



Article

# Simulating, Off-Chain and On-Chain: Agent-Based Simulations in Cross-Organizational Business Processes <sup>†</sup>

Timotheus Kampik <sup>1,2,\*</sup>  and Amro Najjar <sup>3</sup> 

<sup>1</sup> Signavio GmbH, 10787 Berlin, Germany

<sup>2</sup> Department of Computing Science, Umeå University, 901 87 Umeå, Sweden

<sup>3</sup> AI-Robolab/ICR, Department of Computer Science and Communications, University of Luxembourg, 4365 Esch-sur-Alzette, Luxembourg; amro.najjar@uni.lu

\* Correspondence: tkampik@cs.umu.se

† This paper is an extended version of our previous papers “MAS-Aided Approval for Bypassing Decentralized Processes: An Architecture” and “Integrating Multi-agent Simulations into Enterprise Application Landscapes” presented at the Workshop on Blockchain Technology for Multi-Agent Systems (BTC4MAS) in 2018 and 2019, respectively.

Received: 28 November 2019; Accepted: 6 January 2020; Published: 7 January 2020



**Abstract:** Information systems execute increasingly complex business processes, often across organizations. Blockchain technology has emerged as a potential facilitator of (semi)-autonomous cross-organizational business process execution; in particular, so-called consortium blockchains can be considered as promising enablers in this context, as they do not require the use of cryptocurrency-based blockchain technology, as long as the trusted (authenticated) members of the network are willing to provide computing resources for consensus-finding. However, increased autonomy in the execution of business processes also requires the delegation of business decisions to machines. To support complex decision-making processes by assessing potential future outcomes, agent-based simulations can be considered a useful tool for the autonomous enterprise. In this paper, we explore the intersection of multi-agent simulations and consortium blockchain technology in the context of enterprise applications by devising architectures and technology stacks for both off-chain and on-chain agent-based simulation in the context of blockchain-based business process execution.

**Keywords:** business process execution; multi-agent simulation; blockchain technology

## 1. Introduction

Business process management (BPM) is defined as “the design, execution, monitoring, and improvement of business processes” [1]. BPM-based systems are typically deployed to automate business processes inside the boundaries of the same organization (i.e., intra-organizational processes). Yet, the rise of decentralized process execution across organizational boundaries has raised a new set of challenges: Multiple parties from different organizations are involved in a joint decision-making process, and organizational boundaries exacerbate knowledge inconsistencies, as well as conflicts of interests. Furthermore, even after a decision has been made, maintaining the resulting agreement in the long-run can be challenging given a continuously changing business environment, the potentially large number of parties involved, and the different perceptions of the exact agreement details.

For information technology support of such cross-organizational processes traditional systems rely on a trusted central authority whose establishment and maintenance can be a costly and time-consuming process. The emergent blockchain technology (BCT) has the potential to change the landscape of cross-organizational BPM since BCT can help execute processes that span multiple

organizations without the need of central authority or of mutual trust among the actors involved. In particular, so-called consortium blockchains [2,3] can be considered as promising enablers in this context, as they do not require the use of cryptocurrency-based blockchain technology, as long as the trusted (authenticated) members of the network are willing to provide computing resources for consensus-finding.

However, when such a decentralized (cross-organizational and BCT-based) process is executed across organizational boundaries, (re-)deployments and updates require the explicit consensus of an even wider range of (inter-organizational) stakeholders. Moreover, decentralized processes cannot be expected to be perfectly aligned with the requirements of an individual organization.

In addition, it is important to highlight that BCT alone is insufficient as an enabler of agile cross-organizational integration; additional technologies are necessary to allow for autonomous execution at the organizational interface. One solution to facilitate autonomy in cross-organizational business process execution is to delegate some of the business decisions to machines. In particular, to assist or fully automate complex decisions, agent-based simulations (ABS) can be considered a useful tool for the autonomous enterprise since they allow assessing potential future outcomes.

In this paper we present a set of synergies between BCT and ABS. Hereby, we focus on two scenario types: off-chain ABS and on-chain ABS. In the off-chain case, ABS can be used when, under specific circumstances (e.g., budget shortages), one organization may tend to allow the bypassing of the decentralized process in favor of a locally executed process variant, since doing so may serve the economic interest of the organization (reducing the costs). Hence, ABS can assist the organization to determine whether to allow a bypass of the decentralized and agreed upon process. However, the results of ABS are typically not automatically integrated into the corresponding business process. Instead, the integration is undertaken by human users who are responsible for adjusting the implemented policy to take into account the results of the ABS. These limitations are exacerbated when the results of the ABS affect multi-party agreements (e.g., contracts) since this requires all involved actors to agree on the validity of the simulation, on how and when to take its results into account, and on how to split the losses/gains caused by these changes. To address these challenges, the second scenario type explores the integration of ABS on-chain into enterprise application landscapes. In particular, we present an architecture that integrates ABS into cross-organizational enterprise resource planning (ERP) processes.

The rest of this article is organized as follows: Section 2 offers the background of this work, in the domains of business process management (BPM), block-chain technology (BCT), and agent based simulation (ABS). Section 3 describes the first scenario type, where the ABS simulation is conducted locally, off-chain. Section 4 details the second scenario type, in which the simulation is executed decentralized, on-chain. Finally, Section 5 discusses the results, raises questions for future research, and concludes the paper.

## 2. Background

### 2.1. Business Process Management

Business process management (BPM) is concerned with the design, execution, monitoring, and improvement of business processes [1]. Business process management systems (BPMS) are “generic software system(s)... driven by explicit process designs to enact and manage operational business processes” [4]. BPMS allow for the rapid development of complex process-oriented applications. This is typically achieved by relying on a business process diagram as a starting point, and often by automatically serializing a graphical model and executing it with a business process execution (BPX) engine [5]. The business process model and notation (BPMN) [6] has established itself as a widely adopted open standard that specifies a graphical notation and a data format for business process models.

## 2.2. Blockchain Technology

Blockchain technology (BCT) appeared as a distributed database technology based on a tamper-proof list of timestamped transaction records [1]. Using this technology for cryptocurrencies (e.g., the Bitcoin) was its first wide-spread application [7]. Thanks to a combination of peer-to-peer networks, cryptography and protocols for automated consensus-findings, BCT allows actors to execute transactions over computer networks where none is considered as a trusted party [1].

Blockchain is a chained list of blocks distributed over a peer-to-peer network whose nodes maintain the latest version of the chain. The blocks of such as chain can be used to keep track of any shared valuable data (e.g., financial transactions, medical records, etc.) in a decentralized manner. Each block in the chain has content data, a hash generated from the data, and the hash value of the previous block in the list. This way, hashes guarantee the identification of each block as well as the chaining of these blocks typically based on a chronological order. With this structure in mind, in case one party seeks to modify one block, this results in changing its hash and thereby breaking the chain itself. Yet, adding new blocks to the chain is possible as long as there's a consensus among the peers on the new version of the chain containing the new block [8].

Ethereum [9] represents a blockchain with a built-in Turing-complete programming language. It provides an abstract layer enabling anyone to create their own rules for ownership, formats of transactions, and state transition functions. This is done by involving smart contracts, a set of cryptographic rules that are executed only if certain conditions are met [9,10]. Thus, it can be seen an open software platform based on BCT that enables developers to build and deploy decentralized applications.

### 2.2.1. Consortium Blockchains

Beyond their use for cryptocurrencies, blockchains have the potential to enable interoperable cross-organization collaboration. However, existing public blockchain exhibit some limitations when applied to such industrial use-cases. (i) Data privacy, (ii) transaction volume and scalability, and (iii) the ease of the protocol updateability have been identified as the main obstacles hindering massive blockchain adoption for entrepreneurial applications [3]. Consortium blockchains emerged to overcome these limitations and allow for a efficient and limited-audience blockchains that still benefit from the decentralized mechanisms of public blockchains. Thus, consortium blockchains provide a platform to undertake regulated business, allow for high transaction throughput without fees, and offer better protection against external disturbances [3].

Despite the improvements brought about by the evolution of consortium blockchains, some limitations and pitfalls need to be addressed. These include a hard-coded consensus and inflexible trust model, as well as the requirement that smart contracts must be executed by all the peers which sometimes violates its confidentiality [11]. Hyperledger Fabric has been proposed to overcome these limitations. One of the projects of Hyperledger [12], Fabric proposes a new blockchain architecture designed to be flexible, resilient and scalable. It allows the executions of distributed applications, written in standard programming languages, across several nodes. For these reasons, Fabric has been described as a distributed operating system for consortium blockchains [11].

### 2.2.2. Blockchain and Business Process Execution

Recently, blockchain technology (BCT), and in particular the concept of smart contracts has been identified as a possible solution to allow for decentralized process execution, i.e., when serving a set of untrusting parties, blockchain-based process execution has the potential to make trusted intermediates obsolete, since involved actors can maintain a ledger of authenticated transactions without the need to rely on a central or external authority [1]. In particular, engines that compile business process (business process model and notation, BPMN [6]) and business decision diagrams (decision model and notation, DMN [13]) and execute the resulting program code have been proposed and implemented as prototypes for public blockchain solutions, i.e., for Ethereum [14,15]. While these works aim to

facilitate cross-organizational process-execution, this paper explores the integration of agent-based simulations into the emerging ecosystem of BCT-supported cross-organizational enterprise software. With this integration, information obtained from the simulation can be automatically employed to update cross-organizational business rules, which allows organizations to accommodate the rapid pace of today's business environment.

