Contract-based Data-Driven Decision Making in Federated Data Ecosystems

Wolfgang Maass[1,2]
[1]German Research Center for Artificial Intelligence (DFKI), 66123 Saarbrücken, Germany
[2]Saarland University, Saarland Informatics Campus, 66123 Saarbrücken, Germany
wolfgang.maass@dfki.de

Abstract

The data-driven economy on the World Wide Web are based on coordination mechanisms for the exchange of and AI-based processing of data sets created by independent actors. Architectures for data-driven economic systems currently focus on the exchange of datasets and leave the processing of datasets to background mechanisms. Without binding commitments, agents in data sharing situations favor the "no data sharing" strategy. We present a broker framework that facilitates contract-based exchange, trading, and processing of data sets between agents under conditions of "lack of trust". Electronic contracts guarantee ownership and control of data datasets, the execution of defined data analysis tasks based on AI models, and the sharing of results according to contractual commitments. A technical architecture (TUCANA) is presented that provides a federated data economic ecosystem including a broker framework. We present an application based on an implementation of TUCANA.

1. Introduction

Management is decision-making under uncertainty [1]. Data-driven decision making tries to reduce uncertainties and, thus, complements managerial practice and increases firm performances [2, 3]. Hence, firms are longing for internal and external high-quality data that enables data-driven decision making. Development of capabilities and infrastructures for collecting internal data is often expensive so that companies reach out for external data. For instance, data-driven optimization of supply chains asks for data-sharing between participants of the supply chain [4]. Proprietary organization of data-sharing along supply chains require strong principals who can enforce data-sharing or adding financial incentives [5]. Firms differ in their ability of collecting data that leads to different data sharing strategies. Firms with advanced data collection capabilities have a comparative advantage in data favoring data sales business while other firms need to develop comparative advantages in high-quality goods production [6].

Data is a key driver for Artificial Intelligence (AI) in general, and machine learning in particular [7]. If data is the new oil of the 21st century, the question arises under which conditions business actors are willing to share data (analog to data sharing in research [8]). In commercial contexts, companies are reluctant to share data with third parties because of fear that company secrets are revealed [9]. However, without data from partners, competitors and other third parties, the performance of AI models is limited to predictions based on local data. Because data is currently not protected beyond contract law, data sharing partners need contracts for controlling input data sharing, data processing and sharing results. In situations without prior trust between data sharing partners ('lack-of-trust' condition), the role of a broker as a trusted-third party is introduced for establishing and enforcing binding contracts between data sharing parties. A broker is either embedded into an infrastructure, e.g., blockchains with smart contracts, or established as an independent party, e.g., by data economic platforms, such as GAIA-X. In this paper, we present a broker framework that can be implemented by both approaches while we introduce an implementation for an independent web-based broker.

2. Related work

Internal data directly related to customers, sales and marketing has been core to business intelligence for decades [10]. Recently, also raw data is not only considered as minor side-effect of company outputs but is used for becoming more efficient and even more responsive to market changes and disruptions [2]. For firms, data is input for AI models that learn patterns used for making predictions transformed into recommendations for managerial decisions. This led to

URI: https://hdl.handle.net/10125/79584
978-0-9981331-5-7

a novel understanding of management decision making under the umbrella of *data-driven decision making* [7]. The quality of recommendations and predictions used for data-driven decision making strongly depends on the quality of selected AI models and on the amount and quality of input data [11]. For companies with little IT capacity, generation of data is challenging and requires initial development of IT and data science capital stock [12].

Beside fear and missing intentions, data sharing between firms requires secure and trust-worthy technical platforms. Data-sharing is supported by various forms of platforms and underlying business models and organizational models, respectively. Internet companies running large platforms favor hub-and-spoke platforms with strong control by platform providers while governments, such as Germany and France, look into federated data ecosystems with no dominant platform provider in the middle [13].

A key element of data economic platforms is handling of contracts between data-sharing agents under 'lack-of-trust' conditions. Contracts are binding promises between commercial agents and requirements on data and models to be shared. If trust between agents is not guaranteed, trusted third parties are introduced for enforcing promises, in particular obligations, permissions, and sanctions. Contracts are usually supervised and sometimes even executed by a trustee. Traditionally, trustees are independent actors or even embedded into the data sharing platform as it has been recently introduced by block-chain architecture and smart contracts [14].

2.1. Data sharing

Business actors are generally reluctant to engage in data sharing because of various fears, such as unauthorized access to data, regulatory risks and lack of competency in monetization of data (IW Kurzbericht 24 04 / 2021, IW Trends 01 / 2021). But they are interesting in getting access to external data, including from partners, competitors, markets, research, and general public. This is exemplified by a two-player data-sharing game. In contrast to sharing media files [15], sharing of self-created data comes with a fear of appropriation loss that the receiving agent will leverage shared data without proper control by the originating agent.

For instance, if a player receives data from a providing player, it will create a potential benefit without costs (benefit: 5) but causes a loss for the providing agent (loss: -2) (cf. Table 1). When both players share their data, they receive a benefit but also have a perceived loss (total payoff: benefit-loss, i.e. $5-2=3$). Without additional mechanisms, both agents will adopt "no sharing" as their dominant strategy [15]. This situation favors decision makers to wait for the others to share their data.

