User Model Interoperability in Education: Sharing Learner Data using the Experience API and Distributed Ledger Technology

Konstantinos Karoudis
University of London, Birkbeck College, United Kingdom
Orcid.org/0000-0002-6848-9020

George D. Magoulas
University of London, Birkbeck College, United Kingdom
Orcid.org/0000-0003-1884-0772

Abstract

Learning analytics and data mining require gathering and exchanging learner data for further processing and designing of activities tailored to learner's characteristics, context, and needs. Currently, systems that store learners' attributes should, ideally, be operated and controlled by responsible and trustworthy authorities that guarantee the protection and sovereignty of data, and use objective criteria to protect and represent all parties' interests. This chapter introduces a peer-to-peer method for storing and exchanging learner data with minimal trust. The proposed approach, underpinned by the Experience API standard, eliminates the need of a mediator authority by using distributed ledger technology.

Introduction

The past decade has seen the development of numerous user-adaptive systems that can adapt their behaviour to the individual user. In order to provide the adaptation effect, these systems typically acquire and store relevant information about each user in an internal representation called user model. The process of creating and maintaining user models, either implicitly by observing users' interactions or explicitly by requesting their direct input, is called user modelling (Brusilovsky & Millán, 2007).

Learner modelling refers to the process of gathering relevant information about a learner, inferring their current cognitive state, and generating a representation that can be used by a system to offer adaptation (Chrysafiadi & Virvou, 2013). Such a representation, which is a special type of user model and consists of data about the learner or about what the learner does, is called learner model (Cocea & Magoulas, 2015). Adaptive systems that employ learner models are called adaptive learning systems. These provide an environment that intelligently adjusts to individual learners by presenting suitable information, instructional materials, feedback and recommendations based on their unique characteristics and situation (Graf & Kinshuk, 2014). In the rest of this chapter, the scope is limited to dynamic user models that can change over time (Kay, 1999).

User modelling and adaptive learning systems have historically contributed to developments in the area of learning analytics, but at the same time research in learning analytics has been used to generate user models for adaptive learning systems (Siemens, 2013). Current practice indicates that learning analytics can make significant contributions and act as an enabler for the development and introduction of adaptive personalised learning because it can enable tracking

individual student engagement, attainment, and progression in near-real time (Sclater, Peasgood, & Mullan, 2016). According to Bienkowski, Feng, and Means (2012), user modelling and adaptation belong to the broad application areas of educational data mining and learning analytics. The latter two provide data that can be used not only to build user models, but also to profile users, i.e. to group similar users into categories using salient characteristics. User modelling and profiling can enable real-time adaptations and their long-term objective is to provide adapted and personalised learning environments for individuals or groups of users in order to maximize learning effectiveness and efficiency.

Both of the terms adaptation and personalisation are used to indicate the capacity of a system to adapt to users. However, adaptation refers to the changes a system produces for individual users based on their models while personalisation refers to the effect the system has on the users. It is thus important to distinguish between personalisation, which is the purpose, and adaptation, which is the mechanism to achieve the purpose (Cocea & Magoulas, 2015).

Recent developments in adaptive learning highlight the advantages of learner data exchange between adaptive learning systems for potentially improved personalisation services (Ghorbel, Zayani, & Amous, 2015; Martínez-Villaseñor, González-Mendoza, & Danvila Del Valle, 2014). The process of exchanging distributed user data across applications is called user model interoperability (Carmagnola, Cena, & Gena, 2011).

This chapter will review the area of user model interoperability and examine the emerging role of distributed ledger technology in the context of learner modelling. The aim is to propose a new methodology for sharing learner information that can guarantee the quality (provenance and accuracy) of the exchanged learner data while minimising the need of trust between the parties that share information. The adopted approach envisages a distributed ledger storing learning experiences in the form of Experience API statements, which can be used to build individual learner models.

The next section will introduce the background of user model interoperability and discuss the classification of interoperable systems, as well as ways of addressing privacy issues. The following section will present the Experience API specification, which enables various types of educational technologies to capture and exchange learner data, and its affordances. It will also define the term provenance and describe the xAPI Extended, a proposed enhancement to the existing standard. The subsequent section will explain basic terms of distributed ledger technology and provide a classification of blockchain systems based on consensus process and determination. The second part of this section will consider six well-known consensus protocols and introduce the reader to smart contracts and the Ethereum blockchain.

The penultimate section will propose a theoretical framework for storing xAPI statements on the blockchain and also provide an overview of the steps performed by the storage mechanism. Finally, the last section will summarise the advantages of the framework and outline future research directions.

User Model Interoperability

In one of the early definitions of the term, interoperability is described as the ability of two or more software components to exchange data and cooperate overcoming differences in language, interface, and execution platform (Wegner, 1996). Three main types of interoperability have been identified in the literature so far (Carmagnola et al., 2011):

- *Structural interoperability* refers to the possibility of overcoming the differences between information systems at the access level (e.g. communication protocols, application programming interfaces, etc.)
- *Syntactic interoperability* (also referred to as *language interoperability*) is required for any attempts of further interoperability and concerns the communication and exchange of data between information systems at the application level, i.e. the capability of different systems to interpret the syntax of the data in the same way.
- *Semantic interoperability* (also called *logical interoperability*) overcomes differences between information systems at the knowledge (meaning) level. Systems exchange information on the basis of shared, pre-established, and negotiated meanings of terms and expressions.

Different educational systems may use distinct representations for the same data, e.g. other syntactic and conceptual structures, terminologies, or interpretations of the same terminology. Therefore, achieving syntactic and semantic interoperability is challenging because it requires a high degree of alignment among the applications, especially in an open and dynamic environment like the Web (Aroyo et al., 2006).

This section will discuss interoperability with regard to user models. The process of user model data exchange can be analysed in four phases (Carmagnola et al., 2011). In the first phase, the systems that collect information about a specific user have to be discovered (service discovery phase) (Carmagnola & Cena, 2009). Then, all discovered systems have to reach an agreement on the identity of the user (user identification phase). The third phase is where the actual exchange of user information takes place (data exchange phase). Finally, the reliability of the exchanged data has to be evaluated qualitatively and quantitatively by the applications (data evaluation phase). In the rest of this chapter, the discussion focuses on issues related to the third phase of the user model data exchange process.

The main motivations for supporting interoperable user models relate to key user modelling issues identified so far. The first issue is the initialisation of user models, also known as the cold start problem, where applications have to provide appropriate adaptation for new users when the corresponding models do not hold enough information (Alfred Kobsa, 2007; Wang et al., 2008). User model initialisation can also save users time because they do not need to fill in information to complete their model every time they interact with a new system (Vassileva, 2001). Other issues described in the literature include the qualitative and quantitative improvement of user models, which have the potential to produce better adaptation results. The former improvement refers to the accuracy of user model data in representing users' interests and needs while the latter can be described as increased coverage, i.e. user models covering more aspects of the users (Berkovsky, Kuflik, & Ricci, 2008;

Heckmann, Schwartz, Brandherm, Schmitz, & Wilamowitz-Moellendorff, 2005; Walsh, O'Connor, & Wade, 2012).

