ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20200922160617
Subject review

Agent-based Modelling: Parallel and Distributed Simulation of Product Development Team

Ivan ČEH*, Mario ŠTORGA, Goran DELAČ

Abstract: The successful collaboration of team members plays a crucial role in product development. Research of different teamwork factors may help understand the process, and computer simulation can be a suitable method of achieving this goal. This work aims to give an overview of agent-based modelling of product development teams and survey existing models that tackle this problem. Some tools and frameworks for agent-based modelling will be described with an emphasis on parallel and distributed architectures which are used to improve performance and enable very big and complex simulations.

Keywords: agent-based models; crowdsourcing; distributed systems; multi-agent models, parallel computing; product development; social innovation; teamwork

1 INTRODUCTION

Ulrich and Eppinger [1] define the product as "something sold by an enterprise to its customer" and product development as "the set of activities beginning with the perception of a market opportunity and ending in the production, sale, and delivery of a product". The product can be something tangible with a physical shape or something intangible as an idea, service or computer software. Product development is rarely done by an individual and often by a team.

Some characteristics useful to optimize are product quality, product cost, development time, development cost and development capability for future projects. Observing and researching different individual and team characteristics can be helpful to better understand how they affect product development success. This knowledge can be used to develop computer models. Using computer simulations makes it possible to get results quickly and cheaply with better control over different parameters and observe how they affect the result. Managers can use this to make long-term predictions and possible interventions to get better results.

Helbing and Balietti [2] mentioned two different approaches: equation-based modelling (often used in natural sciences and engineering) and agent-based modelling, which seems to be more suited for socio-economic systems since most system behaviours have not been formalized mathematically. Thus, the agent-based approach seems better suited for the problem of teamwork in product development.

Some product development processes such as crowdsourcing and open innovation involve a large number of developers. Computer simulation of these systems may be very demanding in terms of time and memory. Some articles have already surveyed different types of existing models of product development [3, 4], but none of them focuses on processes with a large number of developers to our knowledge. This work focuses on agent-based models consisting of a large number of agents and describes some tools and computational aspects of simulation performance for such models.

Section 2 provides a definition and some basic concepts about agent-based modelling. Section 3 discusses some existing agent-based models for product development, crowdsourcing and social innovation. Section 4 describes some existing tools and frameworks for agent-based modelling, while section 5 focuses on agent-based modelling on different parallel and distributed architectures. Finally, some insights are drawn in Section 6, while Section 7 gives conclusions and possible future research goals.

2 AGENT-BASED MODELLING

Agent-based or multi-agent modelling is a widely used method in many sciences, especially social and natural ones. Its usage has been growing rapidly since the 1990s, but there is still no consensus on the definition and main characteristics. This approach is sometimes difficult to distinguish from other simulation methods like discrete-event simulation, Monte Carlo simulation and Petri net simulation.

To define models, Heath [5] starts with the concept of complex systems. He provides several definitions of the term, such as "regularly interacting or interdependent group of items forming a unified whole", but finally defines them simply as "something that transforms inputs into outputs". Complex systems are then divided into two groups: real systems which can be natural or man-made, and model systems. Model systems are simplified, finite representations of real systems. They are easier to comprehend and help to solve some complex real-world problems. They vary in complexity and understanding of real systems associated with them.

Some important elements of agent-based systems are:
- Set of agents: autonomous objects with rules controlling their behaviour.
- The environment in which agents are placed.
- Relationships and rules of interactions between agents.

Macal [6] mentions lack of universal agreement on the terms agent and agent-based model and gives four definitions for different types of agent-based models, in ascending order according to complexity:
- An individual agent-based model is one in which agents in the model are represented individually and have diverse characteristics.
- An autonomous agent-based model is one in which individual agents have internal behaviours that allow them to be autonomous, able to sense whatever condition occurs within the model in any time, and to act on the appropriate behaviour in response.

- An interactive agent-based model is one in which autonomous agents interact with other agents and with the environment.
- An adaptive agent-based model is one in which the interacting, autonomous agents change their behaviour during the simulation, as agents learn encounter novel situations, or as populations adjust their composition to include larger proportions of agents who have successfully adapted.

Simpler models are often used to study some simple processes like reproduction and interactions of bacteria. Social sciences and psychology usually require complex agents to study human behaviour. The behaviour of the system can seem very chaotic, and minor differences in initial conditions can lead to a major difference in outcomes. Errors of approximation, discretization of time and space or omission of some real system factors typically occur. Thus, simulation is often much better in gaining insights about real systems and discovering their properties than accurate predictions.

Some common environments in which agents are placed are space and network (graph). Space can be two-dimensional or three-dimensional, where coordinates of an agent can be integers or arbitrary real numbers and can differ in boundedness. Network environments are represented by mathematical graph structures consisting of points representing agents and edges representing links between them. Edges can be undirected or directed, depending on the type of relationship they represent. Multiple environments can be used in the same model. For example, agents can be placed in space and have a network that represents their relationships at the same time.