		B	B
		No Sharing	Sharing
A	No Sharing	**0, 0**	5, -2
A	Sharing	-2, 5	3, 3

Table 1. Two-player data-sharing game

This situation changes if both player A provides data and player B generates higher-order data as result of processing data back to player A, e.g., access to raw production machine data of player A in return for maintenance predictions by player B. Therefore, agent A perceives returns as being higher than the value of the provided raw data, leading to a dominant "sharing" strategy (cell sharing/sharing: (7,5)). Such conditions favor that one player (here player B) receives a dominant position (principal) with a dependent player A, as typical for hub-and-spoke business models of Internet giants, such as Google and Facebook [16]. This binary ties between two agents results in data sharing systems that favor central principals with many agents and a hub-and-spoke business model.

An alternative game design is given with both players potentially gaining higher returns from sharing at the same time (cell sharing/sharing: (7,7)). This occurs if no loss of input data but higher quality output data for each player is guaranteed. Such a game design is supported by introducing contracts and trustees that mediate in situation with 'lack of trust' conditions.

2.2. Data contracts and trustees

Due to the intangible nature, contractual relations on data are more complicated than for tangible assets [17]. Data can be seen as a commodity, encapsulated by a data-based service or as a data as a service itself [17]. So far, no specific rules exist for data contracts but standard contracts can be used for lowering transaction costs when selling data [17]. We follow Fried's understanding of contracts as promises of parties involved in economic transactions [18]. In cases that provided data is not compliant with contractual promises, mechanisms are required for rolling back transactions. This is challenging because data assets have been already exchanged and leaving the seller side without control. This problem is often solved by introducing trustees as a trust-third party so that data and models are only shared if all contractual obligations are fulfilled.

Formal trustees only evaluate data against contractual obligations. If evaluations are positive, formal trustees grant access on data to receiving agents in the sense of *data as a service*. Data processing is not under the control of formal trustees. *Active trustees* extend formal trustees by applying specified algorithms, e.g., training and application of AI models, and evaluating results against requirements given by contractual promises. Thus, active trustees control execution of data analytical services and sharing of results according to contractual promises. Formal trustees are light-weight agents that only evaluate input data and models and provide no guarantees on results. Active trustees are encapsulated agents that evaluate. for instance, compliance of data, AI models and results to contractual obligations.

Contracts as legal documents require declarative representations that can be scrutinized by legal experts. An e-contracting process consists of four phases, i.e., the information, pre-contracting, contracting, and enactment phases [19].

Several declarative languages for general-purpose contracts have been proposed [20, 21, 22]. The role of contracts for building trust in distributed multi-agent systems for e-commerce have been emphasized [23]. Trusted-thrid parties (trustees) are introduced as norm-enforcing entities [24]. Logical formalizations of contracts are proposed but lack computational infrastructures for contract execution [25, 22]. Contract models represented by business process models, such as BPML, lack semantics for process execution but can be easily deployed on process platforms [26]. Contracts with a focus on financial transactions are focused by *smart contracts* used in blockchain architectures [27]. Level-3 type distributed ledger systems introduced trustees called contract managers [28].

Lee was first in proposing a formalization of contracts by first-order predicate logic [29]. More practical approaches use business models or state-transition models as underlying formalism [30]. Ladleif and Weske propose an ontology for contracts with actions, data sources and legal state as key elements [31]. Updates on actions change legal states while legal states can enable actions. Contract models based on formal logic [29] or ontologies only [31] are theoretically interesting while practical applications are based on state-transition models, process models or rule-based systems (e.g., [32]) while recently implementations of finite-state representations of contracts on blockchain platforms are proposed (e.g.,) [30]).

In this paper, we will focus on the conceptual model of contracts that are negotiated during contracting and used in enactment phases. We follow previous approaches, that understand electronic contracts as event-condition-action programs executed on behalf of market participants [33]. Based on ontological models for contracts [31, 34], we present a state-transition based contract framework for data sharing, data trading and data processing in federated data ecosystems.

3. Broker framework

In the following, we describe a framework that supports data contracts between actors based on software agents and smart services. This provides a minimum of autonomy, control on local data and data processing and communication capabilities for interacting with other agents [35]. Coordination between agents are based on declarative contracts that state which data processing activities are executed on defined data. Contracts define routing and ownership of data, models and software. Agents can apply smart services that share data, train and apply AI models and share results.

These requirements indicate strong resemblance with multi-agent systems but focus more on execution of AI models and on contracts that are used for coordination of data analytical processing tasks. This extends research on MAS that has focused on coordination and communication protocols with little emphasis on the tasks to be executed by agents.

In the following, we will introduce a broker design pattern for data sharing, data trading and data processing in federated data ecosystems. Beside market participants, a broker is introduced as a trust-third party that executes contracts on behalf of market participants. The contract specifies data to be shared, data analytical procedures to be applied on data, handling results, data and models after data analysis.

Based on the smart contract ontology [35], we developed a *broker design pattern* consisting of five elements:

1. *Data sharing agent*: an agent that is in control of data

2. *Broker agent*: an agent that can execute contracts and run data analytics services

3. *Contract*: declarative representation of legal promises, i.e. obligations, permissions and sanctions that can be interpreted by a broker agent based on a legal state

4. *Data Analytical Service*: description of a service that processes data; part of contracts

5. *Data* (input and output/results): data shared by agents as input for data analytical services and output of such processing

Two contract types are distinguished: data analytical contracts and service contracts. A data analytical service specifies data, data analytics and output handling. A service contract specifies promises on service implementations.