A key challenge in this context is that decentralized business processes, executed as smart contracts, are technically immutable. Consequently, once the participating actors have agreed upon and deployed a smart contract, replacing it with a new one (e.g., in case the business requirements of one of the parties have changed) is a socio-technical challenge that cannot be solved with BCT itself. To overcome this challenge, organizations participating in smart contract-based business processes may choose to allow the bypass of decentralized processes and opt for a local process variant if the latter better suits their business requirements in a given case.

### *2.3. Agent-Based Simulation for Decision Support*

ABS have been used to simulate a variety of social phenomena and environments, with the objective of providing decision support for decision makers in various domains. The use of agent technology for simulating social phenomena on a computer, a sub-domain known as agent-based social simulation (ABSS) [16], has been applied to different use cases. Examples include evaluating the effects of marketing strategies [17] and assessing the impact of rumors on social networks [18]. In a business context, ABS have been applied in manufacturing systems and supply chain management [19], and to assist software-as-a-service (SaaS) providers strike a balance between business gains and end-user satisfaction [20].

Furthermore, agent-based simulations (ABS) emerged as a powerful support tool, helping decision-makers to cope with complex and changing environments. Examples of successful ABS applications include the development of rural areas [21], the modeling population of displacement and its consequences [22], and crisis and disaster management [23,24].

Moreover, as exemplified by the application of ABS to simulate drone use in smart cities and megapolises [25,26], ABS are expected to play a key role in helping designers and decision-makers assess their policies in different contexts, dynamic environments, and multi-stakeholder systems. This helps detect potential errors and unforeseen problems.

In most of these applications, ABS are used to enable experts to assess the influence of exogenous variables (e.g., global warming), and the impact of potential policies and their consequences (e.g., the consequences of increasing government spending in rural regions).

On the technical side, multiple ABS tools and development environments have been proposed [27,28]. For example, MASON [29], RePast Symphony [30], and Netlogo [31] are frequently used general-purpose and social simulation environments that help users build complex ABS and provide ready-to-use plugins and support functionalities, such as data visualization and analysis tools.

### *2.4. Multi-Agent Systems and Blockchain Technology*

Recent research has highlighted the existence of synergies between multi-agent systems and BCT. On the one hand, BCT can address well-known trust issues in multi-agent systems. On the other hand, MAS can help solve scalability issues that are common in BCT scenarios [32]. Furthermore, the combination of MAS and BCT can provide the necessary capabilities to make the operation of distributed entities more secure, autonomous, flexible, and profitable [32]. Recent works have employed BCT and MAS in applications such as collaborative governance [33] (self-aware agent-supported contract management on blockchains for legal accountability), multi-agent coordination [34], trust, data integrity and reputation management in market places [35], and robot swarm management [36]. However, although the combination of MAS and BCT is receiving increasing attention from the research community, no other works exist that cover the coupling of agent-based simulations and BCT to support complex business decision-making at the organizational interface.

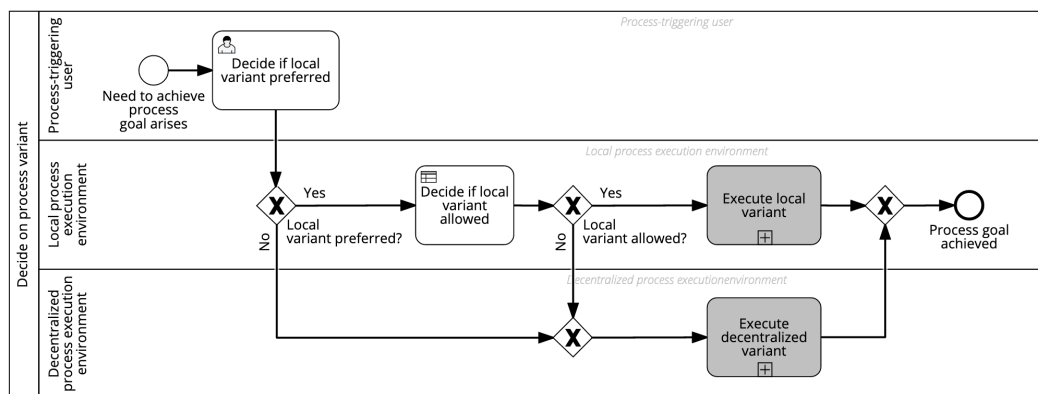
### 3. Simulating Off-Chain

The problem class addressed by this scenario type can be defined as follows. Given an organization that employs a decentralized process running across organizational boundaries, it can be expected that for some cases the process will not match the requirements of this organization particularly well. However, the organization cannot unilaterally update the process since the decentralized process can only be updated if consensus between the corresponding stakeholders of all participating organizations is reached. For this reason, the organization wants to allow for a locally controlled process variant to reach the same business goal in case following the decentralized process is impractical.

#### 3.1. Use Case

Let us introduce an example use case. In this scenario, the process participants who trigger the process need to request approval for bypassing the decentralized process in favor of the local one. However, approving or denying the bypass is a decision with an impact that goes beyond this individual case since it will influence the staff’s opinion about the organization. For example, suppose that the bypass is approved, such a decision will likely encourage other employees to request bypasses more frequently. Consequently, this might lead to too lenient policies within the organization. The opposite is also true, denying too many bypass request, while abstaining from updating the decentralized process, would suggest that the organization is incapable of accommodating the emerging needs of the employees neither by updating the decentralized process, nor through a local bypass.

Our aim is to better assess the consequences of approving/denying a specific bypass request. Figure 1 depicts the overall process—a local and a decentralized process variant, preceded by the case initiator’s decision on whether to request bypassing the decentralized variant and, if a bypass is requested, by an automated decision task to approve or reject the bypass—as a business process model and notation (BPMN) diagram.



**Figure 1.** Business process model and notation (BPMN) diagram of the use case example for off-chain agent-based simulations (ABS): Deciding whether a request to bypass a decentralized process should be granted.

#### 3.2. Architecture

To cope with this problem, we propose to rely on an ABS and integrate it into the process architecture. In particular, the ABS determines the social consequences of approving/denying a bypass on a per-case basis. Then, depending on the outcomes, it adjusts the rules of the decision task accordingly. In this subsection, we describe the architecture of the proposed solution.

Once this overall process is triggered by a human user, a local BPX environment then takes its management. If needed, the BPX environment can call the ABS as a service. In case the decentralized process variant is selected (e.g., the user does not request a bypass or the request is denied as a result



of the ABS recommendation, the decentralized process is executed), this result is handed back to the local BPX environment upon termination. In the BPMN diagram that describes the initial problem (Figure 1), the simulation step can be considered a BPMN service task that precedes the approval task and whose aim is to (Determine whether local variant allowed). This service is called by the local BPX environment to understand the social consequences of a potential approval/denial.

To model our agents we chose the belief–desire–intention (BDI) approach as an example architecture. BDI is frequently used to model humans and create human-like behavior in simulated environments [37]. However, we concede that, depending on the exact use case, simpler agent architectures might be more convenient to employ, depending on the level of granularity that is required for a sufficiently accurate simulation.

The following example shows how a BDI agent can work in our use case. First, a set of agents representing the employees of an organization are instantiated by the ABS. Each agent whose initial reputation is  $r$ , must accomplish a task  $t$  preferably before a deadline  $d$ . For the sake of meeting  $d$ , the agent may request a bypass approval. Nevertheless, in order to preserve its reputation  $r$ , the agent would only ask for bypass in case the reputation loss of not finishing the work in time is higher than the expected combined rejection/missing deadline penalty ( $r$  of the agent can be damaged by losing the deadline and by having a bypass request rejected). Otherwise, the agent simply follows the standard decentralized process, even if this implies not meeting the deadline. Table 1 shows an example of the status of a BDI agent representing an employee at a given time instance  $T$  in the ABS.

**Table 1.** Beliefs, desires and intentions of a belief–desire–intention (BDI) agent representing an employee.

Beliefs	B1	$d$ is the deadline of the task $t$ .
	B2	In case I apply the standard decentralized process on $t$ , I will not meet $d$ .
	B3	My reputation $r$ will be damaged if I do not accomplish task $t$ before $d$ . $p1$ is the penalty subtracted from $r$ .
	B4	Following a simplified local procedure $lp$ would allow me to catch $d$ .
	B5	To execute $lp$ , I can ask to ignore the standard process.
	B6	Other employees in similar circumstances have asked for such a bypass and their requests have been approved. I expect my chances of approval to be $x \in [0, 1]$ .
	B7	If my request is not approved, my reputation would be reduced by a penalty $p2$ .
Desires	D1	I desire to finish the task $t$ , before the deadline $d$ .
	D2	I would rather not ask too many bypass requests that may not be approved.
Intentions	I1	If $p1 \geq (p1 + p2) * (1 - x)$ : Appeal to a local approver and ask to bypass the procedure
	I2	If $p1 < (p1 + p2) * (1 - x)$ : Comply with the standard decentralized process in the first place.

Because the agent is likely to be influenced by the decisions taken by the organization, by the behavior of other worker agents and by and the evolution of the available local and decentralized process, beliefs, desires and intentions listed in the table are subject to change.

The output of the ABS is provided as a process run-time variable to one or multiple rules that inform the approval decision task. For example, the ABS could set the ordinarily scaled process variable `social_spread` to a value  $\in \{none, low, medium, high\}$ , based on the impact the simulated approval decision has in comparison to its rejection counterpart. In addition to the rules that handle these variables, the decision task may contain static rules that can be adjusted during design time, for example when the risk preferences of the process owner change. Listing 1 shows the pseudo-code for an example set of decision rules (note that all variables are ordinarily scaled).