Classification of Interoperable Systems

Systems handling interoperable user models can be classified into three categories with respect to their main interoperability task (Carmagnola et al., 2011). The first category includes systems designed to facilitate user model exchange but are non-adaptive and thus do not need user models for themselves. In the second category, fall systems that are designed to provide the service of real-time reasoning and adaptation over received sets of data. Such systems do not share user model information as they provide a one-to one service. The third type of systems is designed to collect user data from different sources, integrate this information to form richer user models, and then share the resulting models across applications.

The architecture of interoperable systems has changed dramatically since their first appearance. A recent survey reported three possible architectures for classifying interoperable systems defined by the centralisation degree of their user models: The centralised, the decentralised, and the mixed architectures (Carmagnola et al., 2011).

In centralised architectures, a centralised repository known as user modelling server stores user models for a group of (remote) systems (called clients). In this case, all user models (for a given client group) are unique and their data exchange takes place between user modelling servers. By contrast to the centralised architectures, in decentralised architectures each system develops independent, (partial or fragmented) local user models that do not necessarily adhere to a common representation scheme (Niu, McCalla, & Vassileva, 2003). This means that a certain degree of user model duplication may exist either for parts or for complete user models. Decentralised systems exchange user model data using peer-to peer connections. Finally, mixed architectures combine both previously described approaches, i.e. locally stored user models refer to a central user model (stored on a centralised repository) to ensure interoperability.

Interoperable systems can also be classified according to their internal representation of exchanged user data. Two major approaches for representing interoperable data are the standardisation-based user modelling (or common user model representation) and the mediation-based user modelling (or translation approach) (Viviani, Bennani, & Egyed-Zsigmond, 2010). In the former (top-down) approach, all applications have to conform to one or more predefined standards. In learner modelling, such standards are the IEEE PAPI, IMS RDCEO and LIP (Dolog & Schäfer, 2005). A common user model representation is also possible through a shared user model ontology, like the General User Model Ontology (GUMO) (Heckmann et al., 2005), or the Unified User Context Model (UUCM) (Mehta et al., 2005). In the latter (bottom-up) approach, the various user model representations are being translated into a central, shared user model through a mediator component that manages the mapping (Van Der Sluijs & Houben, 2006), whilst there also exist other automatic or semi-automatic ontology mapping techniques (Doan, Madhavan, Dhamankar, Domingos, & Halevy, 2003; Carmagnola & Dimitrova, 2008).

Recent advances in the field address the problem of the representation of exchanged user data using a third approach, which combines the previous two approaches and aims to overcome their disadvantages (Ghorbel et al., 2015). A notable example is the FUSE domain-aware approach that uses a canonical model as a consistent shared user model representation together with a translation process based on mappings (Walsh, O'Connor, & Wade, 2013). Other examples of hybrid approaches found in the literature use multiple mediation types (Berkovsky et al., 2008), or a distributed semantic conversation framework for the exchange of user data (Cena, 2011).

Internal data representation of exchanged user information goes hand in hand with data integration. Therefore, a further classification of interoperable systems can be made with regard to the way they integrate exchanged data and deal with arising conflicts. Two possible data integration approaches have been employed so far, i.e. no integration and fusion of collected data and existing user models (Ghorbel et al., 2015). The majority of existing systems adopt the first approach using exchanged user data only when needed without merging it with existing user models. It is the role of the mediator to convert data from the source to the destination format on the fly upon request without necessarily storing the information locally. The term active learner modelling has been used in the literature to describe the just-in-time computation of user models to a specific breadth and depth when a client application makes a request (McCalla, Vassileva, Greer, & Bull, 2000). Alternatively, user models can be constantly updated as new information is provided (Assad, Carmichael, Kay, & Kummerfeld, 2007). Systems that adopt the second approach incorporate conflict resolution to merge exchanged data with existing user models. Possible ways of resolving conflicts found in the literature include the measurement of credibility of the exchanged data and their supplier (Carmagnola & Dimitrova, 2008), and the manual method where conflict detection is performed by an administrator (Walsh et al., 2013).

Privacy and Trust

Respecting users' privacy and enabling users' control over their data has been recognised as one of the fundamental challenges of user data sharing across applications, not only in the primary phase of data collection, but also in the secondary phase of data sharing and reuse. However, existing work on privacy addresses this challenge mostly in the context of centralised architectures (Iyilade & Vassileva, 2013). Interoperable systems that handle user information must be trustworthy, i.e. they must obtain users' permission to collect and exploit their data, and not release it to third party systems without user consent (Kay & Kummerfeld, 2006).

Proposed solutions and approaches to address the challenge of users' privacy can be grouped into six categories, i.e. different access rights, pseudonymous personalisation, encryption techniques, perturbation techniques, scrutable user model, and joining consortia and organisations (Carmagnola et al., 2011). The first category attempts to tackle the privacy issue by assigning different access rights to services through a role-based access control mechanism. A variation of this approach splits the user model into multiple levels and assigns different access rights to each of them (Kay, Kummerfeld, & Lauder, 2002). In pseudonymous

personalisation, each user has a unique and persistent identifier called pseudonym, which is used to differentiate them from other users (Alfred Kobsa & Schreck, 2003). The third category includes approaches that encrypt user data to preserve user privacy. A notable example in this category uses the distributed Probabilistic Latent Semantic Analysis (PLSA) as a mean to preserve user privacy (Mehta, 2007). Perturbation is a privacy-preserving data mining technique in the area of collaborative filtering where some changes are introduced in the exchanged user information, in order to hide the exact user model from malicious attacks (Berkovsky, Eytani, Kuflik, & Ricci, 2005). The fifth category uses scrutable user models that can be scrutinised, and associated with security preferences by their users (Kay & Kummerfeld, 2006). In the final category, fall approaches where systems adopt standards, and join third-party independent consortia and organisations in order to ensure user privacy. The use of the Lightweight Directory Access Protocol (LDAP) is such an example because it provides a security model that is used to secure user information and privacy (Alfred Kobsa & Fink, 2006).

The Experience API

The Experience API (xAPI) is a specification that enables different learning technologies to capture data about a person's or a group's wide range of learning experiences in a consistent format using the xAPI vocabulary. Having xAPI statements as a common format for sharing collected streams of learning experiences guarantees semantic interoperability because no data translation or mapping is needed. Structural interoperability is achieved by exposing the API as RESTful web services, while following the rules of JavaScript Object Notation (JSON) for serialisation ensures syntactic interoperability.

This section will provide a brief overview of the Experience API (xAPI) specification and discuss its pedagogical and interoperability affordances. Moreover, it will examine the possibility of using xAPI statements as provenance and look into the xAPI Extended, a suggested enhancement to the existing xAPI standard.

xAPI Specification Overview

The Experience API was developed by the Advanced Distributed Learning (ADL) Initiative as a means of tracking/recording learning experiences and learner activities between a client called Learning Record Provider (LRP) and a server called Learning Record Store (LRS). xAPI is both a learning-technology specification and a suite of four APIs provided as RESTful web services, namely Statement, State, Agent, and Activity Profile API. LRPs utilise the Statement API to track formal and informal learning experiences both online and offline, while the rest APIs enable richer reporting and learning analytics. xAPI enables the trusted exchange of information between trusted sources by offering optional security methods (U.S. Department of Defence, Advanced Distributed Learning (ADL) Initiative, 2017).