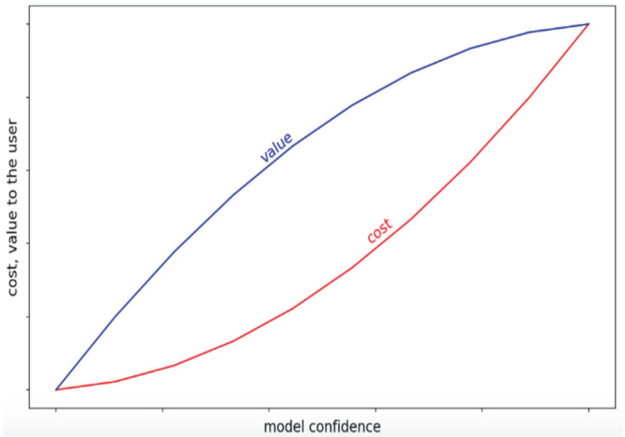

Figure 1 Dependency between model confidence, cost and value

Another important topic in agent-based modelling is verification and validation. Model verification can be defined as "ensuring that the computer program of the computerized model and its implementation are correct", and model validation can be defined as "substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" [7]. Developers are usually satisfied with some level of confidence that is not too hard to accomplish. Fig. 1 shows how it is possible to get some increase in model confidence with low cost and effort, but to get a higher increase in confidence, much higher cost and effort are required, which usually exceeds the value of higher confidence at some point. Some decision-making approaches for validation [8] are decision by the model development team, decision by the model development team with high user involvement, decision by independent third party or scoring model with different subjective scores.

3 AGENT-BASED MODELS OF PRODUCT DEVELOPMENT TEAMS

Perišić et al. [3] surveyed agent-based models of product development teams. Some of these models will be mentioned and some models that appeared after that article. Also, additional focus will be on models with a large number of agents where parallelisation is the most useful.

Part of the research consisted of searching for existing models of product development. Published papers in journals and conference proceedings were found using literature search databases Web of Science, IEEE Xplore, Scopus and ACM Digital Library with the query:
agent AND (model* OR simulat*) AND (team* OR group*) AND ("product development" OR "product design" OR "engineering design" OR crowdsourc* OR "open innovation" OR "social innovation")*.

The query gave 116, 376, 244 and 12 results respectively but only 66 of them were found to be relevant to this topic. Agent-based modelling can be used in teamwork to execute some tasks, help with communication and increase team productivity which will not be the focus of this work. Models examined here are models where agents represent individuals to simulate the reality of teamwork. Some models were only theoretical or mathematical and not oriented on agent-based simulation on a computer. The survey [3], references and citations additionally extended this set. In the end, 28 models which satisfied the criteria were chosen.

3.1 Agent-based Models of Product Development Team

VDT (Virtual Design Team) [9, 10] is a model developed at Stanford University. It uses an attention model, message exchange, learning and determining project duration by finding the longest sequence of non-parallelizable tasks. Later versions introduced some improvements like flexible organization structure, cultural differences in goals and values and dynamic tasks.

Construct [11], and its predecessors were made by Kathleen M. Carley et al. It uses social network analysis to draw some conclusions about adaptation, learning, information diffusion, risks and social change.

Dong et al. [12] developed a model to observe team effectiveness and its connection with different interactions between task process and team members and other factors. Face validation technique was used to evaluate the model. Some limitations mentioned are serial arrival of tasks, focus on highly centralized team and static learning ability.

TEAKS (Team Knowledge-based Structuring) [13] is also a model that uses initial team configuration and a set of tasks to predict team performance. It uses some emotional characteristics that affect team members and their relationships.

Natter et al. [14] used an agent-based model with neural networks consisting of two types of agents: marketing and production agents. They used it to compare

different organizational structures in searching for new products.

Sosa and Gero [15] developed a model to compare individual brainstorming in isolation and team brainstorming and concluded that both have its advantages contrary to early studies that underestimated group performance.

Perišić et al. [16] built the model to observe team behaviour in product development. Agents possess different emotions, cognitive mechanisms and beliefs which together with situation assessment affect its decisions and actions. It was used for different experiments like exploring the effect of experience on team performance and solution space explored. [17]

Singh et al. [18] used a simulation where agents can learn from experience and have properties like expertise, conformity and motivation to observe explored solution space and team agreement.

Songhori et al. [19] used NK performance landscape model to study the effect of team structure on individual performance and experience.

Bott and Mesmer [20] used an agent-based model with states and Markov transitions process to compare agile software development with other approaches.

Hulse et al. [21] used agent-based simulation to study the effects of collaborations and autonomy on design outcomes in a distributed multidisciplinary design team. They also found some advantages of collaboration compared to individual work.

Zhang and Thomson [22] found that project duration increased exponentially with product complexity. They included parameters like work efficiency, consultation, communication efficacy and task quality.

KABOOM [23] is a model developed by Lapp, Jablokow, and McComb to study cognitive style in team problem solving focusing on three important components: communication, specialization and team composition.

McComb, Cagan and Kotovsky [24] developed a CISAT (Cognitively-Inspired Simulated Annealing Teams) model. It uses simulated annealing and other techniques for exploration of solution space. Some interesting concepts modelled in this model are quality bias and self-bias.

Crowder, Robinson, Hughes and Sim [25] developed a model informed by research in two organizations. It includes many variables at the individual, team, and task levels. They conducted three experiments in which they examined in detail how performance depends on each variable and some combinations of variables.