#### Listing 1: Approval decision pseudo-code

```
decline if:
  (social_spread = high
  AND
  business_benefit >= medium)
```

```

OR
  (compliance_violation >= low)
else: Accept

```

BPX environments typically have specialized engines for dealing with decision rules [38].

### 3.3. Components

The architecture consists of the following components:

- a business process repository that contains the process definitions that are to be deployed;
- a service that deploys the process definitions to the local and decentralized process execution engines;
- a local BPX environment that logs the user request and preference (on whether to bypass or not), handles the bypass approval, and calls the decentralized process variant;
- a database that stores the case history and that can be accessed by the ABS to determine the social consequences of granting a bypass approval to a specific user;
- a decentralized BPX environment that is called by the local BPX environment and then runs the decentralized process to finally return the process result back to the local environment;
- an ABS environment that is called as a service from the local BPX engine to help determine whether a bypass request should be approved or not;
- a user database that contains social structures or activities within the organization.

Figure 2 depicts the proposed system architecture.

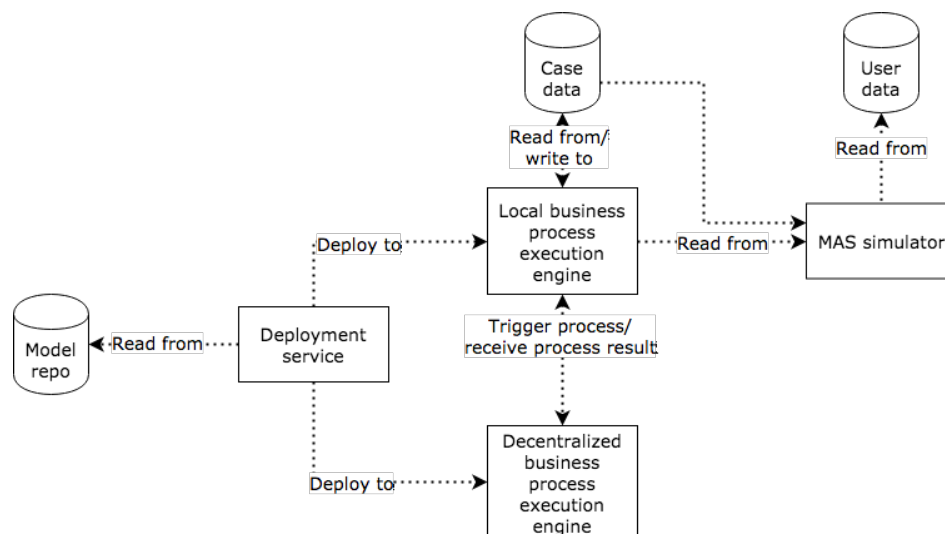


Figure 2. Architecture proposal: Graphical representation.

### 3.4. Technologies

We suggest implementing the architecture with the following technologies:

- **Business process modeling repository: Git-based, BPMN diagrams.** The process definitions can be created and saved in the business process model and notation (BPMN) [6], which is an open standard for a graphical notation, as well as an XML-based data exchange format. BPMN diagrams can be created with a range of different modeling tools [39] (for an overview of BPMN XML-compliant process modeling tools see <http://bpmn-miwg.github.io/bpmn-miwg-tools/>). To model the rules of the bypass approval task, decision model and notation (DMN) diagrams can be used; DMN is an open standard for decision rule modeling that seamlessly integrates with BPMN (see for example Biard et al. [38]). git is an open-source version control system that is commonly used to manage source code versions and hence can be regarded a good fit

to manage the source of the BPMN process definition, which have source code-like properties (for an introduction to git for scientists, see Blischak et al. [40]).

- **Local BPX environment: jBPM.** jBPM is a Java-based business process execution engine [41] that supports the definition and execution of BPMN (and DMN) XML diagrams. As it is open-source and well documented, it is a reasonable choice for implementing the research prototype.
- **Decentralized BPX environment: Hyperledger Fabric.** Hyperledger Fabric (below referred to as Fabric) is a consortium blockchain solution that allows for the definition and execution of smart contracts (in a Fabric context often called chain code [12]. Given that it (i) allows for the implementation of smart contracts in general purpose, high-level programming languages like JavaScript, and (ii) is tailored to enterprise software use cases, with vendors like IBM and SAP being involved in its development, Fabric can be considered a reasonable choice for a consortium blockchain solution in enterprise systems scenarios. It is worth noting that Fabric allows for the execution of Solidity smart contracts, utilizing the Ethereum virtual machine, i.e., it is possible to use the Caterpillar BPX engine in a Fabric context (see <https://tech.signavio.com/2019/caterpillar-hyperledger>).
- **Case data base: MySQL.** Case (process instance) data is stored by jBPM, which supports a range of databases. A reasonable choice for a database is MySQL, a stable and well-established open-source relational database system.
- **Deployment service: Caterpillar/TypeScript-based.** As jBPM integrates with git by default, only the deployments to Fabric need to be custom implemented. The key challenge is to convert smart contract definitions as specified in BPMN XML files into Solidity smart contracts. For this, the Caterpillar engine [14] (an academic research prototype) can be used. To integrate Caterpillar with git, it makes sense to use the JavaScript variant TypeScript, the language in which Caterpillar is implemented. As an alternative to the Caterpillar engine, the BPMN-Sol (<https://github.com/signavio/BPMN-Sol>) compiler can be used. The BPMN-Sol compiler is based on the BPMN-to-Solidity compiler of the Caterpillar engine, but allows for a leaner approach to deploying smart contracts from BPMN diagrams, as it does not require running the whole Caterpillar engine, whose process orchestration components are not necessarily used.
- **ABS environment: Jason with MySQL connector.** Jason [42] is a multi-agent system development and simulation environment that is well-established in multi-agent systems research community. We suggest using Jason, as it has proven itself as a good system for BDI simulations (c.f. [43] and the references therein) and—as a Java-based system—is expected to integrate well with jBPM. Moreover, Jason supports or goal to implement social agents particularly well, as it allows agents to easily receive communications from other agents, and to select “sociably acceptable messages” [42]. As a non-BDI alternative, we propose the Java-based ABS tool Repast [44]. In case lean, Function-as-a-Service deployments or web-based simulations are desirable, the JavaScript agent library JS-son [45] can be used.
- **User data base: MySQL.** To keep complexity low, it makes sense to use the same database system for the user data as for the case data. Hence, it is a reasonable proposal to use MySQL for this database as well. The user database can be filled with data that reflects the directory service structure of the organization in focus, or with aggregates of the communication behavior of the employees from corporate email and messaging applications.

#### 4. Simulating On-Chain

While agent-based simulations have gained momentum within the academic community to the extent that they have been successfully applied to help solve real-world problems (see Section 2), from a practical perspective, ABS are still a niche technology that only scientists and technical experts in academia or research departments can handle. This means that the output a simulation returns exists in its own technological silo. A continuous and automated integration into a production enterprise software landscape is not easily possible. Instead, a manual handover from ABS expert to enterprise application specialist and requires feedback and approval by additional stakeholders, for example by



domain experts and business line managers, i.e., the translation from simulation results to actions that have a real world impact is cumbersome.

Considering the use case type at the organizational interface, the aforementioned problems are even more complicated: Each software update requires the consensus of stakeholders in different organizations, whose interests are more likely to conflict and whose perspective on the problem at hand may be based on inconsistent information. In such scenarios, a handover without a seamless integration between ABS and enterprise application components is likely to further strain the time and budget constraints of information technology projects.

#### 4.1. Use Case

The problem, i.e., the lack of integration of ABS into organizational IT system landscapes, and the complications it implies is illustrated by Figure 3. In the example depicted by the figure, an organization *A* designs the ABS based on the expertise of domain specialists, as well as on structured data that was extracted from other information sources, in particular IT systems. The ABS's output is fed into decision support systems to inform and improve human decision-making. If the decision support system is applied internally, by the same organization *A* that has designed the ABS, its output can be migrated to the local IT ecosystem, e.g., by specifying corresponding decision rules. However, in case the results should be deployed in a cross-organizational setting, for example to automate or speed up decision-making at organizational interfaces, such an update of business rules brings with it the challenge that approval from multiple organizations is required, which can imply a repeated forth-and-back between stakeholders in different organizations.

To address these challenges, we propose an architecture and technology stack for the seamless integration of agent-based simulations into enterprise application ecosystems at the organizational interface. A specific focus of our exploration endeavor is the application of blockchain technology as a facilitator of cross-organizational consensus, i.e., to smoothen the procedure of decision rule updates based on the output of the agent-based simulation. Still, let us highlight that, in large parts, our proposal does not depend on blockchain technology as a necessary dependency of the highlighted approach. Alternatively, other methods for distributed consensus-finding can be applied, or a trusted third party can be installed as the infrastructure provider that hosts the technology at the organizational interface.