Typically, a learning experience is tracked on behalf of a learner by a LRP, who creates a Learning Record (LR) and sends it to one or more LRSs. A LR can take on many forms, including statements, documents, and their parts. The Statement API tracks and retrieves LRs in the form of immutable statements consisting of four parts, i.e. the actor, the verb, the activity and additional properties. A basic form of an xAPI

statement corresponds to the sentence “I did this”, where “I” is modelled as an actor, “did” as a verb, and “this” as an object. xAPI supports persona management for allowing selective access to one's personal data. Each persona represents the “I” in the previous sentence, and multiple personas can be associated with a single learner (U.S. Department of Defence, Advanced Distributed Learning (ADL) Initiative, 2016).

xAPI Affordances

The selection of xAPI for the learner data representation was mainly motivated by both its pedagogical and interoperability affordances. This chapter will refer to the term affordance as “a perceived action-promoting property or relation between particular aspects of the situation and the subject who plans or undertakes actions in a certain environment” (Normak, Pata, & Kaipainen, 2012, p. 268).

One of the most important pedagogical affordances of xAPI is the ability to record both online and offline learning experiences into immutable statements for all types of learning settings. This makes xAPI suitable for lifelong learner modelling, and previous research has established that claim by demonstrating how xAPI affordances can be mapped on to pedagogical framework affordances, in order to enable the design of personalised learning paths for lifelong learners across the cumulative learning continuum (Karoudis & Magoulas, 2016).

Lifelong learner modelling refers to the ability to model a dynamic and changing user throughout lifetime interactions with a variety of resource providers (Kay, 2008). Investigating how to enable the interoperability of dynamic user models could, therefore, provide a promising solution for lifelong learner modelling.

Provenance

One of the prominent characteristics of the xAPI is the ability to track and retrieve learning information in the form of immutable statements, i.e. statements that cannot be changed once they have been created. However, this property alone is not sufficient to guarantee the reliability of recorded learner data. In order to fully trust xAPI statements, one needs information about the people, institutions, entities, and activities, involved in producing, influencing, or delivering that data. A record that contains all the above information and is used to form assessments about the data quality, reliability or trustworthiness is called provenance. W3C has published the provenance specification in PROV, a set of documents that describe how inter-operable interchange of provenance information can be achieved in heterogeneous environments, such as the Web (Groth & Moreau, 2013).

Recent research suggests that learning process logs are, in essence, provenance and thus it is possible to perform a lossless conversion from xAPI statements to W3C PROV (De Nies, Salliau, Verborgh, Mannens, & Van de Walle, 2015). The proposed method uses a formal ontology of the xAPI vocabulary, an xAPI statement interpreter to JSON-LD, and a tool implementing the mapping from JSON-LD to PROV statements. In this way, the trustworthiness and interoperability of xAPI can be increased by creating a reversible mapping workflow from xAPI statements to

valid PROV statements. These statements can then be used by recommender systems e.g. to build recommendations upon learning paths (Corbi & Burgos, 2014).

The xAPI Extended

The current Experience API specification provides the technical means for sufficient user data protection in the form of a Persona Data Locker (PDL), which is a database that stores all learning activities for a specific user. Each PDL is secured through the user credentials, two-factor authentication, and a unique PDL identity. However, the PDL is only a theoretical element of the xAPI specification. In standard practice, xAPI envisages the flow of learning content from a learning provider to a learning record store, which can then transfer all stored information to third party applications without user consent. In this way, the recommended implementation supplants the functionality of the PDL and therefore deprives users' right to sovereignty over their own data.

An enhancement that can resolve this data privacy issue in the original Experience API specification has recently been proposed by Schaffarzyk (2015). The xAPI Extended envisages the flow of data from a learning content provider to the respective PDL of each learner in a first step. Information stored in a PDL can then be transferred to a LRS or xAPI enabled third-party applications according to users' personal settings defined in each PDL.

Although this approach solves the issue of user privacy and control over their data, it still requires systems that provide the PDLs for users to be operated and controlled by responsible and trustworthy authorities. A way to overcome this limitation using distributed ledger technology will be described in the following sections.

Distributed Ledger Technology

A distributed ledger can be defined as a shared database that stores assets (financial, legal, physical or electronic) across multiple sites, geographies or institutions. All changes to the ledger are reflected to all copies, and each participant can have their own identical copy of the ledger. The security and accuracy of the assets is enforced by means of cryptographic keys, signatures and distributed consensus, which control the access and modification rights of the participants (Walport, 2015).

The most notable implementation of distributed ledger technology is blockchain, which became popular as the core technology that underlies the cryptocurrency Bitcoin (Nakamoto, 2008). A blockchain comprises a chain of data packages (called blocks) that hold a complete list of transaction records and thus depicts a ledger of the entire transaction history (Nofer, Gomber, Hinz, & Schiereck, 2017). Additions to the ledger are decided on a consensus basis by multiple network nodes that agree on the validity of a block and its contained transactions. The key characteristics of blockchain are persistency, anonymity, auditability, and decentralisation, i.e. decentralised transactions that take place without the need of a central trusted agency. These properties can reduce transaction cost significantly while at the same time improving efficiency (Zheng, Xie, Dai, Chen, & Wang, 2017).

Classification of Blockchain Systems

There are two important terms that one needs to understand when examining blockchain systems, the *consensus process* and the *consensus determination*. The former term is used to specify who is eligible to join the consensus process and generate blocks. Systems that are free for anyone to join are called unpermissioned (or permissionless), whereas systems that are not open to the public are called permissioned. Therefore, the consensus process measures the openness of a system. The latter term specifies the number of nodes that are allowed to participate in the block validation process and determine the current state of the chain.

With regard to the above-mentioned terms, blockchain systems can be classified into public blockchains, consortium blockchains, and (fully) private blockchains (Buterin, 2015). The rest of this part will analyse these three blockchain categories while also considering additional criteria like read permission, degree of centralisation, immutability, and efficiency (Zheng et al., 2017).

Public blockchains. Public blockchains are unpermissioned blockchains because their consensus process is open, i.e. it is free for anyone to join. All members take part in the block verification process and all transactions are visible to the public. Immutability of transaction records is guaranteed by the large number of participants, which, however, has a negative impact on efficiency because the large number of nodes increases transaction and block propagation time.

Consortium blockchains. In consortium blockchains, a predetermined set of users controls both consensus process and determination. Such blockchains are partially centralised (and therefore permissioned), and have three variants for read permission, i.e. public, restricted to the members or a hybrid approach. In terms of immutability, the limited number of participants makes consortium blockchains less tamper-proof when compared with public blockchains. However, the smaller number of validators increases their efficiency.

Private blockchains. In a private blockchain, consensus process and determination are controlled by a single entity (e.g. an organisation). These blockchains, which are also permissioned like the consortium ones, are regarded as centralised networks and have read permission that is either public or restricted to a random degree. Transactions on private blockchains are less tamper-resistant than those of the previous two categories. However, private blockchains have the lowest transaction cost and the highest efficiency.