Definition 3.1 (Contract)**.** A contract $c_d \in C$ is a model that is specified by actors, contractual obligations, permissions, data analytical functions with service contracts and input and output data. A contract is defined as follows:

$$c_d =< n_c, \mathbf{P}, \mathbf{O}, \mathbf{R}, \mathbf{S}, \lambda(\mathbf{X}), \mathbf{Q} >$$

where n_c is the name of the contract, $\mathbf{P}$ is the set of participants of this contract beyond broker b, $\mathbf{O}$, $\mathbf{R}$, and $\mathbf{S}$ are the set of obligations, permission rights, and sanctions, λ is the data analytical function that is to be applied on input data $\mathbf{X}$, and $\mathbf{Q}$ is a set of quality requirements. Data $\mathbf{X}$ is the integration of data $\mathbf{X}(\mathbf{p})$ provided by participant $p \in \mathbf{P}$.

Definition 3.2 (Obligation)**.** An obligation o_i is specified by an actor, state conditions and actions:

$$o_i =< n_o, a, \mathbf{CO^o_t}, \mathbf{A}, \mathbf{CO^o_{t+1}} >$$

An obligation defines a *mandatory* action set $\mathbf{A}$ performed by actor a if the contract state is fulfilled $\mathbf{CO^o_t}$ at time point t resulting in a contract state that fulfills a set of state conditions $\mathbf{CO^o_{t+1}}$ at time $t+1$.

Definition 3.3 (Permission)**.** A permission p_i specifies an actor, state conditions and actions:

$$p_i =< n_p, a, \mathbf{CO^p_t}, \mathbf{A}, \mathbf{CO^p_{t+1}} >$$

A permission defines an action set $\mathbf{A}$ that an actor a might perform if the contract state is fulfilled $\mathbf{CO^p_t}$ resulting in a contract state that fulfills a set of state conditions $\mathbf{CO^p_{t+1}}$ at time $t+1$.

Definition 3.4 (Sanction)**.** A sanction s_i specifies an agent, action set $\mathbf{A}$, and state conditions:

$$s_i =< n_s, a, \mathbf{CO^p_t}, \mathbf{A}, \mathbf{CO^p_{t+1}} >$$

A sanction defines a set of sanctions (actions) $\mathbf{A}$ on an actor a that might be performed if the a set of contract states are not fulfilled $\mathbf{CO^p_t}$. After the execution of sanctions, this set of state conditions $\mathbf{CO^p_{t+1}}$ is considered to be satisfied at time $t+1$.

Function λ specifies what kind of data analytical function will be applied to input data $\mathbf{X}$ achieving a quality requirements $\mathbf{Q}$.

Service contract c_S specifies how function f will be implemented including data engineering, model training, and model application. Service s is a data analytical function with or without pretrained model M. If s provides a pre-trained model M, no training data is required but predictions can be derived. In practical situations, pre-trained models do not exist but need to be trained based on participants data.

Definition 3.5 (Service contract)**.** A service contract c_s specifies conditions for a service s for implementation of function λ

$$s =< f, \mathbf{X_s}, \mathbf{\Pi} >$$

$\mathbf{X_s}$ data schema of $\mathbf{X}$ and data analytics pipeline requirements $\mathbf{\Pi}$. $\mathbf{\Pi}$ specify contractual promises, i.e. permissions, obligations and sanctions, for the whole data analytical process, including data engineering, model training and model deployment incl. predictions. If $\mathbf{\Pi} = \emptyset$, any service that can implement function type f if it obeys data schema $\mathbf{X_s}$. This allows the integration of a market-based approach by which match-making and negotiation protocols support selection processes. The prediction results $Y(c_d)$ generated by application of s on input data from agent A and B are send to agent A according to contractual promises (cf. Fig. 1).

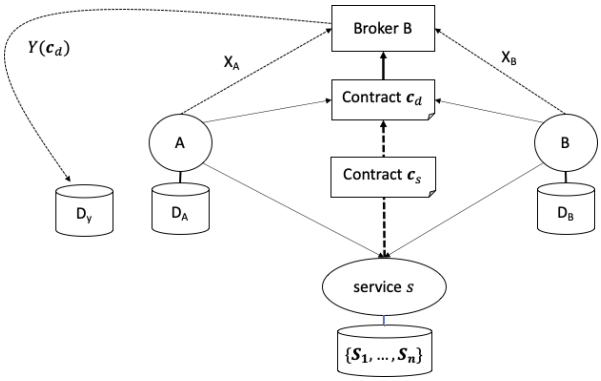

Figure 1. Broker framework.

4. Electronic contracts specification language for federated data economies

From the specification of the broker design pattern, we derived the Contract Specification Language (ECSL+) that extends the proposal by [24]. Kollingbaum and Norman's proposal specifies concepts including *contract*, *agent*, *role*, *obligation*, *permission*, *sanction*, $act_expression$, *do*, *activation*, *expiration*, and *condition*. In this approach, variables are defined externally by additive processes, e.g., negotiation. In federated ecosystems self-descriptive contracts are required that can be

executed by a broker by guaranteed outcome without side effects caused by external processes that are not governed by contractual descriptions. Therefore, we introduce assertion as a means for declaring variables within a contract specification. Furthermore, we allow *contract embedding* which is important for embedding service contracts into data contracts (cf. Fig. 2).

