Modelling Agent Decision Making in Agent-based Simulation - Analysis Using an Economic Technology Uptake Model

Franziska Klügl
School of Science and Technology
Örebro University
Örebro, Sweden
franziska.klugl@oru.se

Hildegunn Kyvik Nordås
Business School
Örebro University
Örebro, Sweden
hildegunn.kyvik-nordas@oru.se

ABSTRACT

Agent-based Simulation Modelling focuses on the agents' decision making in their individual context. The decision making details may substantially affect the simulation outcome, and therefore need to be carefully designed.

In this paper we contrast two decision making architectures: a process oriented approach in which agents generate expectations and a reinforcement-learning based architecture inspired by evolutionary game theory. We exemplify those architectures using a technology uptake model in which agents decide about adopting automation software. We find that the end result is the same with both decision making processes, but the path towards full adoption of software differs. Both sets of simulations are robust, explainable and credible. The paper ends with a discussion what is actually gained from replacing behaviour description by learning.

KEYWORDS

Agent-based simulation; Decision making; Reinforcement Learning; Technology adoption

ACM Reference Format:
Franziska Klügl and Hildegunn Kyvik Nordås. 2023. Modelling Agent Decision Making in Agent-based Simulation - Analysis Using an Economic Technology Uptake Model. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

The advantage of agent-based simulation comes from the possibility to explicitly formulate the individual agents' decision making in its local context. This context may consists of a spatial, economic, organisational or ecologic environment –, and of other agents to interact with. A modeller takes the perspective of an individual agent for capturing the agents' decision making. What does the agent perceive? What does the agent know at a particular point in time? How does the agent decide which goal to pursue and which action to take next? Finally the agent may evolve as well as change its environment.

One of the challenges of developing agent-based simulation models concerns the different perspectives that a modeller needs to match during development: The agent's ego-perspective versus the population perspective often in form of aggregate values or bird's eye type of observations. The individual agent behaviour must be formulated in a way that together with all the other agents' behaviour the intended system-level phenomenon is (re-)produced during simulation run-time. Previous studies on how to meet this challenge include [3],[18],[8] or [22].However, the actual agent-level behaviour formulation still requires creativity and experience. An agent architecture hereby provides an underlying framework or pattern and as such guidance to structure the behaviour description. Although theoretically independent, some behaviour is more suitable to be formulated using a particular architecture than other behaviour.

During the last years, machine learning based approaches have been suggested and presented as successful alternatives to carefully craft agent behaviour models [11], [10], [21]. Yet, it remains unclear what is the actual advantage of a machine learning based approach to a conventional behaviour formulation. This paper is a first attempt to contrast two versions of the same model, one with a carefully elaborated decision making process and one using a simple Reinforcement Learning approach – as the most prominent form of machine learning for agents' decision making. The model that we use for our analysis, is a non-trivial agent-based simulation model of technology adoption[14] where two types of agents co-adapt, so that technology demand is matched with technology provision.

We created two variants of the same model, one in which agents decide based on a prediction of individual future profit and one in which they learn to take that decision using Reinforcement Learning. The questions that are addressed in this paper, concern the consequences of using those different decision making approaches or agent architectures. Do both model variants produce similar results? What are the advantages and disadvantages of a learning-based agent architecture in contrast to a direct formulation of the decision making processes?

In the remainder of this paper, we first summarise relevant background and related work. This is followed by a more detailed description of the technology uptake model that we use for this analysis. Section 4 gives a glance on simulation results, while Section 5 provides an in-depth qualitative discussion of the different technology adoption paths under the two versions of the model. The paper ends with a conclusion and some thoughts on future work.

2 DECISION MAKING PROCESS AND AGENT ARCHITECTURES

Agent decision making is a central element of any agent-based simulation model. It is therefore surprising that there is not more literature that systematically analyses which basic structures, patterns or agent architectures are suitable for which type of decision making

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

behaviour. The architecture describes the underlying structures, the setup of components and their connections, while a component is "an element that implements a coherent set of functionality" ([23], p. 83). A component in such a decision making architecture can be responsible for processing incoming sensor data, for handling communication or for updating the internal belief state. Russell and Norvig [20] use different agent architectures to categorise what agents are able to do. Their categorisation is only of limited value for agent-based simulation.

Agent architecture has been studied in general Multi-Agent Systems (e.g. see [16], [29]) and Siebers and Klügl [22] list agent architectures as a contribution of agent-based software engineering to agent-based simulation. The well-known BDI (Belief-Desire-Intention) agent architecture [9] has been used in agent-based simulation due to its resemblance of practical human-like means-end reasoning. Examples can be found in [17], [25]. Also, some layered architectures have been suggested for specific domains for example in mobility simulations: One component is responsible for the actual movement and a higher-level component for navigation (e.g. in [7]). Bandini et al. [2] also list different agent architectures in their overall treatment of software engineering aspects related to agent-based simulation. Klügl [13] distinguishes between behaviour-generating (first-principle planners), behaviour-configuring (BDI) and behaviour-describing (rules, processes) approaches. Cummings and Crooks [5] list a) reactive agent architectures – simple agents just react to short-term stimuli; b) finite state machines that require full specification and complex cognitive architectures that give more flexibility by integrating goal- and plan-based behaviour.

There are a number of more elaborated architectures specifically developed for generating complex social behaviour. Examples are cognitive science architectures such as Soar (e.g. used in [27]) that were originally designed as elaborated models of human decision making. Other were specifically developed for agent-based simulations, such as PECS [28] or BEN [4] integrating emotions, normative behaviour, values, culture, etc. Balke and Gilbert [1] review 14 decision making architectures relevant for computational social science comparing architectures along five dimensions: cognitive, affective, social, norm consideration and learning.