Farhangian et al. [26] modelled software development group and compared effectiveness of different task allocation strategies and individual attributes on performance improvement. They analyzed different dimensions of individual personality and task (creativity, urgency, sociality and complexity) and their effect on the result.

Hultin et al. [27] modelled product development systems as Network-of-Networks with a focus on knowledge diffusion to help organizations in risk-mitigating decisions.

Many different models and resources can be helpful in different aspects of project management. Some of the most known models used to help in the development process are Capability Maturity Model Integration (CMMI) [28] and Team Software Process (TSP) [29]. CMMI focuses on four bodies of knowledge: systems engineering, software engineering, integrated product and process development and supplier sourcing, while TSP is composed of six competency areas: foundation and fundamentals, team foundations, project planning, project implementation and tracking, gathering and using data and scaling up.

Data from tools used in the development process can be used for data mining and gathering insights about the process. For example, some tools typically used in software development are Git for coding and storing information about the code changes, Confluence for communication and help in group tasks, JIRA for issue tracking and Trello for task tracking and management [30].

3.2 Agent-based Models of Crowdsourcing and Social Innovation

Most of the models already mentioned simulate small teams with less than 30 people. The number of people involved in product development depends on many factors, including company size, product complexity and desired development time. In some cases, the number of participants involved is measured in hundreds or thousands. One example is the Linux kernel with more than 10000 contributors and more than 1000 contributions each week on average [31]. These participants may involve employees of one large company, a community of people paid for contributions to the project or a community of voluntary participants driven by the goal of helping humankind, making software free and accessible for everyone etc. Compared to a small team composed of professionals, some advantages of large teams may be higher innovativeness and lower price, while the main disadvantage is the appearance of unprofessional and malicious contributors, which can cause the low quality of the solution.

Le and Panchal [32] developed a model of mass-collaborative product development. Product architecture is modelled to see how it affects the process. Time consists of cycles, and every participant decides whether he/she will contribute in every cycle depending on needed time, effort and benefit from the product. If they choose to contribute, they also choose the module to contribute based on its completion percentage.

Zhang et al. [33] developed a mass collaborative product development simulation that consists of management agents and design agents. Management agents track information about products and resources, design agents select suitable tasks and execute them while the program judges if the project is completed.

Osipov and Sukthanakar [34] modelled team forming based on social network analysis. Each agent has a set of skills and a set of neighbours. Tasks also have a set of skills needed to solve them. Agents estimate which groups have the potential to solve the task and join them with a preference for the small groups and groups similar to its proposal or neighbourhood.

Levine and Prietula [36] made the model of open collaboration for innovation and observed how different factors affect it. The population is divided into three groups by their level of cooperation: co-operators, reciprocators

and free riders. They found surprisingly good performance of open collaboration even when cooperators are a minority (reciprocators can be good substitution), free riders are present, diversity of goals is lacking or economic goods are rival.

Zhou et al. [37] modelled open source community projects, including participants' selection, performing tasks and cooperation. They found a limited impact of ordinary developers and a greater impact of core developers and innovation leaders. Unsurprisingly, cooperation was found to play an important role, and several management recommendations have been provided.

Zou et al. [35] built an agent-based model of crowdsourcing composed of the requester, task and workers. Workers search for a task and decide whether to accept it or not. Data for real experiment was used for calibration of coefficients via linear regression method and validation.

Yu et al. [38] proposed improvements in existing trust management approaches in crowdsourcing and designed a simulation to test their usefulness consisting of four groups of agents with different honesty. The system tries to identify malicious workers and avoid them in future tasks.

Jespersen [39] explores structuring design decisions to integrate crowdsourcing into the company innovation system. A cluster of companies aiming to innovate and compete for crowd intelligence is simulated. The simulated process consists of four steps: idea, design, choice and launch on the market. Crowd member chooses the company to collaborate based on the distance between user's innovation engagement and company's broadcast specificity.

Zou and Yilmaz [40] studied knowledge creation in global participatory science communities. New knowledge is created by concept creation, combination and elaboration. They observed three types of communities: exploratory, service and utility communities. Cliquish networks with low average path lengths are shown to be more effective in knowledge creation and diffusion than random networks. Innovativeness tends to increase with higher centrality but decrease after a certain point for exploratory and service communities.

4 TOOLS AND FRAMEWORKS FOR AGENT-BASED MODELLING

Agent-based modelling is, in some cases, done directly by general-purpose programming languages. Object-oriented languages are usually better suited to this problem by objects representing agents and functions representing behaviours and interactions. However, there are many specialized tools and frameworks that provide many frequently used functions, modules and graphical user interfaces, which eases visualization of the simulation and enables pausing and runtime modifications. Their appearance in the 1990s is one of the main events in the history of agent-based modelling. Some articles have tried to list all of them [41-45] and describe or compare some of them. They have found more than 100 different tools in total, some of which are used for specific fields within agent-based modelling. Some characteristics to consider when choosing between them are development and maintenance effort, programming language, execution performance (time and memory), scalability, user experience, operating systems supported, documentation etc. The most used programming language for this purpose is Java, but many other languages are used. Some of these tools with their main characteristics are listed in Appendix A but additional description is given in this section.

