and the transport and representation of messages between agents. Sim-a-chains adopts FIPA's proposals for message transport and representation of messages. For the content of a message (e.g., an offer), we looked at a newer development in the web; the Schema.org vocabularies [10] . These vocabularies have become a standard to represent certain types of objects in the web, in particular *products, offers, and demands* (which are the key elements in a supply chain conversation).

Existing agent platforms, and agent simulations platforms, requires users to deploy an agent execution engine to a server. Preparing and maintaining these servers (operating systems, security, updates, etc.) is only worth if simulations will be created and run frequently. A way to reduce the effort of server maintenance is virtualization. Sadly, traditional server virtualization does not help get rid of the cost of idle servers (when no simulation run, the server is still running). Moreover, when simulations grow in number of agents and computation needs, servers need to be provisioned. Recent developments in serverless [11] computing reduce the effort and cost of server maintenance, automate scalability, and remove the need to pay for idle time. Sim-a-chain has been built as serverless.

3 Sim-a-chain's abstract model

The core-modelling concept in Sim-a-chain is the Supply Chain Agent environment. Agents have a unique identifier, a short descriptive name, a list of products they can offer, an internal memory, and a behaviour. Agents have a messaging inbox. An agent's

[1]FIPA specifications - http://www.fipa.org/specs/ - Last accessed on June 2019.

behaviour depends on its type (e.g., a raw materials producer, a factory, a storage facility, a distributor, etc.). In this model, an agent's behaviour is determined by the way the agent reacts to messages from other agents.

Agent are normally dormant (idle). When an agent receives a message, it wakes up and reacts by activating the part of its behaviour that corresponds to the message type (e.g., a Call for Proposals). It accesses its internal memory, which holds information about available stock, or a list of suppliers. If necessary, the agent can send messages to other agents, including the sender of the message that activated it in the first place. After this, the agent goes back to idle state.

Messages in this model reflect what was proposed by FIPA's Contract Net Interaction Protocol Specification. From the protocol specification, we take only the proposal negotiation part, and leave out the contract fulfilment part.

A conversation among two agents starts when an *initiator* agent sends a `cfp` (call for proposal) message to another agent (the *participant*). In response to a `cfp`, the participant can send a `propose` message (if it can satisfy the call), or a `refuse` message to the initiator. When the initiator receives a `propose` message, it can accept the proposal, and consequently send an `accept-proposal` message to the participant. A proposal can also be rejected by sending a `reject-proposal` message to the participant. Upon receiving a `reject-proposal` message, the participant can try with a new proposal, or desist. A conversation among two agents reaches a (local) finish state when a `cfp` message is refused, when a `propose` message is accepted, or when a `propose` message is rejected and the sender of the proposal desists. In response to a `cfp`, an agent can in turn send `cfp` messages to its suppliers thus involving these agents into the conversation. A conversation reaches a (global) finish state when it reaches local finish state for all pairs of two agents involved in it.

When the first `cfp` of a conversation is created, a unique, global `conversation-id` is generated. This id will identify all messages that derive from the first call. In order to pair responses to requests, messages include a `replay-with` and `in-reply-to` attributes.

Agents live in "nodes". A node is a computing infrastructure that offers storage, computing, and communication facilities to agents. Within nodes, agents can be organized in collections. Nodes and collections are not to be confused with Supply Chains. A Supply Chain in this model is an abstraction that emerges from the dependencies and interactions among agents. Upstream connections in a chain are the result of agents knowing other agents as their suppliers. Downstream connections are the result of agents receiving call for proposal messages from other agents. Chains, therefore, span collections and even nodes. In this model, supply chains are not explicitly modelled. They are the result of the creations of multiple, potentially distributed, agent authors.

4 Reference implementation

We have implemented platform model version on the basis of modern digital web standards. At the centre of our approach lies an extensible object-oriented framework that simplifies development of new agent types and simulation models, and fosters extensibility. Nodes, the execution environment for agents, are implemented as serverless functions with minimal deployment requirements. Communication among agents is implemented via REST requests. Messaging protocol follows the FIPA standard, and message content adopts semantic web principles in the form of the Schema.org vocabulary. Collaboration among simulation authors occurs when agents in nodes under the control of different institutions / persons talk to each other. Usable editor tools lower the barrier for non-expert simulation editors. Following, we discuss each of these claims in more detail.