During the last years, the use of learning approaches for constructing agent decision making models in simulations has increased. Already in 2005, Takedama and Fujita compared different Reinforcement Learning architectures in a simulation of a bargaining game [26]. Another early example – not in a game theoretic setting – can be found in Junges and Klügl [11] for learning pedestrian behaviour instead of programming it directly. They looked into the dependency of simulation results from particular configuration of the learning approach, including details of the reward function. Kamwoo et al. [15] use (inverse) reinforcement learning to identify behavioural rules from data that then serve as an input to modelling agent decision making. Their example scenario was an (artificial) kind-of party scenario with agents moving around and talking to each other. Cummings and Crooks [5] develop this idea further suggesting to use the optimisation capabilities of Machine Learning to find optimal decision making strategies of simulated agents in complex scenarios. They combine an actor model that learns using Deep Q-Learning with an interpreter that predicts the actors behaviour in a human readable finite state machine to ensure behaviour explainability.

Sert et al. [21] use Deep-Q-Networks for building an agent-based segregation model. The idea is here to enable to explore the space of potential agent strategies, testing different rules of interaction. Källström and Heintz [12] combine reinforcement learning and agent decision making in simulated settings to generate behaviours that are tunable towards for example higher risk taking or competitiveness in simulated opponents. De Oliveira et al. [6] compare Q-Learning and Multi-Armed Bandit algorithms for learning route choice decision making models. None of those publications actually contrasts rule-based behaviour descriptions versus learning-based approach.

Despite an increasing interest in combining Reinforcement Learning with agent based simulation, there is not much work on complex decision making models comparing traditional behaviour specification with learning specifications. Learning-based approaches are seen as effortful, with partially unpredictable outcomes, while traditional direct behaviour modelling need extensive sensitivity analysis to exclude hidden artifacts. For more complex agent-based simulation models, the effect of learning for the agents' decision making cannot be separated from a complex environmental model including other types of agents which may also learn. Modelling the effects of an agents' action is not simple. Thus the impact of the actual learning architecture is hard to capture. Anyway, calibration and sensitivity analysis cannot be avoided when using learning-based approaches. The question is then, what is actually gained from replacing behaviour description by learning?

3 ILLUSTRATIVE MODEL: TECHNOLOGY UPTAKE

We illustrate our analysis using a dynamic model of joint technology uptake in manufacturing and engineering [14]. This model was chosen as it exhibits a good level of agent decision making complexity, it is not as simple as some game theory inspired model, but also does not have the decision model complexity of some social science models. During model analysis, it turned out that outcomes depend on details of the decision making. This triggered the idea to test a alternative for instead of programming decision making details, enabling the agents to learn the particularities.

3.1 Basic Economic Model

The dynamic model includes two types of active agents, manufacturers and engineering firms and one type of passive entities, engineers. The active agents chose between two alternative business models. In the first, denoted as "consultancy" model, manufacturers engage engineers as consultants to provide services necessary for production. In the second, labelled "automation", manufacturers license AI-enabled systems integration software from engineering firms.

At the start of the simulation, all agents apply the consultancy model. At each simulation cycle they decide whether to switch from consultancy to automation, from automation to consultancy or stay with its previous choice. For both types of agents, switching involves a change in the way they operate. Manufacturers must upgrade skills and employ systems integration engineers to work with the software. Engineering firms must re-deploy their staff from consultancy to R&D where they develop AI-enabled software applications that automate the services they previously offered as consultants. For both types of agents, the cost and benefits of the technology shift is uncertain at the time they make the decision to shift. The switch is not one-way, but both manufacturer and engineering company agents may return to the consultancy model again, if deemed more profitable in a changed environment.

Technically, the two business models are represented by different production functions with corresponding unit cost and profit functions. Manufacturing agents differ as far as productivity is concerned. A Pareto distribution of productivity across firms is assumed. More productive firms have lower unit costs. The functional forms of the manufacturing production functions are such that unit costs fall more steeply with productivity in the automation model. Thus, the most productive firms will be the first to benefit from switching to automation, while the less productive firms will be better off when remaining in the consultancy model for longer [14].

During initialisation, a fixed pool of engineers are allocated at random to engineering firms. The engineers are mobile across firms, activities and type of agents as the model simulations progresses. The engineering firms observe the productivity of manufacturers and form expectations about how many of them are ready to switch to software. Each individual engineering firm agent randomly predicts what market share it might be able to capture. Based on that prediction, the engineering firm agents estimate their future profit and use this expected profit in their decision making.

The dynamics of the model stem from network effects from adopting automation software on the part of manufacturers and learning from experience on the part of consulting engineers. As more manufacturers use software, software producers can harvest more data from them, build better software and offer better customer services. This reduces the switching costs for manufacturers, and thus shifts the productivity threshold for which unit costs are lower in the automation scenario downwards.

Engineering company agents, who offer engineering consultants, learn from working with clients. The accumulated experience helps them identify commonalities across clients, which enables them to standardise and ultimately automate the services they provide – reducing costs for software development. Thus, engineering firms with long experience from working with highly productive manufacturers will be the first to develop software. Both types of agents decide on switching to automation by comparing expected costs and profits in the two business models.

In a standard economic model, the agents would have full information on market conditions, costs and profits, while production, costs- and input demand functions would be smooth and continuous. This approach misses some of the dynamics and constraints related to technology adoption in the real world. Development of the test model was aimed at more realistic settings with agents that have bounded rationality and transactions are lumpy[14]. The following list summarises relevant model features:

- All agents operate in competitive markets where input and output prices are given.
- Manufacturers buy consultancy services or software from one engineering firm only.
- The shift from consultancy to automation is irreversible for manufacturers.
- Engineering firms may sell consultancy services or software to many manufacturers.
- Engineering firms sell either consultancy services or software, not both.
- The shift from consultancy to automation is reversible for engineering firms.