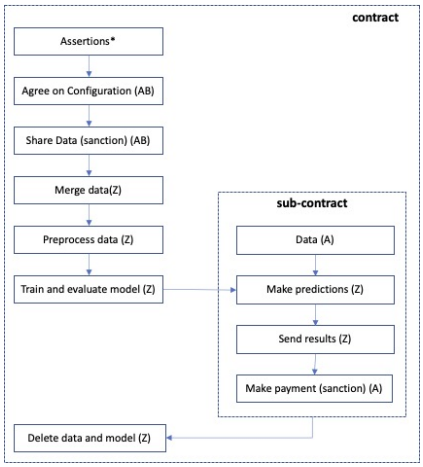

Figure 2. Contract flow including sub-contract.

A data contract consists of several blocks. At the beginning, roles are defined and variables are declared (assertions). The rest of the contract is organized according to a data analytical process: data engineering, model training, model deployment and cleaning:

1. *Sharing data*: obligations of agents to provide specified data
2. *Merge data*: permission of broker Z for merging data provided by agents X
3. *Preprocess data*: permission of Z for standardization and normalization of data into X
4. *Train and evaluate model*: train model M with data X
5. *Delete data and model*: obligation of Z to delete data X, X', Y and model M

Making predictions with model M is described by an embedded contract consisting of several components:

1. *Data*: obligation for agents A to deliver data X'. The contract is terminated if agents A do not share data .
2. *Make predictions*: permission of Z to make predictions with M on X'
3. *Send results*: obligation of Z to send result data Y to specified agents
4. *Make payment*: obligation of agents A to transfer specified amount to agents B (if data trading but not data sharing)

Instantiations of agent sets A and B are contract specific. In a simple case, A and B are instantiated by a single agent each. Permissions can be extended by sanctions. Sanctions can active various forms of actions, including sending fines. Sanctions are optional for obligations of data sharing and payments.

According to [24], promises, ie. permissions, obligations and sanctions, consist of a name, an activity and two conditions: activation and expiration. Activation describes the trigger for activating and an expiration describes a trigger for de-activating a promise. For instance, permission *data_engineering_merge* activates a *merge_data* function on data provided by both agents, if data is received from both (cf. contract *Model_and_Predict_Price*). This permission expires if data is successfully stored in variable *?merged_data*. As long as *?merged_data* exists, this obligation cannot be activated again. The control flow of a contract is implemented by activation and expiration conditions. This also allows modeling of contract flows with parallel activity streams as used for business process modeling [36], such as EPC and BPMN.

```
contract(Model_and_Predict_Price,
 role(?data_supplier_A)
 role(?data_supplier_B)
 role(?broker)
 assertion(?prediction_type = 'regression')
 assertion(?model_type = 'xgboost')
 assertion(?model_architecture =
 [objective ='reg:linear', colsample_bytree = 0.3, lr = 0.1,
 max_depth = 5, alpha = 10, n_estimators = 10])
 assertion(?loss_function = 'rmse')
 assertion(?loss_threshold = 0.8)
 assertion(?deadline = '31-12-2021')
 assertion(?account_A = 'DE19 5919 0000 0000 33 0815')
 assertion(?account_B = 'DE19 5919 0000 0000 33 4711')
 assertion(?payment_A_to_B = 1000 EUR)
 assertion(?fine_A = 100000 EUR)

 obligation(use_configuration,
 do (?broker, use_configuration ( ?prediction_type,
 ?model_type, ?model_architecture, ?loss_function)),
 confirmation(?config_ack_A, ? config_ack_A),
 finish(?ack_config ))

 obligation(share_data,
 do (?data_supplier_A, open_data_safe (?broker,
 ?data_supplier_A, ?data_A_modeling)),
 TRUE,
 grant(?data_A_model_ID))
 ...
 obligation(share_data,
 do (?data_supplier_A, open_data_safe (?broker,
 ?data_supplier_A, ?data_A_prediction)),
 TRUE,
 grant(?data_A_pred_ID))
 ...
 permission(data_engineering_merge,
 do (?broker, merge_data ( ?data_safe_A, ?data_safe_B )),
 data_received (?data_safe_A, ?data_safe_B),
```

```
 checked_in(?merged_data ))

 permission(data_engineering_preprocess,
 do (?broker, preprocess_data(?merged_data)),
 data_received(?merged_data),
 checked_in(?preprocessed_data))

 permission(data_engineering_train,
 do (?broker, train_model(?preprocessed_data,
 ?model_type, ?model_architecture, ?loss_function)),
 data_received(?preprocessed_data),
 checked_in(?trained_model AND ?performance_values))

 Contract(prediction_price) # cf. sub-contract below

 obligation(delete_data_model,
 do (?broker, delete(?trained_model, ?data_A_model_ID,
 ?data_B_model_ID)),
 ack (?payment_received_B),
 ack (?deleted_all)
 )

 sanction(withhold_data_A,
 do (?broker, send_fine (?data_supplier_A, ?fine_A)),
 not received_before(?data_A_modeling, ?deadline),
 received_before(?data_A_modeling, ?deadline)
 )

 sanction(blacklist_A
 do (?broker, put_on_blacklist(?data_supplier_A )),
 not received_before(?data_A_modeling, ?deadline),
 received_before(?data_A_modeling, ?deadline)
 )
 ...
 sanction(send_fine_A,
 do (?broker, send_fine ( ?data_supplier_A, ?fine_A)),
 not received_before(?payment_received_B, ?deadline),
 received_before(?data_B_modeling, ?deadline)
 )
)
```