Consensus Protocols

The previous part of this section discussed the terms consensus process and consensus determination, which describe who can control and update the shared state of a blockchain. This part will present various sets of rules and procedures that preserve a consistent transaction state between multiple nodes and help them reach an agreement about the overall state of the chain. The process by which a majority of nodes comes to an agreement on the state of a ledger is called consensus mechanism (Swanson, 2015).

The problem of finding a consensus first appeared in the literature in the early eighties with the Byzantine Generals Problem (Lamport, Shostak, & Pease, 1982). In a simplified version, the problem describes a scenario where several divisions of the Byzantine army, each commanded by its own general, circle an enemy city. Some of the generals want to attack whereas others prefer to retreat. In order for the attack to be successful, all the divisions must attack together. The challenge for the generals in this case is how to reach a consensus either to attack or to retreat, considering also the possibility that some of the generals may be traitors and may want to mislead the loyal ones. The Byzantine Generals Problem resembles the problem of nodes reaching a consensus in a distributed ledger environment.

Several consensus algorithms have been proposed so far for both permissioned and unpermissioned blockchains. This part will discuss six of these algorithms that are well-known and mostly used by blockchain systems nowadays, i.e. proof of work, proof of stake, practical byzantine fault tolerance, delegated proof of stake, ripple, and tendermint. Table 1 below, adapted from Zheng et al. (2017), provides a brief comparison of the examined consensus algorithms in terms of tolerated power of adversary.

Table 1: Security models of consensus algorithms
[Insert Table 1 Here]

	Ripple	**PoW**	**PBFT**	**Tendermint**	**DPoS**	**PoS**
Consensus process	unpermissioned	unpermissioned	permissioned	permissioned	unpermissioned	unpermissioned
Tolerated power of adversary	< 20% faulty nodes in a UNL	< 25% computing power	< 33.3% faulty replicas	< 33.3% byzantine voting power	< 51% validators	< 51% stake

Ripple is a consensus algorithm for unpermissioned blockchains introduced in 2014 by Schwartz, Youngs, and Britto (2014). Its core characteristic is that it circumvents the requirement for synchronous communication of the nodes by utilising collectively-trusted subnetworks called Unique Node Lists (UNLs). More precisely, the protocol defines two types of nodes, namely server nodes and client nodes. Server nodes have a set of servers in their UNL that allows them to participate in the consensus process, whereas client nodes can only transfer funds. Transactions broadcasted by a server (proposer) can be included in the consensus only when the vast majority (80%) of server nodes in its UNL confirms their validity. Therefore, the security of a blockchain adopting the Ripple protocol can be guaranteed only when the number of faulty nodes in each UNL is less than 20%.

Proof of Work (PoW) is the consensus mechanism underlying Bitcoin and it was described in the original paper by Nakamoto (2008). In PoW, nodes that want to generate a block have to prove they do not intend to attack the network by doing some work. This means they have to calculate the hash value of a block header that

contains a nonce, and reach a calculated value that is equal to or smaller than a target value provided by the network. When the work is finished, the block is broadcasted to all other nodes, which must agree on the correctness of the calculated hash value before the block can be added to the chain.

Practical Byzantine Fault Tolerance (PBFT) first appeared in the late nineties as a replication algorithm that can tolerate any number of Byzantine faults over a system's lifetime, provided that fewer than 1/3 of the replicas become faulty (Castro & Liskov, 1999). In PBFT, the process by which a new block is added to the chain is called a round, and it is divided into the three distinct phases pre-prepared, prepared and commit. In each round, the members of the chain follow a set of rules to select a different primary node, which is responsible for completing the transaction. The primary node needs to receive votes from a majority of 2/3 of all nodes to progress from one phase to the next until it completes the transaction. This consensus mechanism can only be applied to permissioned blockchains. There exist several other implementations of the Byzantine agreement like the stellar consensus protocol (Mazières, 2016), the delegated Byzantine fault tolerance (Zhang, 2016), and the scalable consensus protocol (Luu et al., 2015).

Tendermint is yet another protocol designed to offer Byzantine fault tolerance (Kwon, 2014). Like in the case of PBFT, the next block is determined in a round comprising the steps prevote, precommit, and commit. In the first step, validators have to prevote for a new block broadcasted by a designated proposer chosen in a round-robin fashion. If a prevote is broadcasted from a 2/3 majority of validators, the proposer proceeds to the second step and broadcasts a precommit for the new block. The third step can be entered only when the proposer receives precommits from more than 2/3 of the validators. In that case, the proposer broadcasts a commit after having validated the new block. Finally, the block is accepted if a commit vote is casted by more than 2/3 of the validators. In contrast to PBFT, only nodes that lock their coins can become validators.

Delegated Proof of Stake (DPoS) stems from the PoS protocol and was established in 2014 as the consensus mechanism underlying BitShares (BitShares community, 2014). The only difference between the two algorithms is that in the latter stakeholders can generate and validate new blocks themselves, whereas in the former this job is assigned to elected delegates. The smaller number of block validators improves the overall transaction speed significantly while the safety of the chain can be easily protected by voting out malicious delegates.

Proof of Stake (PoS) emerged in a Bitcoin forum post in 2011 as an alternative design to PoW consensus (QuantumMechanic, 2011), and was later adopted by peercoin (King & Nadal, 2012). In PoS, nodes that want to generate a block must hold a stake in the blockchain's currency, in order to prove their innocuous motives. This approach may seem unfair at first sight because nodes controlling the majority of stakes could dominate the network. However, there exist several solutions to this problem, e.g. randomising the prediction of the next block generator or selecting them based on the age of their stake. A more recent implementation of PoS is described in Kiayias, Russell, David, and Oliynykov (2017).

Smart Contracts

Another concept that has received considerable attention since the introduction of distributed ledger technology is that of smart contract. The term first appeared in the literature in 1997 as an attempt to combine protocols with user interfaces, in order to formalise, secure and execute contractual terms over computer networks (Szabo, 1997). With the benefit of trustless public ledgers, smart contracts have the potential to replace the current online contract law with a law of bargained-for exchange (Fairfield, 2014). In other words, smart contracts may disintermediate online exchange of money or other valuable assets (Kaplanov, 2012). Moreover, smart contracts may also replace human intermediation as a means of verifying the ownership of tangible or intangible property (Nofer et al., 2017), which represents a shift from trusting people to trusting math (Antonopoulos, 2014).

Recent research distinguishes two different types of definitions for the generic term smart contracts, i.e. smart contract code and smart legal contract (Stark, 2016). The first type of definitions is used to refer to code that is stored, verified and executed on a blockchain, and the capability of this code depends on the programming language used for its implementation. The second type of definitions considers ways of applying the specific code to complement or substitute legal contracts. Thus, the capability of smart legal contracts is proportional to their adoption by legal, political and commercial institutions, and does not depend on the underlying technology. For the sake of simplicity, the term smart contract will be used to refer to smart contract code in the remaining of this chapter.