In the following, we first describe the original agents' decision making using a process-oriented architecture for both simulated agents types - the engineering firms and the manufacturers. This is followed by a description of our re-implementation using Reinforcement Learning.

3.2 Process-oriented Decision Making

The process-oriented decision making involves agents deciding individually on which business model to pursue - consultancy or automation. Each agent decides based on which alternative yields the highest expected profits in the current period. Manufacturers know their own costs, productivity and production capacity in both business models, but the switching cost to automation is stochastic. Similarly, engineering firms know their own costs in both business models. They observe the productivity of manufacturers, but they do not know for sure how many manufacturers are ready to take up automation software, let alone what will be their future share of the software market, at the point when they decide whether to develop it.

The decision making process flows as follows:

(1) Manufacturers compare their expected profits in the two business models and decide which business model to pursue in the current period based on the highest expected profits.
    - The consultancy model: Each manufacturer announces a request for consultancy services. The quantity of services requested depends on the manufacturer's productivity.
    - The automation model: Each manufacturer announces vacancies for process integration engineers and a request for an automation software license.
(2) Engineering firms compare expected profits in the two business models and decide which business model to pursue.
    - The consultancy model: The engineering firm responds to manufacturers' requests for consultants by making an offer.
    - The automation model: The engineering firm develops automation software and offers software licenses to manufacturers.
(3) Both manufacturers and engineering firms recruit engineers as either consultants employed in engineering firms, software developers employed in engineering firms or systems integration engineers employed in manufacturing firms.
(4) Manufacturers select engineering firms who offer consultants or software depending on the business model they decided on and produce.

With individual, uncoordinated decision making, supply and demand of engineers in each activity may not match. Manufacturers without an appropriate number of engineers employed cannot produce. That means they face costs, but without income. This leads to an unconditional decision for consultancy in the next update cycle.

After synchronising the agents' decisions, the consultancy engineers' experience is updated and the new manufacturing firms having adopted the automation software, if any, are added to the network of manufacturer agents using automation, and a new cycle begins as illustrated in Figure 1 which shows the coordinated decision making process of both types of agents.

The implementation platform uses a random shuffle of all agents for update. So, the modeller has no control over which agent updates first. The vertical lines in Figure 1 show when update of all agents is explicitly synchronised.

Actually, this is a rather complicated, synchronised process because decisions are explicitly depending on each other: A manufacturer agent will just decide for using software, if there is software on the market. An engineering company agent will not decide for developing software, if there is no signal from a manufacturer agent that it would be actually interested in automation. If there are no system integrators that can be employed, a manufacturer agent would select to hire consultants. Also details of decision making matter. In [14] cost minimisation was used as decision criterion – manufacturer agents compare costs associated with both business models. Here, cost minimisation is replaced by profit maximisation, as profit can be easier matched with reward in Reinforcement Learning. Although economic theory indicates that both should lead to the same results, this leads to a slower technology uptake compared to the original implementation.

3.3 Learning Replaces Reasoning

Reinforcement learning is accepted as an approach for learning sequential decision making models [24]. It became popular during the last years, mainly due to the development of Deep Q-Learning approaches that do not require explicit formulation of state representations. For the endeavour here, we apply *stateless* Q-learning, which is one of the simplest Reinforcement Learning architectures available.

All agents learn individually. Both manufacturer agents and engineering company agents select one out of two alternative actions `consultancy` or `software`: Using/providing consultants versus integrating/developing software. Each agent associates those two actions with a Q-Value, that forms an estimate of the reward that the agent may receive when actually executing this action. Here, reward r_t is the profit that the agent makes in learning episode t from selling software/consultancy or from producing with consultants/software. After action execution, that means at the end of the production round, every agent receives this reward and uses it to update the Q-Value associated with the selected action a in the following way. If no production is possible – for example if the manufacturer agent fails in recruiting engineers as system integrators – the reward r_t is negative as the agent still faces cost.

$$Q_t(a) = (1 - \alpha)Q_{t-1} + \alpha r_t \tag{1}$$

As given above, this is a *stateless* Q-learning update of the Q-value for action a. That means, selecting an action is independent of the state that the agent is currently in. In standard Q-learning, the Q-value for action a would depend on the agents' state, so the agent would learn which action is best in which state. Ignoring state, the formula above becomes simpler than standard Q-learning. The additional term expressing the estimation of optimal actions taken in a future state is missing. α is the learning rate (here $\alpha = 0.5$). We use an ϵ-greedy strategy for balancing exploration and exploitation. During exploration, the agent randomly selects an action, during exploitation, the agents selects the action with the higher Q-value. We start with $\epsilon = 0.9$ probability for exploration. This is reduced in every round with a decay factor set to 0.99.

Our scenario is a clear co-adaptive scenario where many agents learn at the same time. The learning problem for an individual agent is consequently non-static, more in the beginning with a high share of exploration. With more and more manufacturer agents selecting software, the network of manufacturers grows and as a consequence costs of automation decreases and profit for software users increases. So, there is not just a feedback loop via the increased availability of software/consultancy services, but also related to direct cost calculations. To support continued learning in such a dynamic environment, we set a lower bound for ϵ, so that agents at any time can test alternatives with a very low probability. The consequence is that there is always a small share of manufacturer agents which do not use software even when the system has converged.

In the same manner as in the process-oriented decision model, the learning-based decision making is embedded into the basic economic model. The main difference is that while process-oriented decisions are driven by partially random individual expectations about the future, learning-based decisions are based on experience from the current and the previous period. In either case a manufacturer agent which decides to produce with software, must employ system integrators and find software on the market while an engineering company deciding to offer consultancy needs to wait for manufacturer agents to offer a contract. Figure 2 shows the synchronised learning-based decision architecture. Basically, there are three phases: The agents independently draw their decision, prepare the execution and actually produce. In the third phase, the agents evaluate their profit or loss and update the corresponding Q-Values.