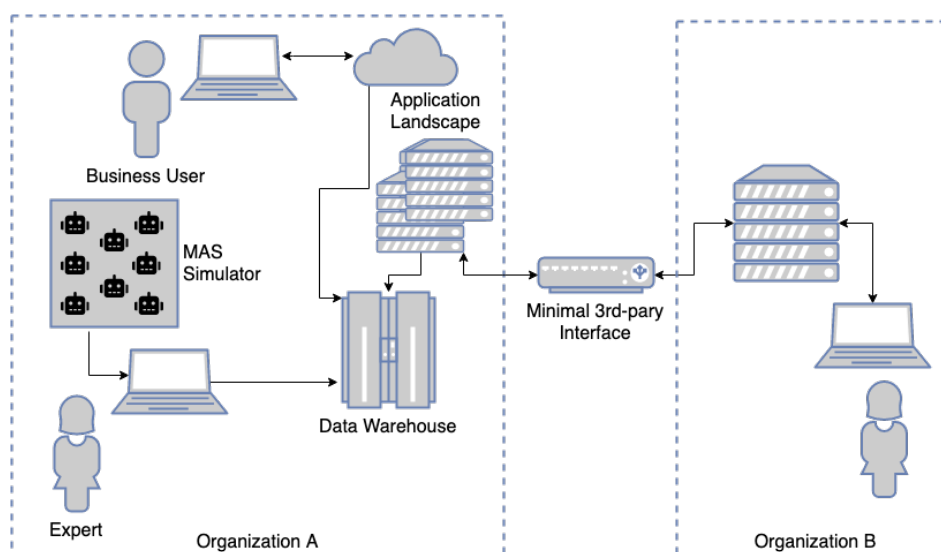


Figure 3. Stand-alone ABS, minimal organizational interface.

As outlined above, an integration of agent-based simulation and enterprise application landscape is necessary to tackle the challenges as introduced above and illustrated in Figure 3. Figure 4 depicts

the suggested approach. As in the traditional approach (Figure 3), we have an organization *A* that wants to integrate an agent-based simulation into its IT ecosystem. However, in contrast to the initial concept that relies on a lean interface for inter-organization exchange, the agent-based simulation is deployed directly at the organizational interface.

The agent-based simulation can be considered a function whose input variables as well as its results are persisted directly at the trusted system that runs at the organizational interface; all participating parties can access the logs and can (semi-)automatically approve updates. This enables faster consensus-finding and makes it possible to continuously integrate simulation results into the cross-organizational process. Reaching a new agreement after an update of the decision-making routine requires little to no human intervention at the involved organizations (*A* and *B*). Such an update can either entail the change of simulation variables, or even edits to the source code of the simulation. Stakeholders at other organizations can approve changes in a semi-automated or fully automated way. The outlined method reduces the time needed to work in updates into the cross-organizational business process and can prevent conflicts between the different organizations' stakeholders, as (a) all changes are transparently persisted to the trusted solution at the organizational interface (e.g., to a blockchain technology solution) and (b) the update process is governed by clear rules and routines that have been agreed upon by all participating organizations.

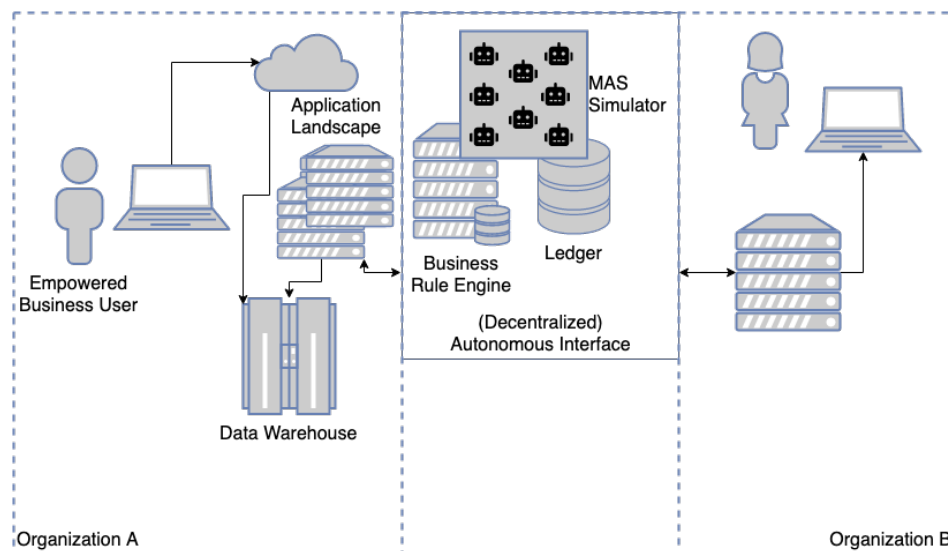


Figure 4. Integrated ABS at organizational interface.

#### 4.2. Architecture

Our architecture proposal that allows for the implementation of the approach we have outlined above is centered on three basic components.

**Decentralized business rule execution (DBREX) environment.** The DBREX system enables the interchange of data, the execution of both agent-based simulations and decision rules, and keeps records of all of these procedures. Thereby execution and record-keeping take place in a decentralized manner, i.e., all participating organizations. However, it is important to highlight that in the context of the use case, the DBREX environment a singleton; it is a distributed system that runs across all organizations that participate in the business process.

**DBREX user interface (UI) application.** The DBREX user interface (UI) application (DBREX UI) provides a business user-friendly abstraction onto the DBREX environment for all participating stakeholders within a specific organization. Business users can configure how their business should automatically interact with the DBREX environment, and interact with it manually if necessary (for example to place new bids). For each participating organization, a separate DBREX UI exists, which can but does not necessarily have to be a custom implementation that is tailored

to the organization's use case and requirements. In many scenarios, it makes sense that the DBREX UI implementation differs depending on the participating stakeholder type; for example, there can be one DBREX UI implementation for suppliers and another implementation for buyers.

**Internal ERP system entry point.** It is of course necessary to connect each organization's enterprise resource planning system to the organizational interface, i.e., via the DBREX UI to the DBREX environment. Considering that most organizations (and in particular large enterprises) have heterogenous and highly customized enterprise system landscapes, each organization will need to implement a custom (or at least customized) entry point.

Figure 5 depicts the proposed high-level architecture in a simplistic two-organization scenario.

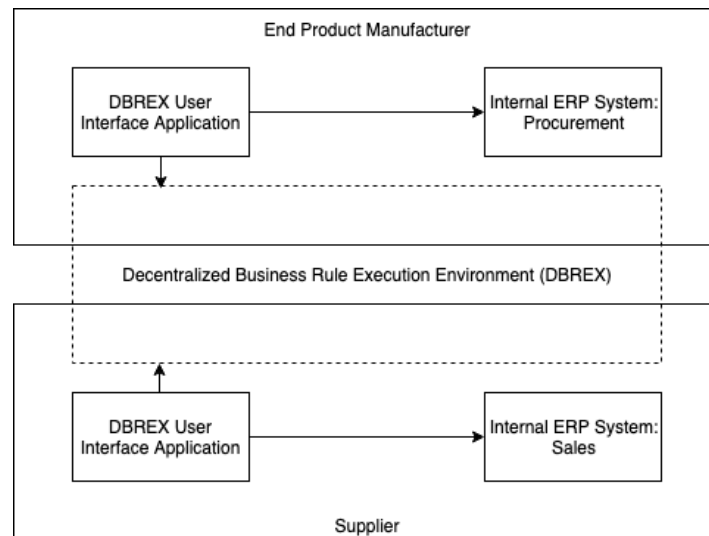


Figure 5. Architecture proposal: Graphical representation.

#### 4.3. Technologies

The following technology stack can be used for an prototypical implementation of the architecture we propose:

**Application execution runtime: Node.js.** Node.js (<https://nodejs.org/>) is a JavaScript execution environment that allows running JavaScript programs outside of the browser (e.g., on a server). Because there are both JavaScript-based ABS tools and smart contract execution environments (see below) and because of its rich ecosystem of plug-and-play dependencies, Node.js is a feasible choice for powering the decentralized business rule execution environment, as well as the front end and back end of the centralized applications that interface with it.

**Decentralized execution environment: Hyperledger Fabric.** Hyperledger Fabric (short: Fabric) is a partial trust smart contract engine [11]. In contrast to public blockchain ecosystems, Fabric is a private, partial trust (consortium) blockchain solution, i.e., participants have to be authorized as eligible users. This has the advantage that a cryptocurrency-based incentive scheme (mining feature) is not required, which improves performance and prevents volatility. Fabric smart contracts can be specified in a range of mainstream programming languages such as Go and JavaScript. In contrast to public blockchain solutions like Ethereum [46], the partial trust approach of Fabric allows for faster and cheaper transactions [47,48].

**Multi-agent simulation library: JS-son.** JS-son is a lean JavaScript-based framework for designing, instantiating, and executing agents of different architecture types. Most notable, JS-son supports the creation of agents with belief-desire-intention (BDI) reasoning loops [45]. Let us note that JS-son does not require a BDI approach; instead, beliefs can activate plans right away.. Because JS-son can be installed as a light-weight Node Package Manager (<https://www.npmjs.com/>) dependency, it can be easily integrated into the Fabric smart contract that implements the

agent-based simulation component. In its ability to integrate seamlessly with the Fabric technology, JS-on distinguishes itself from other agent programming frameworks such as the Java-based Jason platform [49] (for a comprehensive overview of agent platforms, see Kravari and Bassiliades [50]). Moreover, while another JavaScript agent library—Eve [51]—can potentially be considered as an alternative, this library only provides abstractions for agent distribution, which is more little relevance for our use case type, and does not come with reasoning-loop abstractions. Moreover, Eve is at the time of writing unmaintained.