Smart contracts can be broadly classified into two types: deterministic and non-deterministic (Morabito, 2017). Deterministic smart contracts do not require information stored outside their blockchain, in order to be triggered, make decisions, and work effectively. This means that all needed information is stored within their own blockchain environment. On the other hand, non-deterministic smart contracts need information about human behaviour, events or predictions, which is located outside the network that hosts their code. In this case, the information that enables smart contracts to make decisions about the data flow, is provided by an external party called oracle. Nevertheless, research on the subject suggests that the use of external state does not always introduce the need for trusting an oracle (Xu et al., 2016).

Several distributed ledger platforms that provide scripting languages for developing smart contracts have been identified in the literature, with the most notable being Bitcoin, Nxt and Ethereum (Seijas, Thompson, & McAdams, 2017). Bitcoin has a very limited scripting capability, and Nxt provides an application programming interface that delimits the execution of scripts on the client side. Ethereum, on the other hand, offers a virtual machine that runs a Turing-complete stack-based programming language, which allows the creation of more advanced and customised contracts for a hypothetically unlimited number of applications (Al-Khalil, Butler, O'Brien, & Ceci, 2017; Alharby & Moorsel, 2017).

Conceptual Model of the Proposed Architecture Framework

This chapter proposes a conceptual model of an architecture for storing and sharing learner models in the form of xAPI statements using the Ethereum blockchain platform. The framework regards learner models and their parts as assets belonging to learners, and thus the release of such information to third parties needs to be regulated by smart contracts. This approach builds on the xAPI Extended notion of PDL, where learner data from a learning record provider is initially stored on the respective PDL of each learner, which regulates the subsequent release to third-party LRSs and applications. However, PDLs are not implemented according to the xAPI specification. Their functionality is provided by a Decentralised Application (ĐApp), which runs on a P2P network and has its own suite of associated smart contracts that store xAPI statements on the Ethereum blockchain (Ethereum community, 2016b; Yedlin, 2017). ĐApps and smart contracts enable the creation of robust, versatile, cost effective, and low maintenance solutions for all kinds of applications that rely on a network and require some form of property rights management (Glaser & Bezzenberger, 2015). Figure 1 below shows the proposed framework for storing xAPI statements on the blockchain.

[Insert Figure 1 Here]
Figure 1. Proposed framework for storing xAPI statements on the blockchain

As depicted in Figure 1, the components of the framework are the two types of Ethereum accounts controlled by learners, content providers and a ĐApp, and an Ethereum blockchain. In the following, each of the framework parts, as well as the way they communicate and exchange information, will be reviewed starting with the accounts.

Accounts. There are two types of accounts in Ethereum, the Externally Owned Accounts (EOAs) and the Contract Accounts (CAs) (Ethereum community, 2016a). The former type can be controlled by either a learning content provider or a learner, and is used to send transactions, i.e. transfer wei (Ethereum currency) (Ethereum community, 2016c), or trigger smart contract code. The latter is controlled by a smart contract belonging to a ĐApp and has associated code stored on the blockchain. The execution of the code is triggered either by transactions send by EOAs or by other smart contracts. When a smart contract is executed, it manipulates its own state and may also call other contracts. The state of both types consists of a nonce, a balance, a storageRoot and a codeHash (Tikhomirov, 2017). Nonce stores the number of transactions sent by an EOA, or the number of contracts created by a CA. Balance holds the number of wei owned by an account, while storageRoot stores the Merkle Patricia tree root for an account's storage (Ethereum community, 2014). Finally, codeHash stores the hash of the contract bytecode for an account.

Blockchain. The sum of the local storage of the nodes forms a decentralised database (the blockchain) depicted as a grey rounded rectangle area in Figure 1. Apart from learners' data and public keys, the decentralised database stores smart contracts that regulate transactions, as well as the transactions themselves.

Smart contracts. They are used to regulate two types of transactions, namely create transactions and transfer transactions. The former type is used to register learner

data (xAPI statements) in the decentralised database, and also establishes in its outputs (in the form of a new smart contract) the conditions that must be met in order to transfer that data. These conditions may include a list of public keys that have full/partial control over the learner data, or may define that any transfer transaction must be digitally signed by an organisation and/or a learner. Transfer transactions check the validity of the conditions specified in a smart contract before transferring learner data to another node. These transactions must also specify, in a new smart contract, the transfer conditions that bind the new node. Figure 2 shows an example of a smart contract.