4.1 Serveless architecture

Serverless is a new systems architecture approach that removes the requirement of provisioning and maintenances of servers. Applications are implemented as a series of functions that can be activated by a variety of events such as HTTP requests, or database triggers. Serverless platform providers, such as Amazon, Google and Microsoft, take care of provisioning the resources necessary for functions to execute. Platform users are only charged for the resources used by functions (and only when they execute). In our case, this means no servers to provision and administer; no maintenance cost when no simulation is running; automated scaling of resources in response to simulation needs; and high availability and fault tolerance.

For the reference implementation of the Nodes element of the proposed platform we have selected AWS Lambda (Amazon Web Services implementation of serverless functions), and AWS DynamoDB for persistence. It can be ported with minimal effort to other platforms.

Communication among agents is implemented via HTTP REST requests. Communications is asynchronous. This means that a response to a message (e.g., a `propose` that responds to a `cfp`) will take place as an independent and asynchronous REST request.

The FIPA standard for the structure of ACL messages proposes various alternatives for the content of messages. Three attributes of a message describe its content: language, encoding, and ontology. In the case of framework messages are encoded following the mime-type “application/json”. As for the ontology, we have decided to use a selection of elements from the Schema.org vocabulary. In particular, we use the class http://schema.org/Demand and its properties for the content of `cfp` messages, and the class http://schema.org/Offer and its properties for the content of `propose` messages.

4.2 Object Oriented, Supply Chain Simulations Framework

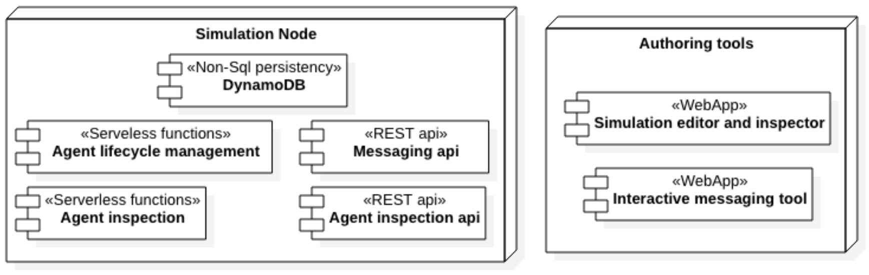

Fig. 1. Overall architecture implemented in the framework. Authoring tools connect to simulation nodes to create and interact with simulations. All communication occurs via REST protocols.

An object-oriented framework [12] captures key knowledge of a domain, offering a well-tested, robust implementation for a family of applications in the domain. It hides the complexity of the implementation of the common parts of all applications in the family (frozen-spots), while providing extension points (hot-spots) for applications developers to introduce variability. An object-oriented framework for supply chain simulations materializes the abstract model presented in previous sections. Fig. 1 provides an overview of the architecture of the framework and supporting tools.

The sequence diagram in Fig. 2 depicts how incoming messages are handled by the framework. The diagram shows the particular case of a call for proposal (cfp) message. The *messageReceived()* serverless function is activated by a REST request to a URL that represents the messaging inbox of an agent in a node. The framework retrieves the agent from the persistent storage, and delegates to it the processing of the Call for Proposals message. A hierarchy of classes model all possible types of messages (following the FIPA ontology as discussed in the previous section). Message classes offer utility methods to create message templates from them (thus reducing the room for configuration errors). In this case, a `propose` message template is created from the `cfp` message. The fields `conversationId, inReplyTo`, `sender` and `receiver` are set accordingly to match those of the original message. The content of the `propose` message depends on the specific behavior of the agent class and its internal state. The Agent class is an extension point (a hotspot) which means that simulation authors with programming skills can define subclasses that implement specific behavior for processing messages. Once the agent finished processing the message, any queued messages are delivered, the agent (its conversations included) is saved, and the *messageReceived()* function deactivates until a new message arrives.