Implementation is done using agent-based simulation platform SeSAm simsesam.org. Both implemented models will be available on the platform's webpage under "Contributed Models".

4 EXPERIMENTS AND RESULTS

The following simulation results - identical parameter settings for both model variants - contain 200 manufacturer agents and 60 engineering company agents. Manufacturer agents have an individual productivity drawn from a Pareto distribution. There are 3000 engineers in the economy. Two thirds of them are initially employed at engineering companies to be hired out as consultants while the remaining third is freely available on the labour market to be hired as consultants or integrators. All figures show averages over 30 runs.

We tested the effect of different learning parameter settings. Different values of α or the decay-factor for ϵ had only minor and

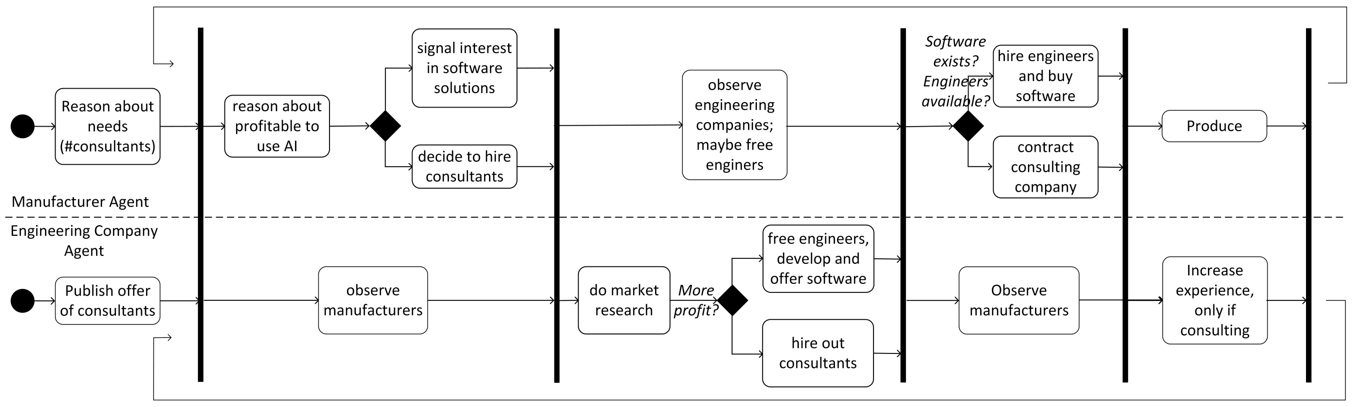
Reason about needs (#consultants)
reason about profitable to use AI
signal interest in software solutions
decide to hire consultants
observe engineering companies; maybe free enginers
Software exists? Engineers available?
hire engineers and buy software
contract consulting company
Produce
Manufacturer Agent
Engineering Company Agent
Publish offer of consultants
observe manufacturers
do market research
More profit?
free engineers, develop and offer software
hire out consultants
Observe manufacturers
Increase experience, only if consulting

Figure 1: Synchronised Decision Making Process

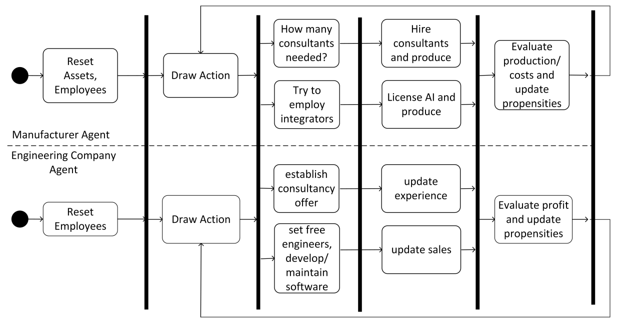
Reset Assets, Employees
Draw Action
How many consultants needed?
Try to employ integrators
Hire consultants and produce
License AI and produce
Evaluate production/ costs and update propensities
Manufacturer Agent
Engineering Company Agent
Reset Employees
Draw Action
establish consultancy offer
set free engineers, develop/ maintain software
update experience
update sales
Evaluate profit and update propensities

Figure 2: Synchronised Process with Learning

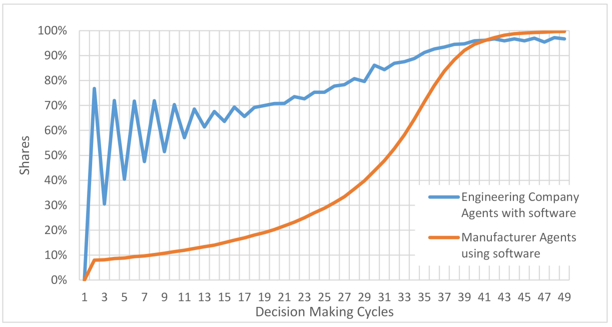

Figure 3: Process-oriented decision making: Percentage of manufacturers and engineering companies having switched to using/developing software.

expectable impact on the convergence speed. The most important setting was actually the decision to apply a non-zero lower bound for ϵ. Therefore, there is no full convergence. On the other hand, the parameter approaching zero, resulted on a premature convergence. Initial bad (unrealistic) decisions that lead to not being able to produce, resulted in highly negative rewards.