[Insert Figure 2 Here]

```
pragma solidity ^0.4.19;

contract Migrations {
  address public owner;

  // Define a function with the signature 'last_completed_migration()', which returns a uint
  uint public last_completed_migration;

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  function Migrations() {
    owner = msg.sender;
  }

  // Define a function with the signature 'setCompleted(uint)'
  function setCompleted(uint completed) restricted {
    last_completed_migration = completed;
  }

  function upgrade(address new_address) restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

Figure 2. A smart contract written in Solidity

Storing xAPI statements on the Blockchain. The process of storing a xAPI statement on the blockchain starts when a learning record provider sends a transfer transaction, which triggers a message from the LRP's EOA to their forwarding contract. Among other things, the message contains an xAPI statement (signed by the LRP) that describes the activity completed by the learner. The forwarding contract forwards the message to the statement contract, which checks the transfer conditions and forwards the message to the learner's forwarding contract. If the learner agrees with the content of the xAPI statement, they act as a validation oracle (human arbitrator) that signs valid transactions (Xu et al., 2016). The learner then sends a transfer transaction in form of a message from their EOA to their forwarding contract, which sends the message to the statement contract. The latter issues a create transaction after checking that both the LRP and the learner have signed the contract, and notifies them that the transaction has been approved. The validated transaction will be stored on the blockchain with the next mined block. The steps of this process are depicted in Figure 3.

[Insert Figure 3 Here]
Figure 3. Storing xAPI statements on the blockchain

Summary

This chapter discussed user model interoperability in education, and tried to address issues arising from the need to implement data mining and learning analytics responsibly. More precisely, it investigated the use of innovative and disruptive emerging technologies, like blockchain, and the Experience API as a means of regulating data ownership, access, and control, in order to increase the quality of and trust in learner data.

The Experience API standard has already received considerable attention in the literature. However, no previous study has investigated the combination of the Experience API with the blockchain technology. Learning process logs in the form of immutable Experience API statements are, in essence, provenance and thus can be used to form assessments about the data quality, reliability or trustworthiness. Trust in these statements can be further increased by storing them on a blockchain because the disintermediation property of blockchain eliminates the need for a trusted third party.

Recent research claims that the amount of effort required to develop a user model that holds a comprehensive set of attributes cannot be carried out by a single application (Dim & Kuflik, 2012). Moreover, existing learner data exchange approaches based on decentralised architectures do not fully leverage the advantages of sharing not only the control of the learner data but also the infrastructure that stores that data. Theoretically, in current decentralised systems, a sharing partner could exchange model parts with all other partners and own all shared models by having a local copy. In blockchain technology, ownership and control of learner models is regulated by smart contracts. Thus, even if a node has a copy of all learner models, it needs the authorisation of a smart contract in order to view or share that data. The literal sharing of infrastructure means that none of the sharing parties controls all of the shared data by itself. This is an inherent property of blockchain technology, which enables data sharing while at the same time allowing sharing partners to own only part of it.

The advantages of distributed ledger technology combined with the those of the Experience API could offer adaptive learning technologies a common data layer for creating better learner models in terms of interoperability, portability, security, privacy and trust. Future research could focus on existing issues like scalability and ways of linking different blockchains.

Chapter questions

1. How can user modelling and interoperability of user models enhance learning analytics?
2. What are the benefits of and challenges for interoperable learning systems?
3. In the context of personalised systems, what approaches have been proposed to enhance users' privacy and trust?
4. Why do the authors suggest the use of the Experience API standard for the creation of learner models?
5. What are the main benefits of using distributed ledger technology for storing learner models?

6. Which features/aspects of the Experience API does the use of a blockchain improve and how?

Listed References

Alharby, M., & Moorsel, A. van. (2017). Blockchain-based Smart Contracts: A Systematic Mapping Study. *Proceedings of the Fourth International Conference on Computer Science and Information Technology, United Arab Emirates*, 125–140. doi:10.5121/csit.2017.71011

Al-Khalil, F., Butler, T., O'Brien, L., & Ceci, M. (2017). Trust in Smart Contracts is a Process, As Well. *Proceedings of the 1st Workshop on Trusted Smart Contracts, Malta*, 74-83.

Antonopoulos, A. (2014). Bitcoin Security Model: Trust by Computation. *O'Reilly Radar.* Retrieved from http://radar.oreilly.com/2014/02/bitcoin-security-model-trust-by-computation.html

Aroyo, L., Dolog, P., Houben, G.-J., Kravcik, M., Naeve, A., Nilsson, M., & Wild, F. (2006). Interoperability in Personalized Adaptive Learning. *Journal of Educational Technology & Society*, *9*(2), 4–18. Retrieved from http://www.jstor.org/stable/jeductechsoci.9.2.4

Assad, M., Carmichael, D. J., Kay, J., & Kummerfeld, B. (2007). PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. In A. LaMarca, M. Langheinrich, & K. N. Truong (Eds.), *Lecture Notes in Computer Science: Vol. 4480. Pervasive Computing* (pp. 55–72). doi:10.1007/978-3-540-72037-9_4

Berkovsky, S., Eytani, Y., Kuflik, T., & Ricci, F. (2005). Privacy-Enhanced Collaborative Filtering. *UM05 Workshop on Privacy-Enhanced Personalization, Edinburgh, UK,* 75–83. Retrieved from http://isr.uci.edu/pep05/papers/w9-proceedings.pdf

Berkovsky, S., Kuflik, T., & Ricci, F. (2008). Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, *18*(3), 245–286. doi:10.1007/s11257-007-9042-9

U.S. Department of Education, Office of Educational Technology. (2012). *Enhancing teaching and learning through educational data mining and learning analytics: An issue brief*. Retrieved from https://tech.ed.gov/wp-content/uploads/2014/03/edm-la-brief.pdf

BitShares community. (2014). Delegated Proof-of-Stake Consensus [Technology Article]. Retrieved from https://bitshares.org/technology/delegated-proof-of-stake-consensus/

Brusilovsky, P., & Millán, E. (2007). User Models for Adaptive Hypermedia and Adaptive Educational Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *Lecture Notes in Computer Science: Vol. 4321. The Adaptive Web* (pp. 3–53). doi:10.1007/978-3-540-72079-9_1