Swarm is the first one that appeared. It was developed in 1994 at the Santa Fe institute. One of the main features is an implementation of the model and graphical user interface (also called virtual laboratory) separately. Another concept specific to Swarm is a design of the model as a hierarchy of "swarms", groups of objects and a schedule of its actions. That is why Swarm is suitable for building hierarchical models. Its libraries are written in Objective-C, and only Objective-C was used for model development until Java version appeared. It was a predecessor of other tools which took many concepts from Swarm.

NetLogo is a user-friendly tool that is easy to use, a good choice for inexperienced developers and educational purposes with an excellent GUI. It is based on the NetLogo programming language, which is known for its simplicity and readability but is criticised for lacking some object-oriented features [42]. Its documentation and tutorials are very professional and extensive. NetLogo is especially suited for mobile agents on a grid space with behaviour dominated by simple local interactions [41]. It is a good choice for prototyping models, which will later be developed in another framework.

Repast (Recursive Porous Agent Simulation Toolkit) started at the University of Chicago as Java implementation of Swarm. One of its advantages is a good choice of many domain-specific tools, especially for applications in social sciences. It has many developments for different programming languages and specific purposes. Repast Symphony is one of the newer versions in Java which superseded some older developments and added some new features, especially related to Graphical User Interface. It is described as the most complete platform with good execution performances.

MASON (Multi-Agent Simulator of Neighbourhoods and Networks) is a Java-based tool focused on short execution time and large, computationally demanding models. It is low-level and gives the developer higher control over the execution, making it harder to learn and the best choice for experienced developers and models for which execution performance is critical. Simulations are reproducible across hardware, and it offers the ability to stop the simulation and move it to another computer. One disadvantage is the lack of domain-specific features, but there are extensions for some of them.

Mesa was developed in 2015 to enable agent-based modelling in Python 3 programming language and provide functionalities of previously mentioned frameworks [46]. Some useful properties of Python are popularity, readability and tools for data analysis and visualization. The browser window serves as a graphical user interface and client-server paradigm is used to separate model and visualization. It already has some useful built-in functions, but many are still being added since it is a new framework.

5 AGENT-BASED MODELLING ON PARALLEL AND DISTRIBUTED ARCHITECTURES

Big simulations may be very resource-intensive and demanding to perform. Parallelization is often used as a method to decrease execution time or execute more simulations at the same time. Program execution is divided between components that execute their part simultaneously. Agent-based models are usually suitable for parallelization because agents are autonomous, and their behaviour is dominated by independent actions. However, the development of parallel programs is typically harder, requires experienced developers and increases the probability of errors. [47] Some parallel architectures used for agent-based modelling are based on concepts of distributed systems and graphic processing units (GPUs).

5.1 Agent-based Modelling on Distributed Systems

Distributed systems consist of components (processes) that do not share the common memory, often located on different computers. Instead, these components communicate through messages which can be slow and diminish the usefulness of parallelization. For this reason, programming in distributed systems is focused on minimization of this communication overhead to fully utilize parallelism.

Rousset et al. [48] described some common problems of agent-based modelling on distributed systems and described some tools used for that purpose. In section 4, some characteristics important for choosing the tool for agent-based modelling are mentioned. Some additional properties characteristic of distributed systems will be described here.

The scheduler starts agent behaviour at the right time or order. Two main discretization approaches used for this are the time-driven and event-driven. In a time-driven approach, time is divided into time steps, and the schedulers run activatable behaviours in each step. The event-driven approach is based on events scheduled on a time scale. Event-driven platforms can be used for time-driven simulations by treating time steps as events. Most existing frameworks use the time-driven approach.

Reproducibility or determinism is a desirable property. When the same simulation is executed with the same parameters multiple times, the desired behaviour is getting the same results each time. Parallel programs are often not reproducible if the order of some events or messages is unknown and affects further program execution. Most frameworks have built-in mechanisms which ensure reproducibility despite these problems.

Properties of random number generator can be important for some models. The most used algorithm for random number generation is Mersenne twister. Parallelism in random number generation can violate the quality of random numbers and reproducibility.

Other phenomena that can cause nonreproducibility are causality errors. They occur when there are two events, one of which should be earlier in simulated time, but it executes later in real physical time and one of them affects the other one. The scheduler must not allow this to happen and ensure that simulated order is respected if there is some causality between events. Conservative synchronization accomplishes this by global barrier after each time step which does not allow events from the next time step to start if events from the current step are not finished. The main disadvantage of this approach is that it can slow down the simulation as many processors in the system can remain idle. Optimistic synchronization tries to resolve this problem by letting events execute in a different order than simulated while using a rollback mechanism. This is usually accomplished by keeping some previous states in memory and returning to the last stable state. This approach uses additional memory and possibly time if rollbacks are frequent. Most frameworks use conservative synchronization.

Implementation of communication between agents may be complex, so already existing communication libraries are usually used as middle layers such as MPI and RMI. Architecture can be fully distributed or in master/slave configuration if a central process starts and coordinates all the other processes. The central process can be overloaded which can slow down the whole simulation. Fully distributed architecture usually performs better and is used in most of today's frameworks.