The sub-contract for prediction on data $\mathbf{X}'$ provided by agent A is goverened by contract *prediction_price*.

```
Contract(prediction_price,
 ...
 permission (prediction_A,
 do ( ?broker, predict(?trained_model, ?data_A_pred_ID)),
 model_exist ( ?trained_model ),
 checked_in ( ?predictions_A ) )

 obligation (send_predictions_A,
 do ( ?broker, send(?data_supplier_A, ?predictions_A)),
 TRUE,
 ack ( ?data_supplier_A )

 obligation (delete_data_A,
 do ( ?broker, delete(?data_A_pred_ID, ?predictions_A)),
 ack ( ?data_supplier_A ),
 ack ( ?delete_A)
 )

 obligation (payment_A,
 do ( ?data_supplier_A, payment (?account_A, ?account_B,
 payment_A_to_B)
 checked_in ( ?trained_model AND ?performance_values),
 ack(?payment_received_B)
 )
)
```

5. Federated Data-Economic Platform

The broker framework is based on the assumption that agents are not necessarily computational agents alone, as investigated by research on multi-agent systems. Instead, a collaborative approach is used by which human decision makers use smart services for data sharing, data trading and data processing in cooperation with other decision makers. Multi-agent systems (MAS) research is often based on computational agents that reason and act on formal representations of believes, desires, intentions and goals (e.g., 3APL [37]). MAS technology is being used to represent, model and simulate dynamic and federated environments [38]. They are a natural way for implementing federated data economic systems. Research on MAS has proposed languages for modeling multi-agent systems (e.g., 3APL [37]), simulation environments (e.g., MATLAB), or implementations by object-oriented languages (e.g., JADE) or with component-based systems (e.g., OSGi [39]).

For reasons of liability but also keeping control of business, decision makers are required to use their believes, desires, and intentions when setting and following business goals, such as on data economic tasks. Therefore, we propose an architecture that supports human decision-making by means of data analytical services, called *smart services*. Smart services work on data analytical tasks and provide decision support. In a simple case, a smart service uses local data for deriving recommendations. This resembles standard data analytical programs. A federation of smart services collaboratively work data analytical tasks. In general, the following requirements are posed for smart service federations (cf. [40, 41, 42]):

1. *Information*: communication between smart services that signal a willingness for cooperation
2. *Negotiation*: building a trust relationship by agreement of contractual promises
3. *Contract execution*: data and model sharing and data analytical processing
4. *Settlement*: exchange of results, execution of financial transactions and contract termination

In this paper, we assume that agents already negotiated a contract on data analysis of shared data. In the following, we use contract execution and settlement for presenting the distributed architecture that supports smart services.

TUCANA is a distributed, web-based architecture that consists of a federation of agents controlling smart services.[1] Agents fully control smart services including development, activation, and termination. In TUCANA; agents can take different roles. In this federated data economic environment, we distinguish between *data agent* and *broker*. This lean organization poses

[1] TUCANA documentation: https://informationservicesystems.github.io/tucana/

requirements on communication channels between agents, i.e. peer-to-peer communication without the need for a central coordinator which would contradict a full federation. Agents are computational environments with local storage. Data is shared via peer-to-peer communication channels. Access to inner states of an agent is controlled by authentication services (e.g., oauth 2.0). An agent can activate several smart services in parallel. Smart services are loosely coupled software systems based on modules, called *minions*. Six different minion types are distinguished:

- *Perceiver*: receiving data from other agents or sensors
- *Data engineer*: data engineering on data
- *Contract manager*: processing contracts, incl. receiving, processing and sharing data; impose sanctions
- *Thinker*: AI capabilities, incl. model training and execution
- *Communicator*: sending data to other agents or communicating to users
- *Guardian*: interface to external data provider and receiver

A smart service controls a protected, local storage, i.e. without access by other smart services. Communication between smart services is channeled through perceiver (receiving messages) and communicator minions (sending messages). A key requirement is that smart services are fully realized in web environments. Communication with technologies external to the World Wide Web is managed by guardian minions, e.g., cloud infrastructures. Smart Services are defined by smart service configurations that describe process flows of minions. Thinker minions encapsulate AI models in general and AI learning in particular.

Minions are encapsulated modules that are activated by messages (cf. Figure 3). A smart service configuration defines a finite state model for minions. Data is stored in local stores by a guardian. Input-output behavior is implemented by perceivers, communicators and guardians.

In federated settings, agents interact with one another based on their smart services (Fig. 4). In this example Cerea runs two smart services (P-T-C) while Bob maintains three services with one (P-C) only visualizing messages received by Cerea and Woda.