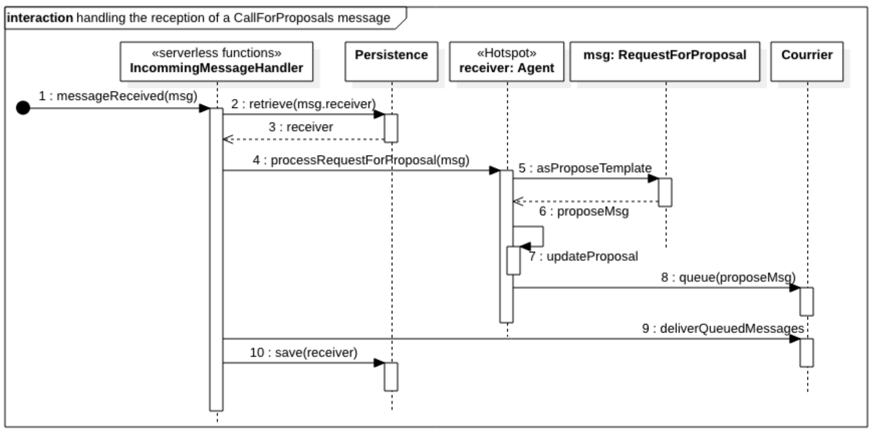
interaction handling the reception of a CallForProposals message
«serverless functions» IncommingMessageHandler
Persistence
«Hotspot» receiver: Agent
msg: RequestForProposal
Courrier
1 : messageReceived(msg)
2 : retrieve(msg.receiver)
3 : receiver
4 : processRequestForProposal(msg)
5 : asProposeTemplate
6 : proposeMsg
7 : updateProposal
8 : queue(proposeMsg)
9 : deliverQueuedMessages
10 : save(receiver)

Fig. 2. Sequence diagram for the handling of incoming messages. In this case, a Call for Proposal message. This interaction constitutes one of the frozen-spots of the framework (adapted from [6]).

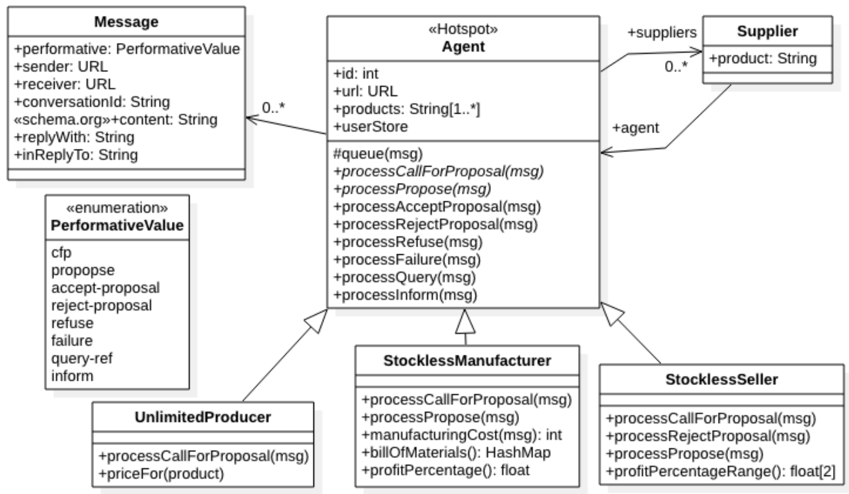

Fig. 3. UML class diagram showing the Agent class hotspot. Subclasses of the abstract agent class implement variability in agent behaviour. The message class with the possible values for its “performative” attribute constitute a frozen-spot (non-variable aspect) of the framework.

The (abstract) Agent class and its subclasses implement the most important hotspot offered by the framework. Each possible activation event of an agent (i.e., the reception of a different type of message) is implemented as a hook method that subclasses can override. The UML class diagram in Fig. 3 shows three example subclasses of agent, implementing specific behavior. The UnlimitedProducer agent class, for example, will answer to a `cfp` (call for proposals) with a `propose` if the product of the call matches one of those offered by the agent, regardless of the quantity requested (thus the

“Unlimited” name) of the call. The userStore property of a UnlimitedProducer agent stores pricing information for each product.

The framework hides the complexity of inter-agent communication from simulation authors. It provides out of the box agent behaviour for common agent patterns. It provides extension hot-pots to implement more specific agent behaviour for expert users (that requires programming).

4 Preliminary evaluation

In order to obtain a preliminary evaluation of the applicability of the approach we conducted two pilots. Asking a software developer to create new agent types using the framework, and presenting the platform to an expert on value chains (with experience in simulations) to obtain feedback.

The framework developer: A skilled software developer was tasked to use the framework to build new agents. He first received training in the hotspots and frozen spots of the framework. Then, a description of the desired agent behaviour was provided. He used one of the scripts provided with the framework to generate a stub agent class (JavaScript). Then, he modified the generated until de desired agent behaviour was obtained. For this, he had to iterate multiple time in a cycle: program - build simulation - run simulation - program. The following paragraphs summarize the findings. He concluded that the tool in fact allows the agent developer to abstract from agent life-cycle management, packaging of messages, and agent persistency. However, he noticed that with the current capabilities, the program-model-run-program cycle was too time demanding. There is currently no provision for isolated agent testing (alike Unit Testing). Moreover, bugs in agent code were hard to track as error reports intertwined framework and agent code.