Buterin, V. (2015, August 7). On Public and Private Blockchains [Web log post]. Retrieved from https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

Carmagnola, F., & Cena, F. (2009). User Identification for Cross-system Personalisation. *Journal of Information Science*, *179*(1–2), 16–32. doi:10.1016/j.ins.2008.08.022

Carmagnola, F., Cena, F., & Gena, C. (2011). User model interoperability: a survey. *User Modeling and User-Adapted Interaction*, *21*(3), 285–331. doi:10.1007/s11257-011-9097-5

Carmagnola, F., & Dimitrova, V. (2008). An Evidence-Based Approach to Handle Semantic Heterogeneity in Interoperable Distributed User Models. In W. Nejdl, J. Kay, P. Pu, & E. Herder (Eds.), *Lecture Notes in Computer Science: Vol. 5149. Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 73–82). doi:10.1007/978-3-540-70987-9_10

Castro, M., & Liskov, B. (1999, February). Practical Byzantine Fault Tolerance. In P. J. Leach & M. Seltzer (Chair), *Third Symposium on Operating Systems Design and Implementation*. Symposium conducted at the meeting of the USENIX Association, New Orleans, USA.

Cena, F. (2011). Integrating web service and semantic dialogue model for user models interoperability on the web. *Journal of Intelligent Information Systems*, *36*(2), 131–166. doi:10.1007/s10844-010-0126-3

Chrysafiadi, K., & Virvou, M. (2013). Student modeling approaches: A literature review for the last decade. *Expert Systems with Applications*, *40*(11), 4715–4729. doi:10.1007/s10844-010-0126-3

Cocea, M., & Magoulas, G. D. (2015). Participatory Learner Modelling Design: A methodology for iterative learner models development. *Information Sciences*, *321*, 48–70. doi:10.1016/j.ins.2015.05.032

Corbi, A., & Burgos, D. (2014). Review of current student-monitoring techniques used in elearning-focused recommender systems and learning analytics. The Experience API & LIME model case study. *International Journal of Interactive Multimedia and Artificial Intelligence*, *2*(7), 44–52. doi:10.9781/ijimai.2014.276

De Nies, T., Salliau, F., Verborgh, R., Mannens, E., & Van de Walle, R. (2015). TinCan2PROV: Exposing Interoperable Provenance of Learning Processes Through Experience API Logs. *Proceedings of the 24th International Conference on World Wide Web, USA,* 689–694. doi:10.1145/2740908.2741744

Dim, E., & Kuflik, T. (2012). User Models Sharing and Reusability: A Component-based Approach. *Workshop and Poster Proceedings of the 20th Conference on User Modeling, Adaptation, and Personalization*, *Montreal, Canada*.

Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., & Halevy, A. (2003). Learning to Match Ontologies on the Semantic Web. *The VLDB Journal*, *12*(4), 303–319. doi:10.1007/s00778-003-0104-2

Dolog, P., & Schäfer, M. (2005). A Framework for Browsing, Manipulating and Maintaining Interoperable Learner Profiles. In L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *Lecture Notes in Computer Science: Vol. 3538. User Modeling 2005* (pp. 397–401). doi:10.1007/11527886_52

Ethereum community. (2014). Merkle Patricia Trie Specification (also Merkle Patricia Tree) [Ethereum Homestead Documentation]. Retrieved from https://github.com/ethereum/wiki/wiki/Patricia-Tree

Ethereum community. (2016a). Account Types, Gas, and Transactions [Ethereum Homestead Documentation]. Retrieved from http://www.ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html

Ethereum community. (2016b). Dapps [Ethereum Homestead Documentation]. Retrieved from http://www.ethdocs.org/en/latest/contracts-and-transactions/developer-tools.html#dapps

Ethereum community. (2016c). Ether [Ethereum Homestead Documentation]. Retrieved from http://www.ethdocs.org/en/latest/ether.html

Fairfield, J. A. T. (2014). Smart Contracts, Bitcoin Bots, and Consumer Protection. *Washington and Lee Law Review Online*, *71*(2), 35–50. Retrieved from https://scholarlycommons.law.wlu.edu/wlulr-online/vol71/iss2/3/

Ghorbel, L., Zayani, C. A., & Amous, I. (2015). A Novel Architecture for Learner's Profiles Interoperability. In R. Lee (Ed.), *Studies in Computational Intelligence: Vol. 614*. Computer and Information Science 2015 (pp. 97–108). doi:10.1007/978-3-319-23467-0_7

Glaser, F., & Bezzenberger, L. (2015). Beyond Cryptocurrencies - A Taxonomy of Decentralized Consensus Systems. *Proceedings of the 23rd European Conference on Information Systems, Münster, Germany*. doi:10.18151/7217326

Graf, S., & Kinshuk. (2014). Adaptive Technologies. In J. M. Spector, M. D. Merrill, J. Elen, & M. J. Bishop (Eds.), *Handbook of Research on Educational Communications and Technology* (pp. 771–779). New York, NY: Springer.

Groth, P., & Moreau, L. (2013, April 30). PROV-Overview [W3C Working Group Note]. Retrieved from https://www.w3.org/TR/prov-overview/

Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., & Wilamowitz-Moellendorff, M. (2005). Gumo – The General User Model Ontology. In L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *Lecture Notes in Computer Science: Vol. 3538. User Modeling 2005* (pp. 428–432). doi:10.1007/11527886_58

Iyilade, J., & Vassileva, J. (2013). A Framework for Privacy-Aware User Data Trading. In S. Carberry, S. Weibelzahl, A. Micarelli, & G. Semeraro (Eds.), *Lecture Notes in Computer Science: Vol. 7899, User Modeling, Adaptation, and Personalization* (pp. 310–317). doi:10.1007/978-3-642-38844-6_28

Kaplanov, N. M. (2012). Nerdy Money: Bitcoin, The Private Digital Currency, And The Case Against Its Regulation. *Loyola Consumer Law Review*, *25*(1), 111–174. Retrieved from https://lawecommons.luc.edu/lclr/

Karoudis, K., & Magoulas, G. D. (2016). Ubiquitous Learning Architecture to Enable Learning Path Design across the Cumulative Learning Continuum. *Informatics*, *3*(4), 19. doi:10.3390/informatics3040019

Kay, J. (1999). *A scrutable user modelling shell for user-adapted interaction*. (Doctoral thesis, Basser Department of Computer Science, University of

Sydney, Australia). Retrieved from http://www.cs.usyd.edu.au/~judy/Homec/Pubs/thesis.pdf

Kay, J. (2008). Lifelong Learner Modeling for Lifelong Personalized Pervasive Learning. *IEEE Transactions on Learning Technologies*, *1*(4), 215–228. doi:10.1109/TLT.2009.9

Kay, J., & Kummerfeld, B. (2006). Scrutability, User Control and Privacy for Distributed Personalization. *Proceedings of the CHI2006 Workshop on Privacy-Enhanced Personalization, Canada* 21–22.

Kay, J., Kummerfeld, B., & Lauder, P. (2002). Personis: A Server for User Models. In P. De Bra, P. Brusilovsky, & R. Conejo (Eds.), *Lecture Notes in Computer Science: Vol. 2347. Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 203–212). doi:10.1007/3-540-47952-X_22

Kiayias, A., Russell, A., David, B., & Oliynykov, R. (2017). Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In J. Katz & H. Shacham (Eds.), *Lecture Notes in Computer Science: Vol. 10401. Advances in Cryptology – CRYPTO 2017* (pp. 357–388). doi:10.1007/978-3-319-63688-7_12

King, S., & Nadal, S. (2012). PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Retrieved from https://peercoin.net/assets/paper/peercoin-paper.pdf

Kobsa, A. (2007). Generic User Modeling Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The Adaptive Web: Methods and Strategies of Web Personalization* (Vol. 4321, pp. 136–154). Berlin, Heidelberg: Springer.