#### 4.4. Example

Supply chain management is considered a promising use case for blockchain technologies (c.f. [52]). In particular, the abilities to track the properties of items across complex delivery networks for quality assurance purposes and to (semi-)automatically adjust an organization’s supplier network are features a decentralized ledger offers as value propositions in supply chain management scenarios. Let us hence consider the following supply chain management example to illustrate the use case type as presented in this section. Our supply chain management process allows a buyer to create invitations to tender (i.e., invitations to supply certain goods). We simply the example for illustrative purposes: We have exactly one purchaser that requests an offer to supply parts and one supplier that agrees to deliver the requested product(s). Moreover, we only cover the happy path, i.e., we ignore possible exceptions (and also low-level details).

The process covers two levels: The initial agreement (human-in-the-loop) level and the automated run-time level. The top-level process describes the supplier management procedure, abstracting from the details of the delivery cycles. The process flow can be described as follows (Figure 6):

1. The process starts with the purchaser issuing an invitation to tender with specific conditions.
2. For the process to continue, a supplier needs to accept the invitation.
3. As soon as a supplier has accepted the invitation, the agreement is persisted to the ledger of the DBREX environment as a smart contract, and the planning and delivery cycle (subprocess) starts. As part of the smart contract, an ABS is specified that informs the dynamic adjustment of the product quantity over the different delivery cycles. Potentially, product quality could be dynamically adjusted as well.
4. When the contract expires, the DBREX environment automatically terminates the supplier relationship.

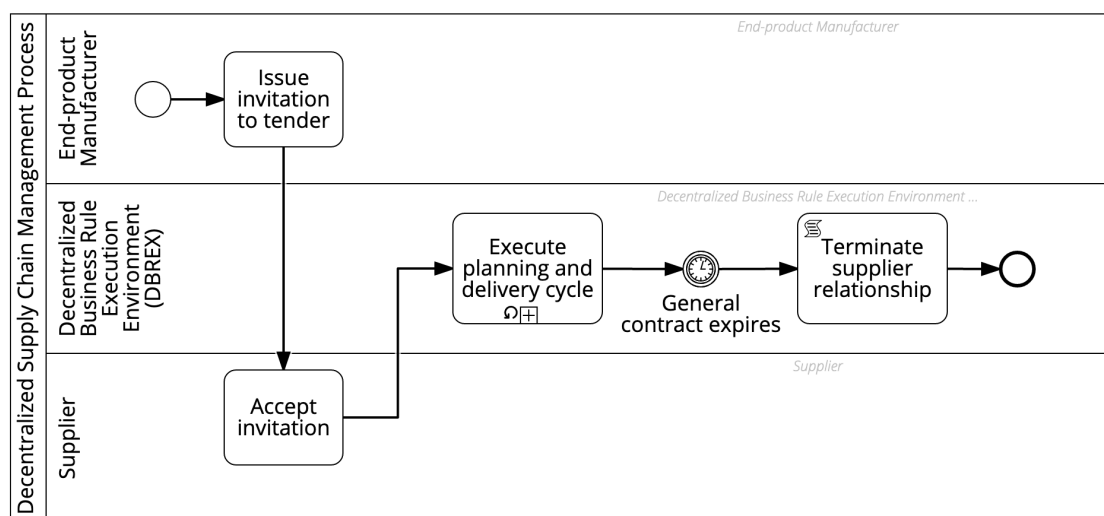


Figure 6. Example: Supply chain management process.

The process flow of a delivery cycle can be described as follows (Figure 7):

1. A delivery cycle starts with the purchaser proposing the current simulation parameters. For example, key indicators that describe the purchaser’s business environment might have been specified as variables in the smart contract’s ABS specification.
2. Then, the supplier decides on the acceptability of the parameters. As an alternative to this step, upper and lower bounds, within which the product quantity needs to be, could be specified in advance.
3. If the purchaser has approved the parameter update, the DBREX environment updates simulation parameters; otherwise, the previously (or initially) specified parameters are kept.
4. Subsequently, the DBREX environment executes the simulation.
5. Based on the simulation result, the DBREX environment adjusts the product quantity that has to be delivered this cycle.
6. Then, the supplier delivers the product in the specified quantity.
7. Once the purchaser receives the project, they register this in the decentralized ledger of the DBREX environment.

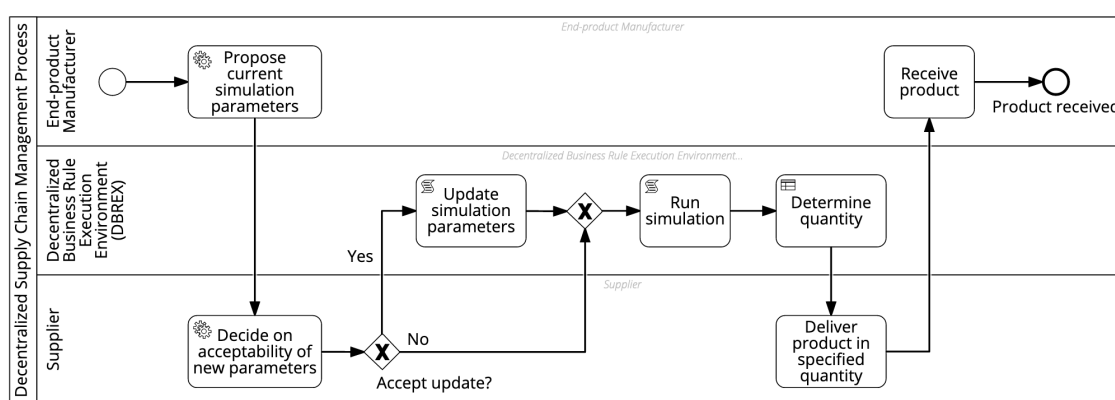


Figure 7. Example: Planning and delivery cycle.

## 5. Discussion

### 5.1. Comparison

This paper provides use case scenarios, architectures and technology stacks for agent-based simulations as facilitators of cross-organizational business process execution. To enable an at-one-glance comparison, Table 2 provides an overview of the key differences between the off-chain and on-chain simulation scenario types.

Table 2. Overview: Key difference between off-chain and on-chain simulation scenario types.

	Off-Chain	On-Chain
Primary use case	Facilitation of intra-organizational agility	Facilitation of cross-organizational autonomy
Distinguishing technology	Any type of chain code	General-purpose programming language for chain code
Key challenge	Accurate digital model of organization (digital twin)	Potential dependence on external data sources (oracles) during execution
Key stakeholders	Internal	External
Integrates with	Internal enterprise systems and blockchain interface	Chaincode, potentially orcale

### 5.2. Key Performance Indicators for Agent-Based Simulations

A key challenge for any organization is to ensure its operations are aligned with overall business objectives. This alignment is typically ensured by defining key performance indicators (KPIs), metrics



that reflect organizational goals and are defined on different levels throughout the organization. The integration of KPIs into business process context is a common practice recommended by industry experts [53] and academics [54] alike, i.e., process-level KPIs are derived from business goals and/or higher-level KPIs.

It can be considered important to monitor whether the agent-based simulations that we propose to integrate into business processes are aligned with process-level and organization-wide KPIs and their underlying business goals. In the off-chain example as presented in Section 3, process level KPIs could be average cycle time and percentage of cases that use the decentralized process variant, while an organization-wide KPI could be employee satisfaction. In the on-chain scenario (presented in Section 4), bottlenecks caused by material shortages could be a process KPI, and time to respond to a shifting market need an organization-wide KPI. To assess whether an agent-based simulation facilitates decision-making that is beneficial to the business environment, the effects of employing the simulations (or specific simulation parameters) have on these KPIs can be measured. Reactions can either be manual—tweaking parameters by hand—or automatic, using methods and algorithms similar to the ones discussed in Section 5.4.

### 5.3. Agent-Based Simulations on Public Blockchains

In Section 4 we provide a proposal for running agent-based simulations on consortium blockchains. In a consortium blockchain scenario, the partial trust nature of the blockchain network allows us to assume that the inherent and emerging complexity of agent-based simulation is relatively unlikely to be used by a smart contract/ABS developer with an information advantage to deceive other participants into committing to a smart contract that will turn out to be misaligned with the presumed use case. It would be naive to make the same assumption in a public blockchain scenario, where it can be assumed that the use case of many stakeholders is primarily speculation, and where the sustainability of the relationship—which would be affected by deceptive actions—plays only a minor role. Hence, let us briefly elaborate on different scenarios, in which agent-based simulations can be deployed to public blockchains and discuss feasibility issues as well as risks.

**Enterprise ABS on public blockchains without open participation.** Considering that public blockchains such as Ethereum are typically associated with slow performance and high operating costs (due to the monetary incentive structure that fuels transactions) [47,48], there are no good reasons for executing cross-organizational processes with ABS-powered automation or decision support that run across a limited number of trusted organization in a public blockchain environment.

**Enterprise ABS with open participation.** A clear disadvantage of consortium blockchains is self-evidently the limitation to a set of trusted parties. In many supply chain management scenarios, creating open exchanges and marketplaces has the potential to create more dynamic and flexible supply chains and decrease the dependency of buyers on specific suppliers. In such cases, agent-based simulations can be employed in an analogous manner as described in Section 4. However, then it will be important to assure that all participants that commit having an ABS influence their organizational routines are fully aware of the implications, e.g., the marketplace could be policed to ensure that no fraudulent/deceptive providers of ABS-enhanced smart contracts exist.