Figure 3 shows the results of the process-oriented decision making. Manufacturer agents compare potential profits. Due to the network effect, the more of them use software, the cheaper it becomes and thus the more manufacturers can afford replacing consultants in the next period. The zick-zack of software development at engineering company agents comes from overoptimism as agents predict a higher income from software than they actually can realise. Making a loss, leads them to return to the consultancy model. With an increasing share of manufacturer using software, the share of engineering companies making profit in software market increases. As time passes, 100% of manufacturer agents as well as almost 100% of engineering company agents use or develop software respectively. Importantly, the S-shape of the manufacturer adoption rate curve reflects stylised facts from the technology diffusion literature.

Also in Figure 4 the curve of manufacturer agents using software shows an S-shape, at least after the initial phase in which the agents mostly do exploration and thus a random decision. One can see that the initial random exploration phase relatively fast changes to a phase with little software use. Only for the most productive manufacturer agents does it pay off to use software. With few manufacturers demanding software, the market is small and only a few engineering companies actually register positive profit from offering software. The share of software users never becomes zero, so network effects come slowly into play and eventually an increasing number of manufacturer agents can afford to use software, reducing the potential profit that engineering companies can make from offering consultancy. The lack of a zickzack curve for engineering companies originates from that the learning architecture is inherently backward-looking, agents do not generate expectations and react on wrong expectations. Also in this scenario almost 100% of all agents use/provide software after convergence.

Both curves have been generated by models with identical parameter settings. The only difference is the decision making model of the individual agents. One can see that for a stable convergence to occur, the learning architecture needed 1000 episodes, while the process-oriented decision making just needed 50 simulation cycles. This difference comes from that agents in the learning setting learn based on randomly trying out actions; agents in the process-oriented approach predict and only implement the selected option, if it can be done. Learning agents adapt based on past experiences that means they adapt to an environment in which less and less

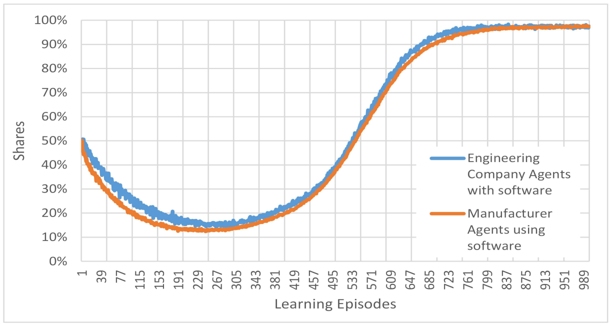

Figure 4: Learning-based decision making: Percentage of manufacturers and engineering companies having switched to using/developing software.

failures happen when the agent select to provide software. The principle is similar, but it leads to the different curves. We also tested other learning parameter settings only affecting the learning speed, not convergence or general shape of the curve.

In the following, we show in two example tests in how far both models behave similarly when particular parameters are changed. Figures 5 and 6 show the dependency on system size. The smallest tested system size consists of 10 manufacturer agents, 3 engineering companies and 100 engineers (factor=1). The given factor reported in the chart is a multiplicand to all system size numbers. Factor 20 corresponds to the baseline scenario used above. Thus, the learning-based approach only works in larger scenarios with a sufficient number of agents.

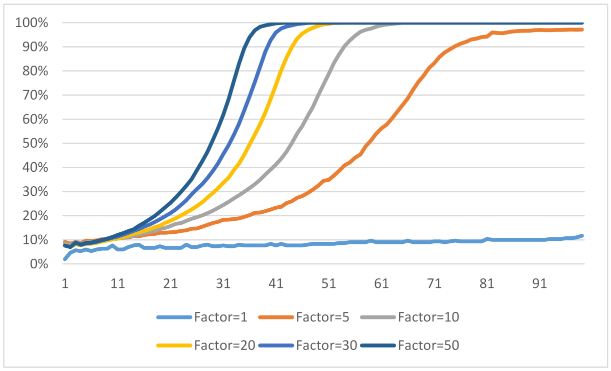

Figure 5: Effect of overall system size on technology uptake in the process-oriented model version

Another set of tests done to compare whether the two agent architectures react differently, concerns the price of a produced unit. All experiments so far have been done with $p = 1$. Raising the price increases the absolute value of the profit in both models. Costs of production do not depend on product price. Differences in the production functions for consultancy versus automation result with higher prices in higher revenues. In both architectures, (see Figure 7 for the process-oriented and Figure 8 for the learning-based approach) technology uptake is faster in scenarios with lower unit prices

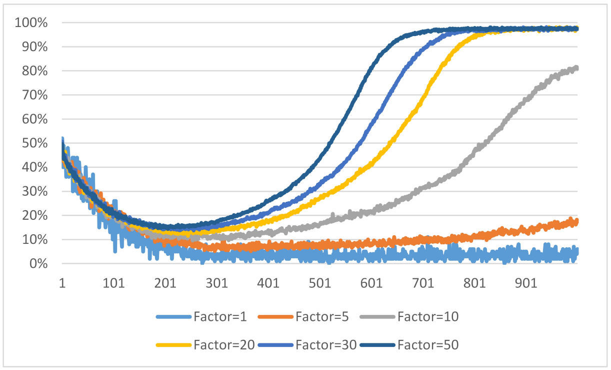

Figure 6: Effect of overall system size on technology uptake in the learning-based model version

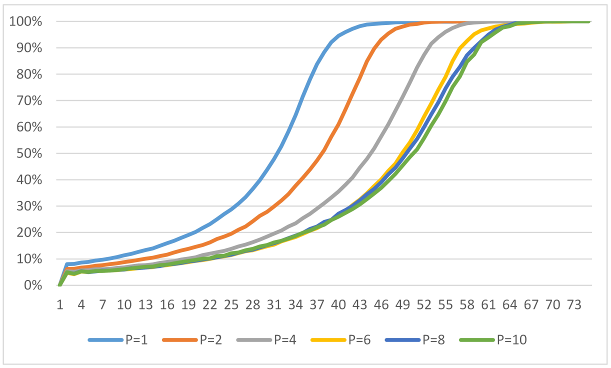

Figure 7: Effect of unit price on technology uptake in the process-oriented model version