Actors in federated data economic ecosystems are realized as data agents maintaining smart service that

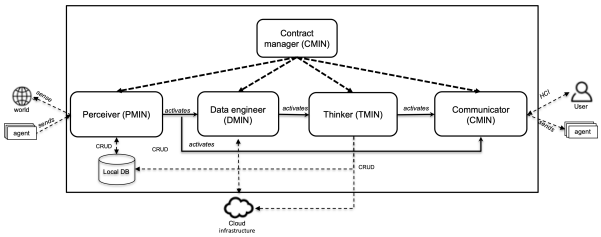

Figure 3. Visualization of a general smart service configuration

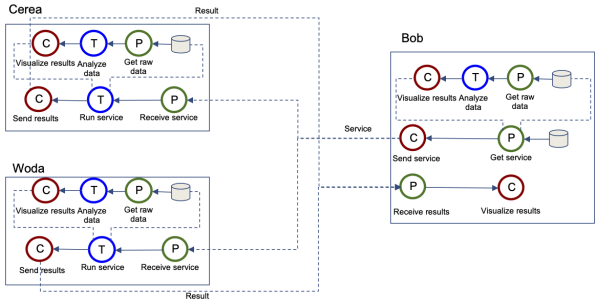

Figure 4. Example with three agents running interconnected smart services.

exchange data with local smart services but also with smart services maintained by other agents (cf. example in Figure 4). Requests are received by a perceiver and data sharing in handled by a communicator minion. The broker is modeled by a perceiver - thinker - communicator process. This process is activated and controlled by a contract manager. Contractual promises associated with data agents are send as requests (e.g., *open_data_safe* in contract of Section 4). Promises for the broker are send to associated minions (e.g., *merge_data* is send as an activation to the *data engineer*) with a handle to the appropriate local data (*?data_safe_A* and *?data_safe_B*).

Modeling the broker framework in TUCANA is straight-forward. A contract is modeled as a finite state machine c_f that is superimposed on the smart service configuration sc. Contractual promises are triggered based on the legal state that is the set of all local variables of an agent defined by the contract model c_f. Activities are registered with minion instances by a activity registry. For instance, *train_model* is registered with the thinker minion of the smart broker service. Registered minions are activated as soon as activation conditions of contractual promises are met. Designers of smart contracts can leave out specifications for control and data flow in smart service configurations and leave this to contract models alone. This helps to avoid unintended side-effects.

6. TUCANA platform implementation

TUCANA is implemented by using web-technologies, in particular in HTML and JavaScript deployed for web browser. Thus, a data ecosystem build with TUCANA can be deployed on any device that supports browser technologies, including embedded devices, smartphones, laptops workstations but also cloud environments. Because HTML and Javascript are integral parts of web environments, both can be treated as data and executable code. The data property allows light-weight exchange of smart services between agents, as it is feasible for data and AI models. Thus, data, models and code are all fist-class citizens of smart services and can be transferred without restrictions and installations.[2]

Peer-to-peer communication is realized by WebRTC [43]. WebRTC is an industry effort for adding peer-to-peer communication paradigm between browsers. By establishing a peer connection, messages can be directly exchanged between browsers without intermediating servers. The WebRTC API provides peer-to-peer connection management, encoding/decoding negotiation, selection and control, media control, firewall and complex NAT traversal [44].

The runtime environment of TUCANA consists of a data-access service layer that handles access to local data, data provided by other agents and backend data services incl. cloud infrastructures. Local data is stored in IndexedDB of the browser. Minions are implemented as Javascript classes. Smart services are controlled by a minion state controller executed via an underlying state-transition engine. An agent starts a smart service by selecting a smart service configuration that contains requirements for minions. Minions are selected from the local minion repository when satisfying minion manifests of the smart service configuration (version, interfaces) and connection requirements, ie. connection manifests. Both manifest sets describe a Smart Service. A smart service configuration is fully descriptive (non-binary) and can be transferred to other agents.

All data exchanged between agents is rendered by self-describing web data formats, ie. data is accompanied by semantic descriptions in particular JSON-LD. RESTFul APIs are used for data exchange (JSON-LD serialization and API).

7. Application

In this section, we describe a data economic application implemented in TUCANA with two market particpants, farmer A and farmer B, and a trusted-third party. A and B produce avocados with interest in predicting avocado prices for making appropriate investment decisions. A and B have complementary datasets on avocado prices. Both have come to an agreement that using data from both sides would give more accurate price predictions for the next three months. But neither A nor B are willing to share data directly ('lack-of-trust' condition) but they are interested in prediction results (user interface of implementation cf. Figure 5). After joint discussions, they realized that a contract can be applied (cf. contract from Section 4).

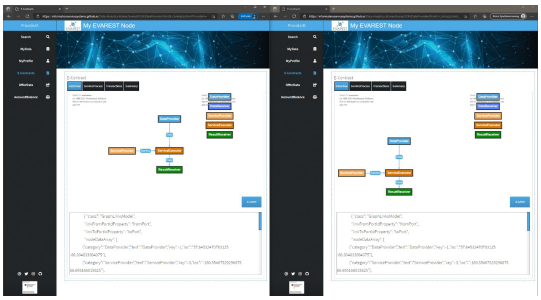

Figure 5. User interface with agent A (left) and agent B (right)

The broker receives and executes the contract. After receiving data from both agents A and B, it performs data engineering tasks, model training. By executing the embedded contract, the broker receives new data from agent A and uses this it for making predictions. Agent B is compensated for providing data and finally, all data and the model is deleted.