Load balancing is the mechanism used to assign each agent to a certain process. Each process should get its part of the work, but no process should be overloaded. It is beneficial to place close agents who interact a lot in the same process to reduce the communication between processes. Load balancing can be divided into static and dynamic techniques. It is static if agents are assigned to processes initially and stay on the same processor during the simulation. Dynamic load balancing means agents can move to another process during the simulation. Moving agents consume some extra time but can speed up the process if agents move and change their activity level during the simulation. Space partitioning is often used for agents placed in space. Two standard methods used in two-dimensional space are field partitioning in which agents are divided by one dimension and grid partitioning in which agents are divided by both dimensions, as shown in Fig. 2. Sometimes adjacent areas can overlap, and the information about agents in the overlapping area is shared between processes which can simplify the communication between close agents. Network partitioning is often used with agents placed in the network and is based on graph partitioning algorithms.

Figure 2 An example of field partitioning (left) and grid partitioning (right) [41]

5.2 Tools and Frameworks for Agent-based Modelling on Distributed Systems

Some tools surveyed and compared in [48] are D-MASON, RepastHPC, FLAME and Pandora. Other tools mentioned were Jade, PDES-Mas, SWAGES, Ecolab, MACE3J and ABM++.

D-MASON (Distributed MASON) was developed by the ISISLab at the University of Salerno as a tool for scaling existing MASON models without the need for rewriting them. Initially, it used centralized (master/slave) architecture using ActiveMQ JMS library but later adopted fully distributed architecture with MPI. Agents can communicate with agents from the same process (including overlapping areas) or adjacent agents in the network by method calls. Space and network partitioning can be used together in the same simulation, which can be useful for simulations where agents are placed in space but often communicate with certain distant agents. Global parameters over several processes can be used.

FLAME (Flexible Large-scale Agent-based Modelling Environment) is a multi-purpose framework developed by the University of Sheffield based on C programming language. It uses finite-state automata with memory and XMML language to describe them for implementation. Two common distributions used in FLAME are space distribution and Round Robin distribution which assign each agent to one processor in a round. Communication between agents is done by message boards which can be limited to one process or broadcasted to all or some processes.

RepastHPC (Repast for High-Performance Computing) is a framework developed by the Argonne National Laboratory as an update of Repast Symphony for parallel and distributed environments. Communication is implemented using MPI and Boost libraries. Space partitioning with overlapping areas is provided, and communication is done by calling methods for agents in the same process or overlapping areas. It provides high scalability and good performances for the high number of cores and big simulations.

Pandora is a framework developed by the Supercomputing centre of Barcelona. MPI and μsik libraries are used for communication. It is specially adapted to spatial simulations and provides useful features like support for geographic information systems and grid partitioning with overlapping areas.

5.3 Agent-based Modelling on GPU

Graphics processing units (GPUs) are often used for the fast execution of big simulations. They can provide high speed and parallelism for a relatively low price. Some standard tools used for this purpose are FLAME GPU, RepastHPC and TurtleKit. Perumalla and Aaby [49] described some challenges of agent-based modelling on GPUs and compared its runtime performance to traditional CPU-based implementation. They observed an excellent speedup of two or three orders of magnitude at the expense of higher development effort, worse code readability, modularity and reusability. The existence of global memory makes remote communication and some other concepts easier to implement compared to distributed systems. GPU architecture is better suited to agents placed in a space environment and simpler homogenous models dominated by local interactions. The most used middlewares between GPU and the application or framework are OpenGL, CUDA and OpenCL. Some models used multiple GPUs to improve their performances [50, 51].

6 DISCUSSION

Many differences among agent-based models of teamwork in product development can be observed. Depending on the goals and phenomena that the model tries to simulate, different levels of complexity are needed. Some models are satisfied with elementary and unrealistic representations of human characteristics and behaviours, which can still yield useful insights into social phenomena. Models differ in many aspects, including representation of agents and their mental models, representation of tasks, environment, representation of their cooperation and communication, presence of different agent properties that affect their behaviour, number of agents, representation of management, modelling goals, the complexity of input and output, type of product and number of agents. Representation of tasks and agents can consist only of several parameters and formulas which determine behaviour from them (e.g. [13, 22, 25, 33, 34]) or some more complicated structures like geometrical shapes [15], graphs [17] and NK landscape [19]. Number of agents can vary from 4 [15] to 1000 (e.g. [38, 39]) or 2000 [32]. Communication differs in the complexity of its representation and its result on the agent and the process. It can be predefined [13], performed periodically (e.g. [18, 28]) or triggered by agent's lack of knowledge (e.g. [25, 37]) and other factors. It can involve two agents or the whole team via meetings. The hierarchy can highly affect communication if agents tend to contact their supervisors for help [10]. The last three factors mentioned significantly affect the model's computational complexity. Therefore, models usually limit complexity in some of them. Models with a large number of agents usually use simpler agent representation and behaviours.

Existing models developed for distributed systems and GPU architecture are mainly used for models with simple agent behaviours and interactions where the number of agents is large. These architectures usually work better with space environments dominated by local interactions between agents. Models in chemistry, biology and natural sciences usually have these properties, while in social sciences and economics space component is not so emphasized. In economy and social sciences, distant communication and interactions are common, and network environment is often more useful in describing relationships between agents. GPU architecture is especially suitable for simple models, space environments and heterogeneity where many agents execute the same action simultaneously. Therefore, it is rarely used for agent-based modelling in social sciences and economics.