**ABS for different use cases on public blockchains.** Agent-based simulations can be deployed on blockchain technology solutions for other purposes than the use case type we outline in Section 4. Indeed, a broad range of use cases exists [32]. Still, the inherent and/or emerging complexity that agent-based simulations bring with them means that the behavior of ABS-enhanced smart contracts is hard to predict. Participants who are able to predict this behavior accurately, for example because they have designed the ABS, have a clear information advantage that they can exploit. However, the possibility to run ABS in a cryptocurrency context bears societal risks. Evidence exists that deception and fraudulent behavior are rampant on public blockchains,

in particular in the context of cryptocurrencies [55]. Hence, it can be expected that malicious actors will attempt to use the complexity of agent-based simulations as an obfuscation tool to disguise behavior that is unfavorable for other participants. For example, in a smart contract applications that appeals to non-expert, a malicious actor might implement an agent-based simulation that is supposed to resemble a real-world ecosystem—let us assume a national football league simulation with gambling features—but in fact have hidden behavior implemented by design leads to counter-intuitive outcomes (for example: The underdog challenge to the incumbent champion turns out to surprisingly win the league in the "last minute"). This behavior can then of course be exploited by the malicious actor.

#### 5.4. Learning Smart Contracts

It is clear that instead of using multi-agent simulations to facilitate the cross-organizational integration of processes, machine learning (ML) methods can be applied. For example, the assessment whether the approval of a bypass request (as presented in Section 3) should be granted or not can be considered as a classification problem that can be approached with supervised learning methods, given that labelled data exists. If labelled data does not exist, reinforcement learning methods can be applied. However, we argue that, in many cases, ABS are preferable to (deep) machine learning methods, because they can be based on more detailed and explicit (cross-)organizational models, offer a-priori interpretability and do not bring with them the challenge of managing a (deep) learning model as an evolving software artifact. We concede that an obvious advantage of machine learning methods is comparably small modeling effort the application of many ML methods requires; the idea that one can simply "throw data" into an ML algorithm to solve a business problem is certainly enticing. However, the aforementioned disadvantages will in many cases outweigh the convenience edge of machine learning.

Still, when looking beyond standard classification approaches, promising concepts in machine learning—and in particular in reinforcement learning—exist that can be utilized to facilitate flexibility and autonomy at the organizational interface, in particular in a blockchain context.

An example of a particularly useful class of algorithms is the multi-armed bandit concept [56]. Let us provide an informal description of how a simple multi-armed bandit works.

1. We have a set of different options to choose from. Let us assume these options are  $a$  and  $b$ . Each option is a random variable. Choosing a random variable—an arm of the bandit—will return a reward. However, we do not know which random variable has the highest expected value.
2. First, we choose all arms and see what rewards we get. We get 1 for  $a$  and 2 for  $b$ .
3. Given our reward history,  $b$  is the better arm. From now on, we choose the best arm—at the moment  $b$ —with a probability of  $1 - \epsilon$ , given some  $\epsilon$  we have specified, for example  $\epsilon = 0.05$ . With a probability of  $\epsilon$ , we choose a random arm, since we cannot be sure that the arm that is best given our reward history will in fact provide the highest expected value on the long run. Of course, we update the reward history, as well as the expected rewards, each time we make a choice.

Because of the algorithm's property to act greedily with a probability of  $\epsilon$ , i.e., to take the arm with the highest expected reward given the currently available history, this multi-armed bandit version is referred to as an  $\epsilon$ -greedy bandit. Different algorithms that improve this naive approach exist. For example, the  $\epsilon$ , i.e., the probability that the multi-armed bandit takes a random arm, can be gradually reduced over time. An overview of multi-armed bandit algorithms is provided by Kuleshov and Precup [56]. A clear shortcoming of these basic multi-armed bandits algorithm is that the only state they carry is the reward history. For example, they do not take the user context into account. Contextual bandits address this limitation and consider context. Zhou provides a survey on contextual bandits algorithms [57].

To provide an intuition of how multi-armed bandits can facilitate flexibility at the organizational interface in a blockchain (“on chain”) scenario, let us show by example how a simple  $\epsilon$ -greedy multi-armed bandit with decaying  $\epsilon$  can be applied to a supply chain automation problem.

1. We automate the purchase of raw materials on a trading platform. A smart contract specifies how bids are placed. For a specific purpose, we can purchase any of the raw materials  $a$ ,  $b$ , and  $c$ . We want to (i) converge to one raw material on the long run (this will help scaling production) and (ii) buy the raw material for which our purchase offers are responded to in the shortest amount of time.
2. We place initial bids for each material type and document the response times.
3. Then, we proceed iteratively as follows: With probability  $1 - \epsilon$ , we place the bid for the raw material that has the fastest expected response time, based on the data we have accumulated. We gradually decrease our  $\epsilon$  with each bid we place.
4. Eventually, we converge to one raw material, which is likely (but not guaranteed) to be the raw material with the shortest expected purchase time.

Note that his procedure can be autonomously executed as part of a smart contract. Of course, in practice, more advanced multi-armed bandit-based algorithms will be more feasible as they converge faster/cause less regret.

### 5.5. Agents for Enterprise Applications: Technology Maturity

In recent years, the ability of agent-oriented software engineering tools and frameworks to keep up with the pace in which the industrial software ecosystem evolves has been the subject of intense discussion within the agent engineering community [58,59]. Hence, it makes sense to evaluate the ability of agent programming technology to function in a modern enterprise software ecosystem is possible. Thereby, we focus on an analysis of the JS-son library as proposed and implemented by one of the authors of this paper [45], and on three aspects of modern software technology that play a role in an enterprise software context: Web front ends of increasing complexity, function-as-a-service environments, and business process execution engines.

**Web front ends.** Modern web front ends are increasingly complex and powerful, and take over an increasing amount of tasks that traditionally have been handled by server-side programs. Hence, it can be assumed that in some scenarios, agent-based simulations should run as part of web front ends as well (for example in case the ABS in the scenario type as presented in Section 3 should run as a a decision-support system in the browser of the decision-making human). In such scenarios, using a tool like JS-son, as a library that is implemented in an for JavaScript, is a feasible approach.

**Function-as-a-service (serverless).** Function-as-a-service (also called serverless) is the provision of computing services that allows users to deploy web services without having to manage configuration, scaling and maintenance of the underlying infrastructure [60]. Because of its convenience, function-as-a-service components are increasingly prevalent in enterprise application landscapes, in which self-hosted applications are more and more replaced by cloud-base solutions. As many function-as-a-service providers support the deployment of Node.js-based functions and the installation of node package management dependencies, JS-son, which is available as a Node.js package (<https://www.npmjs.com/package/js-son-agent>) is an agent-oriented technology that seamlessly integrates with the function-as-a-service ecosystem.

**Business process execution engines.** Many large organizations use business process orchestration engines to connect and orchestrate a multitude of applications in their enterprise landscape in a process-oriented way. Thereby, the business process execution engine typically calls third-party or in-house developed applications as web services. Given JS-son’s ability to be deployed in Function-as-a-Service (serverless) environments, it is clear that an integration into a service/micro-service ecosystem that is orchestrated by a business process execution engine

is possible. Alternatively, JS-son-based applications can also be wrapped into self-hosted web services. Integration with other traditional enterprise software solutions will—in many cases—be possible in a roughly analogous manner.

Figure 8 displays an example integration of a JS-son-based multi-agent simulation and modern enterprise systems, i.e., a function-as-a-service environment and a software-as-a-service business process execution engine, in the context of the scenario type we describe in Section 3.