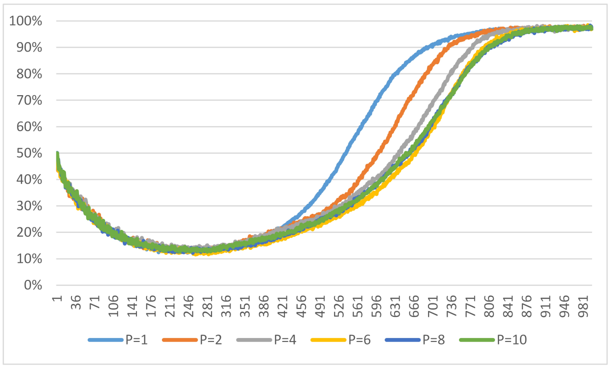

Figure 8: Effect of unit price on technology uptake in the learning-based model version

In the following, we will discuss differences and observations in more detail.

5 DISCUSSION

The experiments show that agent-based simulations with both decision making architectures can produce qualitatively similar results. The curves for manufacturer agents show the same qualitative shape, if some initial drop due to initial exploration is ignored. Differences observed for the engineering companies adaptation originate from that the learning architecture is inherently backward-looking while the process-oriented architecture uses forward-looking. Forward-looking agents form expectations about future demand which become more realistic, the more manufacturer use software. Learning agents adapt based on past experiences that means they adapt to an environment in which less and less failures happen when the agent select to provide software. The principle is similar, but it leads to the different curves. This demonstrates that results are robust and independent from the actual decision making architecture. Beyond that, there are some observations that need to be discussed.

5.1 Simulation Runtime and Handling of Simulated Time

We implemented both models using the same platform using the same underlying basic economic model. We did not use a given Q-Learning library or some other form of generic implementation, but kept the learning architecture as simple as possible, without any overhead. Every agent handles a table with two Q-values. Despite the simplicity, Q-learning does not converge fast. Thus, the duration of simulation experiments was significantly longer than using the direct process-oriented architecture. Unfortunately, we missed to take measurements of runtime using different computers.

While in the process-oriented approach a modeller can establish a correspondence between one simulated decision making round and time intervals at which companies normally take such economic decisions. Due to the explore-exploit approach, this possible correspondence between reality and simulation is impossible using Reinforcement Learning. In reality no company would buy and integrate software or decide to stop consulting business just to test what would be the effect of selecting this action without checking whether an action would actually make sense. The decision for using software is only done, if there is somebody selling software for example. So, a learning episode cannot correspond to for example one “year” of simulated time. As a consequence, statements on how long technology uptake may take absolutely are not feasible with Reinforcement Learning - only relative to other settings. Thus, numbers on the horizontal axis on Figures with process-oriented architecture agents versus on Figures with learning-based architecture agents have no relation.

5.2 Number of Parameters, Calibration and Sensitivity

The underlying model is the same for the two decision making architectures. Agents calculate their profits and costs in exactly the same way. Thus, the number of parameters necessary for the underlying economic model of production, consultancy and software development is the same.

However, there are differences in the number of parameters beyond the basic economic model: There is a specific parameter in the process-oriented architecture that do not appear in the Reinforcement Learning model. On the other hand, the Reinforcement Learning setup comes with parameters for the actual learning.

The specific parameter in the process-based decision making concerns the deliberation of agents when predicting profits while missing experience: Every engineering company predicts its current potential market share for software. This is done using a random selection from a uniform distribution with a parameter for the upper bound of market share. This upper bound is a parameter that has a clear effect on the variation of the zick-zack shape.

Using a Reinforcement Learning based approach comes with a set of learning parameters such as learning rate α, initial ϵ value and decay; these parameter were mentioned above. Another parameter is the number of learning episodes and the lower bound for epsilon applied.

While the learning parameters (except the lower bound for epsilon), had only effects on the convergence rate, the prediction related parameter in the process-oriented setting had dramatic effects: engineering firms were either too optimistic about their potential market share or did not dare to switch at all. Process-oriented models were highly sensitive to the actual value of this parameter. This is typical for parameters impacting the agents' decision making.

Variation of the other parameters - e.g. the size of the overall system (Figure 5 and 6) or the value of the price parameter (Figure 7 and 8) show that both model variants react in a qualitatively similar way to changes in the parameter values.

5.3 Expressiveness

An important question for modelling in general is about expressiveness of the decision making architecture. That means the question, what complexity of reasoning can be formulated. The learning-based approach is directed towards identifying the individually optimal decision. As shown in Section 3, we could explicitly include a lot of the decision making constraints and context into the model. However, preconditions for decisions are not explicitly handled beforehand, but needed to be integrated into the consequences of a decision and eventually into the reward function. This makes the full behaviour model of an agent simpler as the agent not explicitly checks whether prerequisites are fulfilled for coming to a suitable decision that allows production. On the other hand, we had to formulate what costs emerge, if a manufacturer cannot produce - which is kind-of unrealistic. Consequences of actions that impact future costs - such as networking effects - could be integrated into the learning model. So, this shows that a Reinforcement Learning based simulation model does not need to capture decisions in an oversimplified context. Yet, the actual decision is binary, the context determines the actual value of the reward given to the agent.

5.4 Transparency, Explainability, Credibility

Explicitly describing, how the agents decide based on which information, can be clear and is direct. This way of modelling requires an analysis what information each individual agent actually can and does access for making decisions. In contrast to traditional econometric modelling there is no market as “god's hand” that automatically regulates everything so that an equilibrium is reached. So, with direct behaviour formulation, the modeller is in full control, yet has to specify the full agent behaviour.