When distributed architectures are applied to agent-based models with network environments, some network properties can positively affect the performance and reduce communication between processes. One such property is cliquishness which can be described as a tendency for

nodes (agents) to cluster together. Network which can be divided into its parts (clusters) with a lot of interaction and communication inside them and little or rare interaction between them is suitable for distributed architectures because network partitioning can be applied and communication between processes can be relatively low. Fortunately, networks in social sciences tend to have this property. People often form groups usually connected to some place, activity or interest. In the case of product development, the formation of teams dedicated toward a certain part or component of product development is usual. Participants of the same team communicate and frequently interact to share knowledge, successfully fulfil their role and contribute to successful processes while interactions between different teams are not needed so often. If participants frequently change their roles and teams during the process, dynamic load balancing can improve the performance. These insights suggest that distributed architectures can be suitable for agent-based modelling of product development teams, especially if the simulations are computationally demanding because they contain many agents with complex properties and behaviours. GPU architecture does not seem the best choice for this purpose.

7 CONCLUSIONS

Successful cooperation between team members in product development plays a key role in the economic success of many firms. Research of different process factors and their contribution to success can help in the organization and management. Computer models are the fast and cheap way of performing many experiments and analyses of the system and drawing conclusions. Agent-based models proved to be a good tool for this purpose. As big product development projects are becoming more frequent and certain paradigms like crowdsourcing, open innovation and social innovation are gaining in importance, the number of participants in the process can become very big, and the processes can be very complex and computationally demanding to simulate. Parallel and distributed architectures may help take over that burden and improve computer constraints like execution time and memory.

Existing models are usually used to observe a narrow set of phenomena occurring in real systems and may be useless for usage outside of it. However, models covering a large set of real system behaviours can lose focus and omit certain factors significant for some behaviours. A decision about the exact purpose of the model and its scope is an important part of the modelling process. Simulations are usually less confident indicators of causal relationships than real data collection. This can be resolved by validation, but it is missing or of poor quality in many existing models.

Some goals of future research can be:

- Using ideas from different existing models to develop a new agent-based model of big product development teams to simulate emergent properties and analyse the effect of different factors on the project success.
- Utilizing parallel and distributed architectures to get good simulation execution performance when it becomes computationally demanding.
- Model validation and comparison with other models despite the lack of real-world data.

Acknowledgement

This work was funded by the Ministry of Science, Education and Sports of the Republic of Croatia and Croatian Science Foundation project TAIDE (Team Adaptability for Innovative Product Development, grant number 7269, www.taide.org).

8 REFERENCES

[1] Ulrich, K. T. & Eppinger, S. D. (2004). *Product design and development*. McGraw-Hill/Irwin.

[2] Helbing, D. & Balietti, S. (2012). Social Self-Organization, Understanding Complex Systems. *Time*. https://doi.org/10.1007/978-3-642-24004-1

[3] Perišić, M. M., Štorga, M., & Podobnik, V. (2018). Agent-based modelling and simulation of product development teams. *Tehnicki Vjesnik*, *25*(Supplement 2), 524-532. https://doi.org/10.17559/TV-20170912134413

[4] Browning, T. R., Ranga, Ramasesh, V., & Neeley, M. J. (2007). A SurveyofActivity Network-Based ProcessModels for Managing Product Development Projects. *Production and Operations Management*, *16*(2), 217-240. https://doi.org/10.1111/j.1937-5956.2007.tb00177.x

[5] Heath, B. (2010). The History, Philosophy, and Practice of Agent-Based Modeling and the Development of the Conceptual Model for Simulation Diagram. *Browse all Theses and Dissertations*.

[6] Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, *10*(2), 144-156. https://doi.org/10.1057/jos.2016.7

[7] Schlesinger, S., Crosbie, R. E., Gagne, R. E., Innis, G. S., Lalwani, C. S., Loch, J., & Bartos, D. (1979). Terminology for model credibility. *Simulation*, *32*(3), 103-104. https://doi.org/10.1177/003754977903200304

[8] Sargent, R. G. (2010). Verification and validation of simulation models. *Proceedings-winter Simulation Conference*, 166-183. https://doi.org/10.1109/WSC.2010.5679166

[9] Jin, Y., Levitt, R. E., Chrlstiansen, T., & Kunz, J. C. (1994). *The "Virtual Design Team": A Computational Model of Engineering Design Teams*.

[10] Levitt, R. E. & Nissen, M. E. (2004, March 30). *The virtual design team (VDT): a multi-agent analysis framework for designing project organizations*. 115-120.

[11] Schreiber, C., Singh, S., & Carley, K. M. (2004). *Construct-A Multi-agent network model for the co-evolution of agents and socio-cultural environments*. https://doi.org/10.21236/ADA460028

[12] Dong, S., Hu, B., & Wu, J. (2008). Modelling and simulation of team effectiveness emerged from member-task interaction. *Proceedings-winter Simulation Conference*, 914-922. https://doi.org/10.1109/WSC.2008.4736157

[13] Martínez-Miranda, Aldea, A., Bañares-Alcántara, R., & Alvarado, M. (2006). TEAKS: Simulation of human performance at work to support team configuration. *Proceedings of the International Conference on Autonomous Agents*, *2006*, 114-116. https://doi.org/10.1145/1160633.1160649

[14] Natter, M., Mild, A., Feurstein, M., Dorffner, G., &Taudes, A. (2001). The effect of incentive schemes and organizational arrangements on the new product development process. *Management Science*, *47*(8), 1029-1045. https://doi.org/10.1287/mnsc.47.8.1029.10228

[15] Sosa, R. & Gero, J. S. (2012). Brainstorming in Solitude and Teams: A Computational Study of Group Influence. *International Conference on Computational Creativity*.