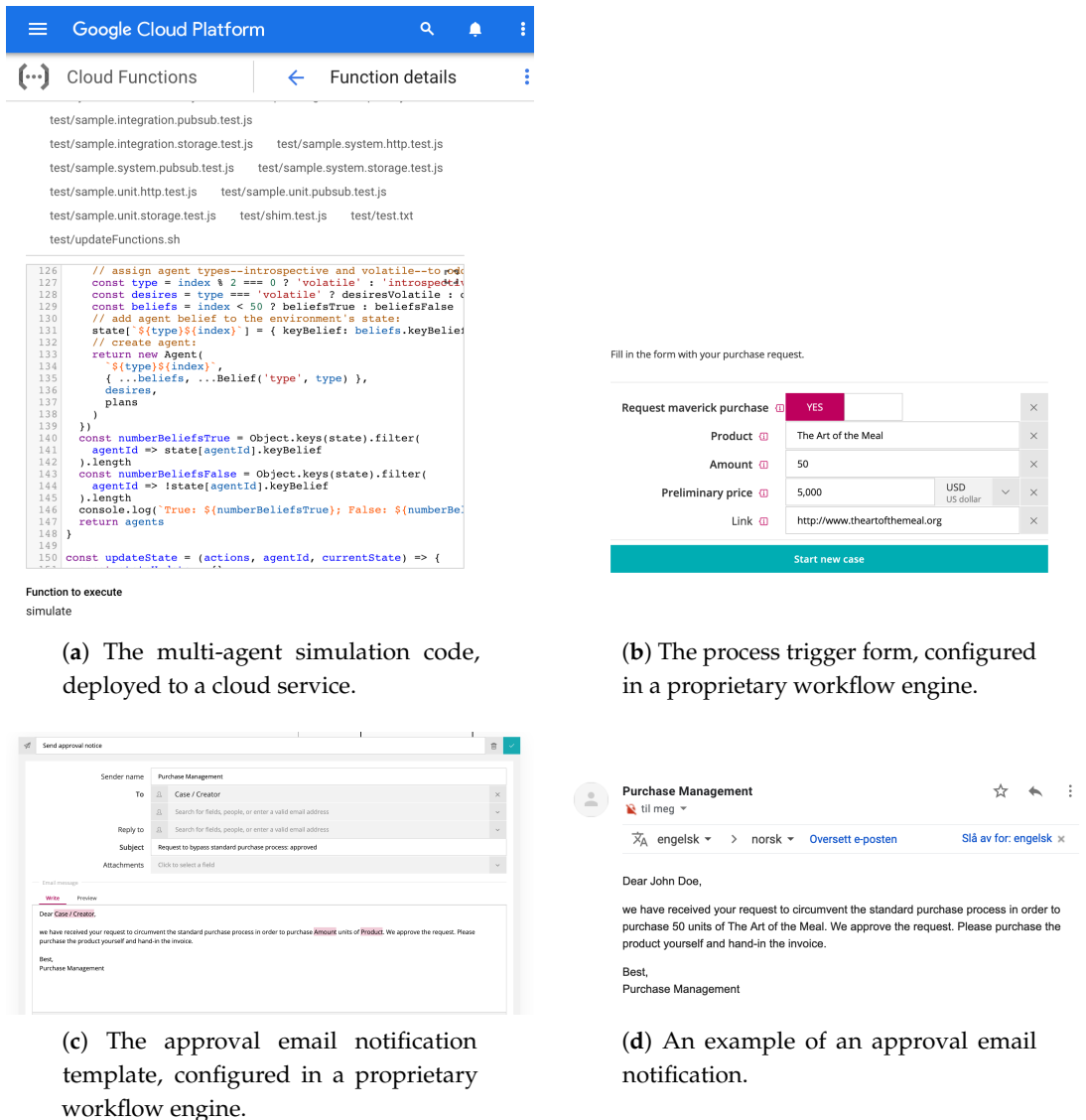


Figure 8. Example configurations, created with proprietary cloud technology.

From an agent programming perspective, it is worth highlighting that alternative approaches are emerging alongside with JS-son. A notable example is jacamo-web [61], which is essentially a web-based, low-code development environment for agents that allows the on-the-fly instantiation of new agents through a graphical user interface. However, it must be conceded that, given the bleeding edge prototype status of the aforementioned frameworks, the establishment of a stable bridge between the agent-oriented programming and enterprise software ecosystems still remains to be established.

### 5.6. Limitations

In this paper, we have presented architectures and tool chains for integrating ABS into enterprise application landscapes at the interface between organizations, in off-chain and on-chain contexts.

The presented work is purely conceptual and lacks an implementation and evaluation. In particular, we want to highlight the following limitations:

- The specifics of how to run the multi-agent simulation that informs the bypass approval decision remain unspecified, primarily because many aspects of this simulation are specific to the organization that applies the solution.
- We have not yet fully implemented the architectures with the proposed technologies. While the technology selection is well motivated, changes to the technology stack might be considered reasonable at the time of implementation.

In addition, from a practical perspective, the future relevance of decentralized, blockchain-based solutions for cross-organizational business process execution is hard to assess, as the impact blockchain will have on enterprise system landscapes is dependent on a range of socio-technical factors. This issue is, however, not central to the contribution of this paper. It is worth highlighting that our architectures are relevant beyond blockchain-based decentralized processes and can be applied for assessing the social consequences of approving the bypass of any process variant. However, applying the architectures to more generic scenarios requires re-thinking the proposed technology stacks.

### 5.7. Future Work

We propose the following future research directions.

**Provide proof-of-concept examples that implement the proposed architectures.** This work proposes two architectures and technology stacks for integrating agent-based simulations into the cross-organizational execution of business processes, with a technology focus on consortium blockchains. A logical next step is the implementation of a full scientific or industrial prototype based on the developed concepts and suggested technology stacks.

**Use a simulation-based approach as a first empirical evaluation of the architectures.** To allow for first empirical evaluations without having to deploy the systems in practice, simulations can be conducted. In the case of the architecture proposal as presented in Section 3, simulations can compare the effect of the ABS-aided bypass approval with the effects automated bypass approvals with a set of static decision rules would have. In an additional comparison, human approval behavior (that can be affected by social pressure) can be simulated. In the case of the on-chain architecture proposed in Section 4, simulations can compare business performance (time, costs) in a traditional environment with the performance that can be achieved by employing the proposed architecture, *ceteris paribus*.

**Assess the architecture proposals in real-world contexts.** To have a practical impact, our architecture needs to be deployed in real-world scenarios. BCT-based business processes execution is to our knowledge not common practice at the time of writing. Hence, we suggest first real-world implementations of the architecture abstract from the smart contract scenario, and instead use alternative solutions that can, for example, be provided by centralized trusted third parties.

**Author Contributions:** Both authors (T.K. and A.N.) contributed to literature review, conceptualization and architecture design, writing, and revision. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and partially funded by the German Federal Ministry of Education and Research (BMBWF) within the Framework Concept "Industrie 4.0—Kollaborationen in dynamischen Wertschöpfungsnetzwerken (InKoWe)"/managed by the Project Management Agency Forschungszentrum Karlsruhe (PTKA).

**Acknowledgments:** We thank the anonymous reviewers for their valuable feedback. Moreover, we thank Gowtham Mohan for sharing his expertise at the intersection of blockchain technologies and business process execution.

**Conflicts of Interest:** The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.



## Abbreviations

The following abbreviations are used in this manuscript:

ABS	Agent-Based Simulation
ABSS	Agent-Based Social Simulation
BCT	BlockChain Technology
BDI	Belief, Desire, Intention
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMS	Business Process Management System
BPX	Business Process eXecution
DBREX	Decentralized Business Rule EXecution
DMN	Decision Model and Notation
ERP	Enterprise Resource Planning
KPI	Key Performance Indicator
ML	Machine Learning
UI	User Interface
SaaS	SaaS Software-as-Service

## References

- Mendling, J.; Weber, I.; Aalst, W.V.D.; Brocke, J.V.; Cabanillas, C.; Daniel, F.; Debois, S.; Ciccio, C.D.; Dumas, M.; Dustdar, S.; et al. Blockchains for business process management-challenges and opportunities. *ACM Trans. Manag. Inf. Syst. TMIS* **2018**, *9*, 4. [[CrossRef](#)]
- Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.
- Dib, O.; Brousmiche, K.L.; Durand, A.; Thea, E.; Hamida, E.B. Consortium blockchains: Overview, applications and challenges. *Int. J. Adv. Telecommun.* **2018**, *11*, 1–2.
- Van Der Aalst, W.M.; Ter Hofstede, A.H.; Weske, M. Business process management: A survey. In Proceedings of the International Conference on Business Process Management, Eindhoven, The Netherlands, 26–27 June 2003; pp. 1–12.
- Li, G.; Muthusamy, V.; Jacobsen, H.A. A distributed service-oriented architecture for business process execution. *ACM Trans. Web TWEB* **2010**, *4*, 2. [[CrossRef](#)]
- OMG. *Business Process Model and Notation (BPMN)*; Version 2.0; OMG: Needham, MA, USA, 2011.
- Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 6 January 2020).
- Drescher, D. *Blockchain Basics: A Non-Technical Introduction in 25 Steps*; APRESS: New York, NY, USA, 2017.
- Buterin, V. A next-generation smart contract and decentralized application platform. *White Pap.* **2014**, *3*, 37.
- Vujičić, D.; Jagodić, D.; Randić, S. Blockchain technology, bitcoin, and Ethereum: A brief overview. In Proceedings of the 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 21–23 March 2018; pp. 1–6.
- Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Heraklion, Greece, 27–30 April 2018; p. 30.
- Hyperledger—Open Source Blockchain Technologies. Available online: <https://www.hyperledger.org/> (accessed on 6 January 2020).
- OMG. *Decision Model and Notation (DMN)*; Version 1.1; OMG: Needham, MA, USA, 2016.
- López-Pintado, O.; García-Bañuelos, L.; Dumas, M.; Weber, I. Caterpillar: A blockchain-based business process management system. In Proceedings of the BPM Demo Track and BPM Dissertation Award Co-Located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, 13 September 2017.