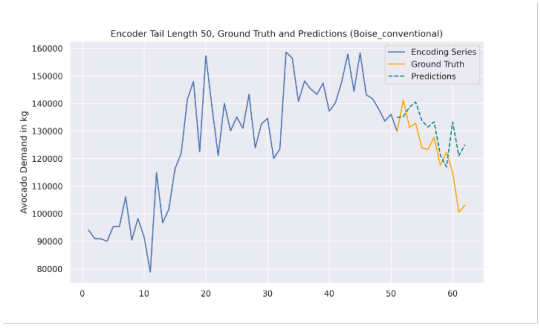

Figure 6. Prediction of avocado prices

Thus, this contract is side-effect free which is a key element for trust-building between agent A and B. The combined dataset of A and B consists of only 307 datapoints, i.e., average price for a week.[3] A LSTM model was used within a thinker minion that resulted in a MAPE of 12.9% for the predictions of the next

[2]This currently holds for minions implemented in Javascript compatible with ECMAScript 6 (ES6)

[3]Data set: https://www.kaggle.com/timmate/avocado-prices-2020

three months (dashed line in Figure 6). Finally, farmer B received a payment according to the contract.

8. Summary and Outlook

Business actors are always longing for additional data collected and maintained by other actors. Sharing and trading data is a necessity for higher-quality predictions and data-driven decision making. We have shown by a game-theoretic model that a standard two-player situation generally favors no-sharing strategies. Contracts are a means for overcoming mistrust between business actors. We have presented a conceptual framework for a broker and an electronic contract specification language specifically adapted to data economic situations. Electronic contracts are conceived as finite state automata. Furthermore, we have discussed how this conceptual broker framework including electronic contracts is implemented on a federated data-economic platform (TUCANA). TUCANA is based on a service-based architecture with re-usable software modules, called minions. Smart services are configurations of minions. In this paper, we have focused on smart services that enable broker for executing electronic contracts on behalf of business agents. A web-based architecture for TUCANA has been discussed and used for implementing an example for training and using a machine learning model for making predictions on joint data based on shared data.

This paper is a first step towards a full understanding of electronic contracts and the role of a broker in federated data ecosystems. The proposed conceptual model for electronic contracts needs to be enhanced by secure identification schemes for actors and brokers (first implementation with oauth2.0 is currently evaluated). Generally, the approach needs to be refactored by security requirements while the communication based on WebRTC is secure by design. Even though that the broker only shares data between business actors according to contract, it is feasible that data can be reconstructed from result data. Differential privacy [45] and homomorphic encryption [46] are both means for protecting input data from malicious actors.

We are working on a mapping the electronic contract model onto different smart contract schemes, i.e. IOTA and Blockchain 2.0 architectures. A prototype has been tested that ports our contract model to WASP[4] that implements the IOTA Smart Contract Protocol.

[4] https://github.com/iotaledger/wasp

9. Acknowledgments

This work is partially funded by the German Ministry for Economic Affairs and Energy project EVAREST (grant 01MT19003B, www.evarest.de). I am grateful for the conversations I had with Sabine Janzen, Hannah Stein, Jannis Eickhoff, and Jacob Chen.

References

[1] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *science*, vol. 185, no. 4157, pp. 1124–1131, 1974.

[2] E. Brynjolfsson, L. M. Hitt, and H. H. Kim, "Strength in numbers: How does data-driven decisionmaking affect firm performance?," *Available at SSRN 1819486*, 2011.

[3] P. Tambe, "Big data investment, skills, and firm value," *Management Science*, vol. 60, no. 6, pp. 1452–1469, 2014.

[4] L. Li, "Information sharing in a supply chain with horizontal competition," *Management science*, vol. 48, no. 9, pp. 1196–1212, 2002.

[5] A. Y. Ha, S. Tong, and H. Zhang, "Sharing demand information in competing supply chains with production diseconomies," *Management science*, vol. 57, no. 3, pp. 566–581, 2011.

[6] M. Farboodi and L. Veldkamp, "A growth model of the data economy," tech. rep., National Bureau of Economic Research, 2021.

[7] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big data*, vol. 1, no. 1, pp. 51–59, 2013.

[8] C. L. Borgman, "The conundrum of sharing research data," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 6, pp. 1059–1078, 2012.

[9] B. Otjacques, P. Hitzelberger, and F. Feltz, "Interoperability of e-government information systems: Issues of identification and data sharing," *Journal of management information systems*, vol. 23, no. 4, pp. 29–51, 2007.

[10] S. Negash and P. Gray, "Business intelligence," in *Handbook on decision support systems 2*, pp. 175–193, Springer, 2008.

[11] D. M. Strong, Y. W. Lee, and R. Y. Wang, "Data quality in context," *Communications of the ACM*, vol. 40, no. 5, pp. 103–110, 1997.

[12] N. Melville, K. Kraemer, and V. Gurbaxani, "Information technology and organizational performance: An integrative model of it business value," *MIS quarterly*, pp. 283–322, 2004.

[13] A. Braud, G. Fromentoux, B. Radier, and O. Le Grand, "The road to european digital sovereignty with gaia-x and idsa," *IEEE Network*, vol. 35, no. 2, pp. 4–5, 2021.

[14] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial internet of things," *Journal of Software Engineering and Applications*, vol. 9, no. 10, pp. 533–546, 2016.

[15] T. E. Pronk, P. H. Wiersma, A. van Weerden, and F. Schieving, "A game theoretic analysis of research data sharing," *PeerJ*, vol. 3, p. e1242, 2015.