[16] Perišić, M. M., Štorga, M., & Gero, J. (2017). Building a computational laboratory for the study of team behaviour in product development. *DS 87-5 Proceedings of the 21st International Conference on Engineering Design (ICED 17)*, *5, Design for X, Design to X, Vancouver, Canada, 21-25.08.2017*.

[17] Perišić, M. M., Štorga, M., & Gero, J. S. (2019). Exploring the Effect of Experience on Team Behavior: A Computational Approach. *Design Computing and Cognition '18*. https://doi.org/10.1007/978-3-030-05363-5_32

[18] Singh, H., Cascini, G., Casakin, H., & Singh, V. (2019). A Computational Framework for Exploring the Socio-Cognitive Features of Teams and their Influence on Design Outcomes. *Proceedings of the Design Society: International Conference on Engineering Design*, *1*(1), 1-10. https://doi.org/10.1017/dsi.2019.3

[19] Jafari Songhori, M., Tavana, M., & Terano, T. (2019). Product development team formation: effects of organizational- and product-related factors. *Computational and Mathematical Organization Theory*. https://doi.org/10.1007/s10588-019-09302-8

[20] Bott, M. & Mesmer, B. (2019). Agent-Based Simulation of Hardware-Intensive Design Teams Using the Function-Behavior-Structure Framework. *Systems*, *7*(3), 37. https://doi.org/10.3390/systems7030037

[21] Hulse, D., Tumer, K., Hoyle, C., & Tumer, I. (2019). Modeling Collaboration in Parameter Design Using Multiagent Learning. *Design Computing and Cognition '18*, 577-593. https://doi.org/10.1007/978-3-030-05363-5_31

[22] Zhang, X. & Thomson, V. (2019). Modelling the development of complex products using a knowledge perspective. *Research in Engineering Design*, *30*(2), 203-226. https://doi.org/10.1007/s00163-017-0274-3

[23] McComb, C., Jablokow, K., & Lapp, S. (n.d.). *KABOOM: An Agent-Based Model for Simulating Cognitive Sale in Team Problem Solving*.

[24] McComb, C., Cagan, J., & Kotovsky, K. (2015). Lifting the Veil: Drawing insights about design teams from a cognitively-inspired computational model. *Design Studies*, *40*, 119-142. https://doi.org/10.1016/j.destud.2015.06.005

[25] Crowder, R. M., Robinson, M. A., Hughes, H. P. N., & Sim, Y. W. (2012). The development of an agent-based modeling framework for simulating engineering team work. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, *42*(6), 1425-1439. https://doi.org/10.1109/TSMCA.2012.2199304

[26] Farhangian, M., Purvis, M. K., Purvis, M., & Savarimuthu, B. T. R. (2016). Modeling Team Formation in Self-assembling Software Development Teams. *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*.

[27] Hultin, A., Tryfonas, T., Johnston, N., & Kirkman, M. (2014). Modelling effective product development systems as network-of-networks. *8th Annual IEEE International Systems Conference, SysCon 2014-Proceedings*, 89-96. https://doi.org/10.1109/SysCon.2014.6819241

[28] Constantinescu, R. (n.d.). *Reliability and Quality Control-practiceand Experience 31 Capability Maturity Model Integration*.

[29] Humphrey, W. S., Chick, T. A., Nichols, W., & Pomeroy-Huff, M. (2010). *Team Software Process (TSP) Body of Knowledge (BOK)*. https://doi.org/10.21236/ADA634307

[30] Ciancarini, P., Missiroli, M., & Sillitti, A. (2019). Preferred Tools for Agile Development: A Socio cultural Perspective. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11771 LNCS*, 43-58. https://doi.org/10.1007/978-3-030-29852-4_3

[31] GitHub-torvalds/linux: Linux kernel source tree. (n.d.). Retrieved August 27, 2020, from https://github.com/torvalds/linux

[32] Le, Q. & Panchal, J. H. (2011). Modeling the effect of product architecture on mass-collaborative processes. *Journal of Computing and Information Science in Engineering*, *11*(1). https://doi.org/10.1115/1.3563054

[33] Zhang, S., Li, Y., & Zhang, X. (2015). Agent Behavior-Based Simulation Study on Mass Collaborative Product Development Process. *Mathematical Problems in Engineering*. https://doi.org/10.1155/2015/689383

[34] Osipov, K. & Sukthankar, G. (2012). Forming effective teams from agents with diverse skill sets. *Proceedings of the 2012 ASE International Conference on Social Informatics, SocialInformatics 2012*, 44-48. https://doi.org/10.1109/SocialInformatics.2012.55

[35] Zou, G., Gil, A., & Tharayil, M. (2015). An agent-based model for crowdsourcing systems. *Proceedings - Winter Simulation Conference*, *2015*, 407-418. https://doi.org/10.1109/WSC.2014.7019907