15. Haarmann, S.; Batoulis, K.; Nikaj, A.; Weske, M. DMN Decision Execution on the Ethereum Blockchain. In Proceedings of the International Conference on Advanced Information Systems Engineering, Tallinn, Estonia, 11–15 June 2018; pp. 327–341.
16. Davidsson, P. Agent based social simulation: A computer science view. *J. Artif. Soc. Soc. Simul.* **2002**, *5*, 1–7.
17. Serrano, E.; Iglesias, C.A. Validating viral marketing strategies in Twitter via agent-based social simulation. *Expert Syst. Appl.* **2016**, *50*, 140–150. [[CrossRef](#)]
18. Serrano, E.; Iglesias, C.A.; Garijo, M. A survey of twitter rumor spreading simulations. In *Computational Collective Intelligence*; Springer: Cham, Switzerland, 2015; pp. 113–122.
19. Lee, J.H.; Kim, C.O. Multi-agent systems applications in manufacturing systems and supply chain management: A review paper. *Int. J. Prod. Res.* **2008**, *46*, 233–265. [[CrossRef](#)]
20. Najjar, A.; Mualla, Y.; Boissier, O.; Picard, G. AQUAMan: QoE-driven cost-aware mechanism for SaaS acceptability rate adaptation. In Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, 23–26 August 2017; pp. 331–339.
21. Berger, T.; Schreinemachers, P.; Woelcke, J. Multi-agent simulation for the targeting of development policies in less-favored areas. *Agric. Syst.* **2006**, *88*, 28–43. [[CrossRef](#)]
22. Sokolowski, J.A.; Banks, C.M.; Hayes, R.L. Modeling population displacement in the Syrian city of Aleppo. In Proceedings of the Winter Simulation Conference 2014, Savannah, GA, USA, 7–10 December 2014; pp. 252–263.
23. Bañgate, J.; Dugdale, J.; Beck, E.; Adam, C. A Multi-agent System Approach in Evaluating Human Spatio-temporal Vulnerability to Seismic Risk using Social Attachment. *WIT Trans. Eng. Sci.* **2018**, *121*, 47–58.
24. Ogie, R.; Adam, C.; Perez, P. A review of structural approach to flood management in coastal megacities of developing nations: Current research and future directions. *J. Environ. Plan. Manag.* **2019**, *63*, 1–21. [[CrossRef](#)]
25. Mualla, Y.; Najjar, A.; Daoud, A.; Galland, S.; Nicolle, C.; Yasar, A.U.H.; Shakshuki, E. Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions. *Future Gener. Comput. Syst.* **2019**, *100*, 344–364. [[CrossRef](#)]
26. Mualla, Y.; Najjar, A.; Galland, S.; Nicolle, C.; Haman Tchappi, I.; Yasar, A.U.H.; Främbling, K. Between the megalopolis and the deep blue sky: Challenges of transport with UAVs in future smart cities. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, Montreal, QC, Canada, 17 May 2019; pp. 1649–1653.
27. Abar, S.; Theodoropoulos, G.K.; Lemarini, P.; O’Hare, G.M. Agent based modelling and simulation tools: A review of the state-of-art software. *Comput. Sci. Rev.* **2017**, *24*, 13–33. [[CrossRef](#)]
28. Mualla, Y.; Bai, W.; Galland, S.; Nicolle, C. Comparison of agent-based simulation frameworks for unmanned aerial transportation applications. *Procedia Comput. Sci.* **2018**, *130*, 791–796. [[CrossRef](#)]
29. Luke, S.; Cioffi-Revilla, C.; Panait, L.; Sullivan, K.; Balan, G. Mason: A multiagent simulation environment. *Simulation* **2005**, *81*, 517–527. [[CrossRef](#)]
30. North, M.J.; Collier, N.T.; Ozik, J.; Tatara, E.R.; Macal, C.M.; Bragen, M.; Sydelko, P. Complex adaptive systems modeling with Repast Symphony. *Complex Adapt. Syst. Model.* **2013**, *1*, 3. [[CrossRef](#)]
31. Sklar, E. NetLogo, a multi-agent simulation environment. *Artif. Life* **2007**, *13*, 303–311. [[CrossRef](#)]
32. Calvaresi, D.; Dubovitskaya, A.; Calbimonte, J.P.; Taveter, K.; Schumacher, M. Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Toledo, Spain, 20–22 June 2018; pp. 110–126.
33. Norta, A.; Vedeshin, A.; Rand, H.; Tobies, S.; Rull, A.; Poola, M.; Rull, T. Self-Aware Agent-Supported Contract Management on Blockchains for Legal Accountability. 2017; p. 89. Available online: [http://whitepaper.agrello.org/Agrello\\_Self-Aware\\_Whitepaper.pdf](http://whitepaper.agrello.org/Agrello_Self-Aware_Whitepaper.pdf) (accessed on 6 January 2020).
34. Mariani, S.; Andrea, O.; Giovanni, C. Novel Opportunities for Tuple-based Coordination: XPath, the Blockchain, and Stream Processing. In Proceedings of the 18th Workshop “From Objects to Agents”, Scilla, Italy, 15–17 June 2017; Volume 1867, pp. 61–64.
35. Bonino, D.; Vergori, P. Agent marketplaces and deep learning in enterprises: The composition project. In Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 4–8 July 2017; Volume 1, pp. 749–754.
36. Ferrer, E.C. The blockchain: A new framework for robotic swarm systems. In Proceedings of the Future Technologies Conference, Vancouver, BC, Canada, 15–16 November 2018; pp. 1037–1058.

37. Norling, E. Folk psychology for human modelling: Extending the BDI paradigm. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems 2004, New York, NY, USA, 19–23 July 2004; pp. 202–209.
38. Biard, T.; Le Mauff, A.; Bigand, M.; Bourey, J.P. Separation of decision modeling from business process modeling using new “Decision Model and Notation” (DMN) for automating operational decision-making. In Proceedings of the Working Conference on Virtual Enterprises, Albi, France, 5–7 October 2015; pp. 489–496.
39. Kurz, M. BPMN Model Interchange: The Quest for Interoperability. In Proceedings of the 8th International Conference on Subject-oriented Business Process Management, Erlangen, Germany, 7–8 April, 2016; ACM: New York, NY, USA, 2016; pp. 6:1–6:10.
40. Blischak, J.D.; Davenport, E.R.; Wilson, G. A quick introduction to version control with Git and GitHub. *PLoS Comput. Biol.* **2016**, *12*, e1004668. [[CrossRef](#)]
41. Cumberlidge, M. *Business Process Management with JBoss jBPM*; Packt Publishing Ltd.: Birmingham, UK, 2007.
42. Bordini, R.H.; Hübner, J.F. BDI agent programming in AgentSpeak using Jason. In Proceedings of the International Workshop on Computational Logic in Multi-Agent Systems, London, UK, 27–29 June 2005; pp. 143–164.
43. Adam, C.; Gaudou, B. BDI agents in social simulations: A survey. *Knowl. Eng. Rev.* **2016**, *31*, 207–238. [[CrossRef](#)]
44. Collier, N. Repast: An extensible framework for agent simulation. *Univ. Chic. Soc. Sci. Res.* **2003**, *36*, 2003.
45. Kampik, T.; Nieves, J.C. JS-son—A Minimalistic JavaScript BDI Agent Library. In Proceedings of the 7th International Workshop on Engineering Multi-Agent Systems (EMAS 2019), Montreal, QC, Canada, 13–14 May 2019.
46. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
47. Rimba, P.; Tran, A.B.; Weber, I.; Staples, M.; Ponomarev, A.; Xu, X. Comparing blockchain and cloud services for business process execution. In Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA), Gothenburg, Sweden, 3–7 April 2017; pp. 257–260.
48. Pongnumkul, S.; Siripanpornchana, C.; Thajchayapong, S. Performance analysis of private blockchain platforms in varying workloads. In Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–6.
49. Bordini, R.H.; Hübner, J.F.; Wooldridge, M. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007.
50. Kravari, K.; Bassiliades, N. A survey of agent platforms. *J. Artif. Soc. Soc. Simul.* **2015**, *18*, 11. [[CrossRef](#)]
51. De Jong, J.; Stellingwerff, L.; Pazienza, G.E. Eve: A novel open-source web-based agent platform. In Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, UK, 13–16 October 2013; pp. 1537–1541.
52. Korpela, K.; Hallikas, J.; Dahlberg, T. Digital supply chain transformation toward blockchain integration. In Proceedings of the 50th Hawaii International Conference on System Sciences, Hilton Waikoloa Village, HI, USA, 29 December 2017.
53. Silver, B. *BPMN Method and Style*; Cody-Cassidy Press: Aptos, CA, USA, 2009.
54. Wetzstein, B.; Ma, Z.; Leymann, F. Towards Measuring Key Performance Indicators of Semantic Business Processes. In *Business Information Systems*; Abramowicz, W., Fensel, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 227–238.
55. Drozd, O.; Lazur, Y.; Serbin, R. Theoretical and legal perspective on certain types of legal liability in cryptocurrency relations. *Balt. J. Econ. Stud.* **2018**, *3*, 221–228. [[CrossRef](#)]
56. Kuleshov, V.; Precup, D. Algorithms for multi-armed bandit problems. *arXiv* **2014**, arXiv:1402.6028.
57. Zhou, L. A survey on contextual multi-armed bandits. *arXiv* **2015**, arXiv:1508.03326.
58. Logan, B. An agent programming manifesto. *Int. J. Agent-Orientated Softw. Eng.* **2017**, *22*, 253–269.
59. Mascardi, V.; Weyns, D.; Ricci, A.; Earle, C.B.; Casals, A.; Challenger, M.; Chopra, A.; Ciorte, A.; Dennis, L.A.; Díaz, Á.F.; et al. Engineering Multi-Agent Systems: State of Affairs and the Road Ahead. *SIGSOFT Eng. Notes SEN* **2019**, *44*, 18–28. [[CrossRef](#)]

60. Baldini, I.; Castro, P.; Chang, K.; Cheng, P.; Fink, S.; Ishakian, V.; Mitchell, N.; Muthusamy, V.; Rabbah, R.; Slominski, A.; et al. Serverless Computing: Current Trends and Open Problems. In *Research Advances in Cloud Computing*; Chaudhary, S., Somani, G., Buyya, R., Eds.; Springer: Singapore, 2017; pp. 1–20. [\[CrossRef\]](#)
61. Amaral, C.J.; Hübner, J.F. Jacamo-web is on the fly: An interactive Multi-Agent System IDE. In Proceedings of the EMAS 2019—7th International Workshop on Engineering Multi-Agent System, Montreal, QC, Canada, 13–14 May 2019.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).