With the learning settings, the modeller looses this direct formulation. The context of the decision making with its impact on the reward becomes essential. On a technical level, the co-evolutionary learning setting may not be so different from equilibrium searching optimisation approaches. Nevertheless – depending on the sophistication of the part of the overall model that determines the reward, the actual outcome of the learning may be surprising.

Agent-based simulation with individual agents that decide in parallel based on their local context in both settings may challenge explainability of simulation results even without involving learning agents. If there are many agents, it is not trivial to follow how a particular phenomenon on the population level is actually generated from the local decisions and interactions. This is already difficult with direct behaviour modelling, with a learning-based approach one in addition needs to explain why an individual agent has learnt to decide in its specific way. Given that exploration is based on random choices and on changing overall context with more or less competition for software or consultants, each individual history may differ. So, explaining why a particular simulation setting produced a particular result may be even more challenging with agents learning. For credibility of any agent-based simulation model outcome, explanation and justification of every result facet is inevitable [19].

One has to remark, that this particular example model is actually rather benign – in both versions there is a strong global attractor state. Despite of all the random processes involved, different runs of the same setting show only small deviations. Results are clearly sensitive to parameter settings, yet in a way that allows to explain the impact of those parameter values. There is no chaotic model behaviour.

5.5 Relevance of Data

Both presented variants of this model are theoretic abstractions, not case studies of particular clearly defined real-world systems for which relevant data could be collected. So, available, published data is only to inspire typical parameter settings such as for example markup costs for consulting services.

Agent learning is connected to data-driven approaches. So, the learning setup would actually offer more opportunities to integrate data on technology uptake. Agents then do not learn for optimising their profits as in our model, but learn to reproduce given data on how many manufacturer use service automation software. The engineering company agents then could follow the data-driven learning of the manufacturer agents. In this way, the reinforcement learning architecture opens new possibilities to work with data that may be available in the future.

6 SUMMARY, CONCLUSION AND FUTURE WORK

In this paper, we present and analyse two different decision making architectures applied for modelling agents in the same model of technology uptake. Both architectures – a process-oriented direct model of predicting profits and deciding for the more profitable version and a model using an agent architecture in which agents learn Q-values that predict reward (profit) in a stateless Q-Learning setup – actually produced qualitatively similar results. In a rather extensive discussion we compared the different approaches from a model engineering point of view. The result of this discussion is not a clear statement pro or contra one of the approaches, rather a deliberation of consequences when deciding to either use a learning or a direct modelling approach.

In current times of machine learning hype, direct modelling has advantages related to explainability and transparency which is important for credibility of simulation results. Learning may be convenient as details of the decision making do not need to be fixed and fully specified. Using agent learning instead of fully specifying the agent behaviour may have the advantage that overall model outcome does not depend on small details in the behaviour specification, yet in details in the reward calculation.

So, there are advantages and disadvantages in both approaches. A modeller needs to be aware of those. A relevant question is about the generalisability of the analysis done here. The model that we used to base our comparison on exhibits the following generic features:

- Agents decide between discrete options (consultancy versus automation).
- There is a clear concept of reward (profit) for an individual agent depending on the option the agent and other agents chose.
- Decision making is repeated in synchronised rounds.

We assume that any model with such features is possible to be formulated in either process-oriented or learning based way. The features listed above do not refer to Markov properties nor sophistication of the decision making. For being sure about generalisability beyond identifying these rather obvious model features, other models need to be identified and compared in a similar way as we did with the given technology uptake model.

In this regard, it is interesting to look into how a translation from a process-oriented to a learning-based model formulation can be done. The starting point must be the identification of the agents' core decisions plus an analysis the conditions and consequences of such actions. The process-oriented approach consists to a large extend of preparing the core decisions. In a learning-based model, the agents learn about those conditions. Implementing the consequences of decisions, that means performing the actions could be similar in both model versions. The consequences in form of reward require deeper analysis. In the model here, this feedback was straight forward, as with the concept of cost/profit a measure for success was already build in. Nevertheless, before we can formulate a methodology around translation, we need to actually test translation of different models.

Another element of future work is to look into alternative learning architectures. Multi-armed bandits form a learning approach which is similar to stateless Q-learning, yet instead of a Q-value, a probability distribution is learnt for each action (arm). Another alternative that enables the agents to learn a full policy instead of just the suitability of one action is traditional, stateful Q-learning. This brings the difficult question, what does an agent need to know to make a good decision on the table, as this is the information that needs to be captured in the state representation.

ACKNOWLEDGMENTS

The authors would like to thank Magnus Lodefalk and the AI-Econ Lab at Örebro university for an inspiring environment for interdisciplinary research.

REFERENCES

[1] Tina Balke and Nigel Gilbert. 2014. How Do Agents Make Decisions? A Survey. *Journal of Artificial Societies and Social Simulation* 17, 4 (2014), 13. https://doi.org/10.18564/jasss.2687

[2] Stefania Bandini, Sara Manzoni, and Giuseppe Vizzari. 2009. Agent Based Modeling and Simulation: An Informatics Perspective. *Journal of Artificial Societies and Social Simulation* 12, 4 (2009), 4. https://www.jasss.org/12/4/4.html

[3] Eric Bonabeau. 2002. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences* 99, suppl_3 (2002), 7280–7287. https://doi.org/10.1073/pnas.082080899

[4] Mathieu Bourgais, Patrick Taillandier, and Laurent Vercouter. 2020. BEN: An Architecture for the Behavior of Social Agents. *Journal of Artificial Societies and Social Simulation* 23, 4 (2020), 12. https://doi.org/10.18564/jasss.4437

[5] Paul Cummings and Andrew Crooks. 2020. Development of a Hybrid Machine Learning Agent Based Model for Optimization and Interpretability. In *Social, Cultural, and Behavioral Modeling*, Robert Thomson, Halil Bisgin, Christopher Dancy, Ayaz Hyder, and Muhammad Hussain (Eds.). Springer International Publishing, Cham, 151–160.