[36] Levine, S. S. & Prietula, M. J. (2014). Open collaboration for innovation: Principles and performance. *Organization Science*, *25*(5), 1414-1433. https://doi.org/10.1287/orsc.2013.0872

[37] Zhou, H., Hu, Y., Zhang, X., & Zhao, D. (2016). Influence study of participants' characteristics on project evolution in open source community based on agent-based simulation. *Proceedings of the 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2016*, 527-531. https://doi.org/10.1109/CSCWD.2016.7566045

[38] Yu, H., Shen, Z., Miao, C., & An, B. (2012). Challenges and opportunities for trust management in crowdsourcing. *Proceedings 2012 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2012*, *2*, 486-493. https://doi.org/10.1109/WI-IAT.2012.104

[39] Jespersen, K. R. (2018). Crowdsourcing design decisions for optimal integration into the company innovation system. *Decision Support Systems*, *115*, 52-63. https://doi.org/10.1016/j.dss.2018.09.005

[40] Zou, G. & Yilmaz, L. (2011). Dynamics of knowledge creation in global participatory science communities: Open innovation communities from a network perspective. *Computational and Mathematical Organization Theory*, *17*(1), 35-58. https://doi.org/10.1007/s10588-010-9068-0

[41] Railsback, S. F., Lytinen, S. L., & Jackson, S. K. (2006). Agent-based Simulation Platforms: Review and Development Recommendations. *Simulation*. https://doi.org/10.1177/0037549706073695

[42] Berryman, M. (2008). *Review of Software Platforms for Agent Based Models*.

[43] Nikolai, C. & Madey, G. (2009). Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. *Journal of Artificial Societies and Social Simulation*.

[44] Allan, R. (2009). Survey of Agent Based Modelling and Simulation Tools. Retrieved November 22, 2019, from https://www.researchgate.net/publication/30421560_Survey_of_Agent_Based_Modelling_and_Simulation_Tools

[45] Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. P. (2017). Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review*, *24*, 13-33. https://doi.org/10.1016/j.cosrev.2017.03.001

[46] Masad, D. & Kazil, J. (2015). Mesa: An Agent-Based Modeling Framework. *Proceedings of the 14th Python in Science Conference*, 51-58. https://doi.org/10.25080/Majora-7b98e3ed-009

[47] Logan, B. & Theodoropoulos, G. (2001). The distributed simulation of multiagent systems. *Proceedings of the IEEE*, *89*(2), 174-185. https://doi.org/10.1109/5.910853

[48] Rousset, A., Herrmann, B., Lang, C., & Philippe, L. (2016, November 1). A survey on parallel and distributed multi-agent systems for high performance computing simulations. *Computer Science Review*, *22*, 27-46. https://doi.org/10.1016/j.cosrev.2016.08.001

[49] Perumalla, K. S. & Aaby, B. G. (2008). Data parallel execution challenges and runtime performance of agent simulations on GPUs. *Proceedings of the 2008 Spring Simulation Multiconference, SpringSim'08*, 116-123.

[50] Aaby, B. G., Perumalla, K. S., & Seal, S. K. (2010). Efficient simulation of agent-based models on multi-GPU and multi-core clusters. *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. https://doi.org/10.4108/ICST.SIMUTOOLS2010.8822

[51] Ho, N. M., Thoai, N., & Wong, W. F. (2015). Multi-agent simulation on multiple GPUs. *Simulation Modelling Practice and Theory*, *57*, 118-132. https://doi.org/10.1016/j.simpat.2015.06.008

Contact information:

Ivan ČEH
(Corresponding author)
University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: iceh@fsb.hr

Mario ŠTORGA, PhD.
1) University of Zagreb,
Faculty of Mechanical Engineering and Naval Architecture,
Ivana Lučića 5, 10000 Zagreb, Croatia
E-mail: mario.storga@fsb.hr
2) Luleå University of Technology,
Department of Business Administration, Technology and Social Sciences,
A 3535 Luleå, Sweden
E-mail: mario.storga@ltu.se

Goran DELAČ, PhD
University of Zagreb, Faculty of Electrical Engineering and Computing,
Unska 3, 10000 Zagreb, Croatia
E-mail: goran.delac@fer.hr

APPENDIX A

TOOLS AND FRAMEWORKS WITH CHARACTERISTICS

(Some characteristics are applicable only to frameworks for agent-based modelling on distributed systems.)

Framework	Programming language	Ease of use	Scalability	Reproducibility	Random number generator	Synchronization	Load balancing	Communication mechanisms
Swarm	Objective-C, Java	hard	high	-	-	-	-	-
NetLogo	NetLogo	easy	medium	-	-	-	-	-
Repast	Java, Python, C++, .NET framework	moderate	high	-	-	-	-	-
MASON	Java	hard	high	-	-	-	-	-
Mesa	Python	moderate	medium	-	-	-	-	-
D-MASON	Java	hard	high	yes	Mersenne twister	time-driven	dynamic	overlapping zones
FLAME	XMML/C	hard	high	no	Marsaglia	time-driven	static	message boards
RepastHPC	C++/ReLogo	hard	high	yes	Mersenne twister	event-driven	dynamic	overlapping zones
Pandora	C/C++	hard	high	yes	not available	time-driven	dynamic	overlapping zones