[16] J. M. Puaschunder, "Data fiduciary in order to alleviate principal–agent problems in the artificial big data age," in *Information for Efficient Decision Making: Big Data, Blockchain and Relevance*, pp. 41–90, World Scientific, 2021.

[17] H. Zech, "Data as a tradeable commodity–implications for contract law," in *Proceedings of the 18th EIPIN Congress: The New Data Economy between Data Ownership, Privacy and Safeguarding Competition, Edward Elgar Publishing, Forthcoming*, 2017.

[18] C. Fried, *Contract as promise: A theory of contractual obligation*. Oxford University Press, USA, 2015.

[19] S. Angelov and P. Grefen, "A conceptual framework for b2b electronic contracting," in *Working Conference on Virtual Enterprises*, pp. 143–150, Springer, 2002.

[20] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Computer Architecture Letters*, vol. 29, no. 12, pp. 1104–1113, 1980.

[21] S. P. Jones and J.-M. Eber, "How to write a financial contract," 2003.

[22] J. Andersen, E. Elsborg, F. Henglein, J. G. Simonsen, and C. Stefansen, "Compositional specification of commercial contracts," *International Journal on Software Tools for Technology Transfer*, vol. 8, no. 6, pp. 485–516, 2006.

[23] Y.-H. T. Cristiano Castelfranchi, "The role of trust and deception in virtual societies," *International Journal of Electronic Commerce*, vol. 6, no. 3, pp. 55–70, 2002.

[24] M. J. Kollingbaum and T. J. Norman, "Supervised interaction: creating a web of trust for contracting agents in electronic environments," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pp. 272–279, 2002.

[25] M. Dignum, *A model for organizational interaction: based on agents, founded in logic*. SIKS, 2004.

[26] W. Van Der Aalst, K. M. Van Hee, and K. van Hee, *Workflow management: models, methods, and systems*. MIT press, 2004.

[27] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–4, IEEE, 2018.

[28] B. Egelund-Müller, M. Elsman, F. Henglein, and O. Ross, "Automated execution of financial contracts on blockchains," *Business & Information Systems Engineering*, vol. 59, no. 6, pp. 457–467, 2017.

[29] R. M. Lee, "A logic model for electronic contracting," *Decision support systems*, vol. 4, no. 1, pp. 27–44, 1988.

[30] A. Mavridou and A. Laszka, "Designing secure ethereum smart contracts: A finite state machine based approach," in *International Conference on Financial Cryptography and Data Security*, pp. 523–540, Springer, 2018.

[31] J. Ladleif and M. Weske, "A unifying model of legal smart contracts," in *International Conference on Conceptual Modeling*, pp. 323–337, Springer, 2019.

[32] E. Solaiman, I. Sfyrakis, and C. Molina-Jimenez, "A state aware model and architecture for the monitoring and enforcement of electronic contracts," in *2016 IEEE 18th Conference on Business Informatics (CBI)*, vol. 1, pp. 55–63, IEEE, 2016.

[33] C. Molina-Jimenez, E. Solaiman, I. Sfyrakis, I. Ng, and J. Crowcroft, "On and off-blockchain enforcement of smart contracts," in *European Conference on Parallel Processing*, pp. 342–354, Springer, 2018.

[34] C. Griffo, J. P. A. Almeida, and G. Guizzardi, "Conceptual modeling of legal relations," in *International Conference on Conceptual Modeling*, pp. 169–183, Springer, 2018.

[35] V. R. Lesser, "Reflections on the nature of multi-agent coordination and its implications for an agent architecture," *Autonomous agents and multi-agent systems*, vol. 1, no. 1, pp. 89–111, 1998.

[36] B. Curtis, M. I. Kellner, and J. Over, "Process modeling," *Communications of the ACM*, vol. 35, no. 9, pp. 75–90, 1992.

[37] K. V. Hindriks, F. S. De Boer, W. Van der Hoek, and J.-J. C. Meyer, "Agent programming in 3apl," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 4, pp. 357–401, 1999.

[38] G. Santos, T. Pinto, I. Praça, and Z. Vale, "Mascem: Optimizing the performance of a multi-agent system," *Energy*, vol. 111, pp. 513–524, 2016.

[39] R. J. C. Benito, D. G. Márquez, P. P. Tron, R. R. CASTRO, N. S. Martín, and J. L. S. Martín, "Smepp: A secure middleware for embedded p2p," *Proceedings of ICT-MobileSummit*, vol. 9, 2009.

[40] B. F. Schmid and M. A. Lindemann, "Elements of a reference model for electronic markets," in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, vol. 4, pp. 193–201, IEEE, 1998.

[41] M. Schoop, A. Jertila, and T. List, "Negoisst: a negotiation support system for electronic business-to-business negotiations in e-commerce," *Data & Knowledge Engineering*, vol. 47, no. 3, pp. 371–401, 2003.

[42] M. He, N. R. Jennings, and H.-F. Leung, "On agent-mediated electronic commerce," *IEEE Transactions on knowledge and data engineering*, vol. 15, no. 4, pp. 985–1003, 2003.

[43] A. B. Johnston and D. C. Burnett, *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC, 2012.

[44] S. Loreto and S. P. Romano, "How far are we from webrtc-1.0? an update on standards and a look at what's next," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 200–207, 2017.

[45] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[46] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Annual Cryptology Conference*, pp. 868–886, Springer, 2012.