[6] T. B. F. de Oliveira, Ana L. C. Bazzan, B. C. da Silva, and R. Grunitzki. 2018. Comparing Multi-Armed Bandit Algorithms and Q-learning for Multiagent Action Selection: a Case Study in Route Choice. In *2018 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN.2018.8489655

[7] C. Gloor, L. Mauron, and K. Nagel. 2003. A pedestrian simulation for hiking in the alps. In *Proceedings of the STRC 2003*. https://svn.vsp.tu-berlin.de/repos/public-svn/publications/kn-old/strc03-ped/strc03-ped.pdf

[8] L. Hamill and N. Gilbert. 2016. *Agent-Based Modelling in Economics*. Wiley.

[9] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. 1992. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert* 7, 6 (1992), 34–44.

[10] G. Jäger and D. Reisinger. 2022. Can we replicate real human behavior using artificial neural networks? *Mathematical and Computer Modelling of Dynamical Systems* 28, 1 (2022), 95–109.

[11] Robert Junges and Franziska Klügl. 2012. PROGRAMMING AGENT BEHAVIOR BY LEARNING IN SIMULATION MODELS. *Applied Artificial Intelligence* 26, 4 (2012), 349–375.

[12] J. Källström and F. Heintz. 2019. Tunable Dynamics in Agent-Based Simulation using Multi-Objective Reinforcement Learning. In *Proceedings of the Adaptive and Learning Agents Workshop (ALA 2019) at AAMAS, Montreal, Canada, May 2019*.

[13] F. Klügl. 2009. Agent-oriented Simulation Engineering. https://www.diva-portal.org/smash/get/diva2:910313/FULLTEXT01.pdf

[14] Hildegunn Kyvik Nordås and Franziska Klügl. 2021. Drivers of Automation and Consequences for Jobs in Engineering Services: An Agent-Based Modelling Approach. *Frontiers in Robotics and AI* 8 (2021).

[15] Kamwoo Lee, Mark Rucker, William T. Scherer, Peter A. Beling, Matthew S. Gerber, and Hyojung Kang. 2017. Agent-based model construction using inverse reinforcement learning. In *2017 Winter Simulation Conference (WSC)*. 1264–1275. https://doi.org/10.1109/WSC.2017.8247872

[16] Jörg P. Müller. 1999. Architectures and applications of intelligent agents: A survey. *The Knowledge Engineering Review* 13, 4 (1999), 353–380. https://doi.org/10.1017/S0269888998004020

[17] E. Norling, L. Sonenberg, and R. Rönnquist. 2000. Enhancing Multi-Agent Based Simulation with Human-Like Decision Making Strategies. In *Multi-Agent-Based Simulation, Second International Workshop, MABS 2000 Boston, MA, USA. Revised Papers. (LNCS, 1979)*, S. Moss and P. Davidsson (Eds.).

[18] M. J. North and C. M. Macal. 2007. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modellign and Simulation.* Oxford University Press.

[19] Bhakti Stephan Onggo, Levent Yilmaz, Franziska Klügl, Takao Terano, and Charles M. Macal. 2019. Credible Agent-Based Simulation – An Illusion or Only A Step Away$f$. In *2019 Winter Simulation Conference (WSC)*. 273–284. https://doi.org/10.1109/WSC40007.2019.9004716

[20] S. Russell and P. Norvig. 2021. *Artificial Intelligence: A modern Approach* (4 ed.). MIT Proess.

[21] E. Sert, Y. Bar-Yam, and A. J. Morales. 2020. Segregation dynamics with reinforcement learning and agent based modeling. *Scientific Reports* 10, 11771 (2020). https://doi.org/doi.org/10.1038/s41598-020-68447-8

[22] Peer-Olaf Siebers and Franziska Klügl. 2017. What Software Engineering Has to Offer to Agent-Based Social Simulation. In *Simulating Social Complexity: A Handbook*, Bruce Edmonds and Ruth Meyer (Eds.). Springer International Publishing, Cham, 81–117. https://doi.org/10.1007/978-3-319-66948-9_6

[23] I. Summerville. 2020. *Engineering Software Products: An Introduction to Modern Software Engineering*. Pearson.

[24] R. S. Sutton and A. G. Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[25] P. Taillandier, M. Bourgais, P. Caillou, C. Adam, and B. Gaudou. 2017. A BDI Agent Architecture for the GAMA Modeling and Simulation Platform. In *Multi-Agent Based Simulation XVII*, L G. Nardin and L. Antunes (Eds.). Springer International Publishing, Cham, 3–23.

[26] Keiki Takadama and Hironori Fujita. 2005. Toward Guidelines for Modeling Learning Agents in Multiagent-Based Simulation: Implications from Q-Learning and Sarsa Agents. In *Multi-Agent and Multi-Agent-Based Simulation*, Paul Davidsson, Brian Logan, and Keiki Takadama (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 159–172.

[27] G. Taylor, R. Frederiksen, R. Vane, and E. Waltz. 2004. Agent-based Simulation of Geo-Political Conflict. In *Proceedings of Innovative Applications of Artificial Intelligence, San Jose, CA, July 2004*. 884–891.

[28] C. Urban and B. Schmidt. 2001. PECS - Agent-Based Modeling of Human Behavior. In *AAAI Technical Report FS-01-02*.

[29] M. Wooldridge. 2013. *An Introduction to MultiAgent Systems* (2 ed.). Wiley.