The value-chain simulation expert: The platform (from the perspective of the simulation authoring tools) was presented to an expert on value chains. He had, in the past, built agent simulations using JADE. His opinion was that the general idea (as explained to him), and the reference implementation adequately captured the needs he had to create simulations of value chains. He was able to propose new agent behaviours that matched the capabilities and philosophy of the framework. In particular, he highlighted the improvement that this approach offered (as opposed to JADE) in terms of distribution of agents and collaboration among agent authors. He criticized the inability of the tools to graphically visualize (animate) message exchange in real time. He stressed the importance of making it possible to define agent behaviour as calls to external systems. This made sense, as many experts in value chain simulation frequently have programming skills, but not in JavaScript (or have already programmed an agent's behaviour that they may want in to integrate in a simulation) .

5 Conclusions and future work

Simulations are a valuable tool to understand and explore value chains. As already exposed by [6], supply chain complexities are certainly a source of limitations for collaboration. In the agri-food world, stakeholders are getting involved in not only one supply chain, but also in several. This necessarily implies multiple relationships, which turns even more complex the management of decision-making process. Hence, a high volume of experiments and scenarios are required to understand and mitigate agri-food complexities. In this context, using general-purpose agent platforms to create value chain simulations involves programming boilerplate code that does not add value to the simulation (e.g., checking the validity of messages, packaging message content, delivering, etc.). For that, a dedicated framework and supporting tools is a valuable contribution. Thus, Sim-a-chain, as an abstract model and a reference implementation has made visible the fact that a key area for future development is support to speed up the agent development cycle, supporting other programming languages and integration with existing systems. As further research, real-based evaluation will be performed to test and evaluate the usability and applicability of the simulation authoring tools to support value-chains decision-making for several types of food products.

Acknowledgement

This research is supported by Ruc-Aps, a H2020 RISE-2015 project, aiming at Enhancing and implementing Knowledge based ICT solutions within high Risk and Uncertain Conditions for Agriculture Production Systems.

References

1. Hieber, R.: Supply Chain Management. A Collaborative Performance Measurement Approach. vdf Hochschulverlag AG, Zurich (2002).
2. Whipple, J.M., Lynch, D.F., Nyaga, G.N.: A buyer's perspective on collaborative versus transactional relationships. Ind. Mark. Manag. (2010). https://doi.org/10.1016/j.indmarman.2008.11.008.
3. M. Ralston, P., Richey, R.G., J. Grawe, S.: The past and future of supply chain collaboration: a literature synthesis and call for research. Int. J. Logist. Manag. 28, 508–530 (2017). https://doi.org/10.1108/IJLM-09-2015-0175.
4. Keszey, T.: Boundary spanners' knowledge sharing for innovation success in turbulent times. J. Knowl. Manag. 22, 1061–1081 (2018). https://doi.org/10.1108/JKM-01-2017-0033.
5. Hernández, J.E., Lyons, A.C., Mula, J., Poler, R., Ismail, H.: Supporting the collaborative decision-making process in an automotive supply chain with a multi-agent system. Prod. Plan. Control. (2014). https://doi.org/10.1080/09537287.2013.798086.

6. Hernández, J.E., Lyons, A.C., Poler, R., Mula, J., Goncalves, R.: A reference architecture for the collaborative planning modelling process in multi-tier supply chain networks: A Zachman-based approach. Prod. Plan. Control. (2014). https://doi.org/10.1080/09537287.2013.808842.
7. Higgins, A.J., Miller, C.J., Archer, A.A., Ton, T., Fletcher, C.S., McAllister, R.R.J.: Challenges of operations research practice in agricultural value chains. J. Oper. Res. Soc. (2010). https://doi.org/10.1057/jors.2009.57.
8. Utomo, D.S., Onggo, B.S., Eldridge, S.: Applications of agent-based modelling and simulation in the agri-food supply chains, (2018). https://doi.org/10.1016/j.ejor.2017.10.041.
9. Moraitis, P., Spanoudakis, N.: The Gaia2JADE process for multi-agent systems development. Appl. Artif. Intell. (2006). https://doi.org/10.1080/08839510500484249.
10. Guha, R. V., Brickley, D., Macbeth, S.: Schema.org. Commun. ACM. 59, 44–51 (2016). https://doi.org/10.1145/2844544.
11. Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: Serverless Computation with OpenLambda. In: 8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16). {USENIX} Association, Denver, CO (2016).
12. Fayad, M., Schmidt, D.C.: Object-oriented application frameworks, (1997). https://doi.org/10.1145/262793.262798.