Kobsa, A., & Fink, J. (2006). An LDAP-based User Modeling Server and its Evaluation. *User Modeling and User-Adapted Interaction*, *16*(2), 129–169. doi:10.1007/s11257-006-9006-5

Kobsa, A., & Schreck, J. (2003). Privacy Through Pseudonymity in User-adaptive Systems. *ACM Transactions on Internet Technology*, *3*(2), 149–183. doi:10.1145/767193.767196

Kwon, J. (2014). Tendermint: Consensus without mining [Specification]. Retrieved from https://tendermint.com/static/docs/tendermint.pdf

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *4*(3), 382–401. doi:10.1145/357172.357176

Luu, L., Narayanan, V., Baweja, K., Zheng, C., Gilbert, S., & Saxena, P. (2015). SCP: A Computationally-Scalable Byzantine Consensus Protocol For Blockchains. Retrieved from https://eprint.iacr.org/2015/1168/20151214:030215

Martínez-Villaseñor, M. de L., González-Mendoza, M., & Danvila Del Valle, I. (2014). Enrichment of Learner Profile with Ubiquitous User Model Interoperability. *Computación y Sistemas*, *18*(2), 359–374. doi:10.13053/CyS-18-2-2014-037

Mazières, D. (2016, February 25). The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus (White Paper). Retrieved from https://www.stellar.org/papers/stellar-consensus-protocol.pdf

McCalla, G., Vassileva, J., Greer, J., & Bull, S. (2000). Active Learner Modelling. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Lecture Notes in Computer*

Science: Vol. 1839. *Intelligent Tutoring Systems* (pp. 53–62). doi:10.1007/3-540-45108-0_9

Mehta, B. (2007). Learning from What Others Know: Privacy Preserving Cross System Personalization. In C. Conati, K. McCoy, & G. Paliouras (Eds.), *Lecture Notes in Computer Science: Vol. 4511. User Modeling 2007* (pp. 57–66). doi:10.1007/978-3-540-73078-1_9

Mehta, B., Niederée, C., Stewart, A., Degemmis, M., Lops, P., & Semeraro, G. (2005). Ontologically-Enriched Unified User Modeling for Cross-System Personalization. In L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *Lecture Notes in Computer Science: Vol. 3538. User Modeling 2005* (pp. 119–123). doi:10.1007/11527886_16

Morabito, V. (2017). Smart Contracts and Licensing. In V. Morabito (Ed.), *Business Innovation Through Blockchain: The B^3 Perspective* (pp. 101–124). Cham, Switzerland: Springer.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

Niu, X., McCalla, G., & Vassileva, J. (2003). Purpose-Based User Modelling in a Multi-agent Portfolio Management System. In P. Brusilovsky, A. Corbett, & F. de Rosis (Eds.), *Lecture Notes in Computer Science: Vol. 2702. User Modeling 2003* (pp. 398–402). doi:10.1007/3-540-44963-9_56

Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, *59*(3), 183–187. doi:10.1007/s12599-017-0467-3

Normak, P., Pata, K., & Kaipainen, M. (2012). An Ecological Approach to Learning Dynamics. *Journal of Educational Technology & Society*, *15*(3), 262–274. Retrieved from http://www.jstor.org/stable/jeductechsoci.15.3.262

QuantumMechanic. (2011, July 11). Proof of stake instead of proof of work [Online forum comment]. Retrieved from https://bitcointalk.org/index.php?topic=27787.0

Schaffarzyk, H. (2015). The enhanced version of Experience API (xAPI Extended) [Open Source Project]. Retrieved from https://personal-data-locker.org/en/

Schwartz, D., Youngs, N., & Britto, A. (2014). The Ripple Protocol Consensus Algorithm (White Paper). Retrieved from https://ripple.com/files/ripple_consensus_whitepaper.pdf

Sclater, N., Peasgood, A., & Mullan, J. (2016). *Learning Analytics in Higher Education: A review of UK and international practice*. Retrieved from Jisc website: https://www.jisc.ac.uk/reports/learning-analytics-in-higher-education

Seijas, P. L., Thompson, S., & McAdams, D. (2017). Scripting Smart Contracts for Distributed Ledger Technology. *Proceedings of the 1st Workshop on Trusted Smart Contracts, Malta*. Retrieved from http://fc17.ifca.ai/wtsc/Scripting%20smart%20contracts%20for%20distributed%20ledger%20technology.pdf

Siemens, G. (2013). Learning Analytics: The Emergence of a Discipline. *American Behavioral Scientist*, *57*(10), 1380–1400. doi:10.1177/0002764213498851

Stark, J. (2016, June 4). Making Sense of Blockchain Smart Contracts [Online article]. Retrieved from https://www.coindesk.com/making-sense-smart-contracts/

Swanson, T. (2015). *Consensus-as-a-service: A Brief Report on the Emergence of Permissioned, Distributed Ledger Systems*. Retrieved from https://pdfs.semanticscholar.org/f3a2/2daa64fc82fcda47e86ac50d555ffc24b8c7.pdf

Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. *First Monday*, *2*(9). doi:10.5210/fm.v2i9.548

Tikhomirov, S. (2017). Ethereum: State of Knowledge and Research perspectives. In A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo, & J. Garcia-Alfaro (Eds.), *Lecture Notes in Computer Science: Vol. 10723. Foundations and Practice of Security* (pp. 206–221). doi:10.1007/978-3-319-75650-9_14

U.S. Department of Defence, Advanced Distributed Learning (ADL) Initiative. (2016). *xAPI-Spec* [Specification]. Retrieved from https://github.com/adlnet/xAPI-Spec/

U.S. Department of Defence, Advanced Distributed Learning (ADL) Initiative. (2017). xAPI Technical Specifications [Specification]. Retrieved from https://www.adlnet.gov/adl-research/performance-tracking-analysis/experience-api/xapi-technical-specifications/

Van Der Sluijs, K., & Houben, G.-J. (2006). A generic component for exchanging user models between web-based systems. *International Journal of Continuing Engineering Education and Life Long Learning*, *16*(1), 64–76. doi:10.1504/IJCEELL.2006.008918

Vassileva, J. (2001). Distributed User Modelling for Universal Information Access. In C. Stephanidis (Ed.), *Proceedings of the 9th International Conference on Human-Computer Interaction: Vol. 3. Universal access in HCI: Towards an information society for all* (pp. 122–126). New Orleans, USA: Lawrence Erlbaum.

Viviani, M., Bennani, N., & Egyed-Zsigmond, E. (2010). A Survey on User Modeling in Multi-Application Environments. *Proceedings of the Third International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services, Nice, France,* 111–116. doi:10.1109/CENTRIC.2010.30

Walport, M. (2015). *Distributed Ledger Technology: beyond block chain*. London, UK: Government Office for Science. Retrieved from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf

Walsh, E., O'Connor, A., & Wade, V. (2012). Evaluation of a Domain-aware Approach to User Model Interoperability. *Proceedings of the 23rd ACM Conference on Hypertext and Social Media, New York, USA,* 197–206. doi:10.1145/2309996.2310030

Walsh, E., O'Connor, A., & Wade, V. (2013). The FUSE domain-aware approach to user model interoperability: A comparative study. *Proceedings of the 14th*

International Conference on Information Reuse Integration (IRI), San Francisco, USA, 554–561. doi:10.1109/IRI.2013.6642518

Wang, Y., Cena, F., Carmagnola, F., Cortassa, O., Gena, C., Stash, N., & Aroyo, L. (2008). RSS-Based Interoperability for User Adaptive Systems. In W. Nejdl, J. Kay, P. Pu, & E. Herder (Eds.), *Lecture Notes in Computer Science: Vol. 5149. Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 353–356). doi:10.1007/978-3-540-70987-9_52

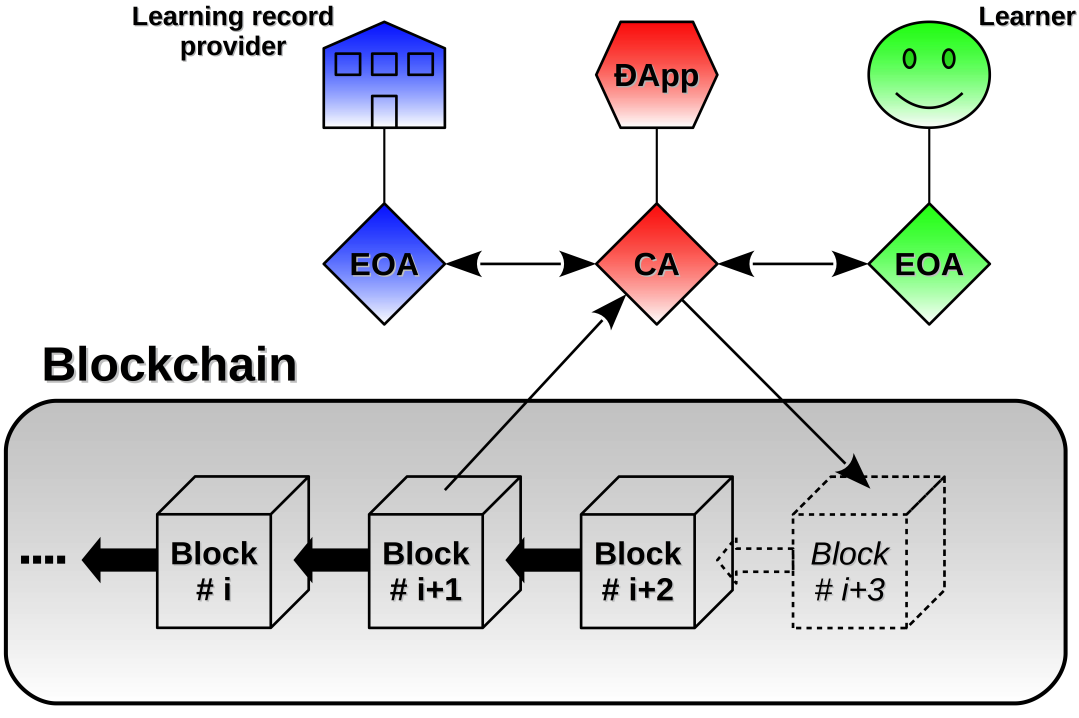
Wegner, P. (1996). Interoperability. *ACM Computing Surveys*, *28*(1), 285–287. doi:10.1145/234313.234424

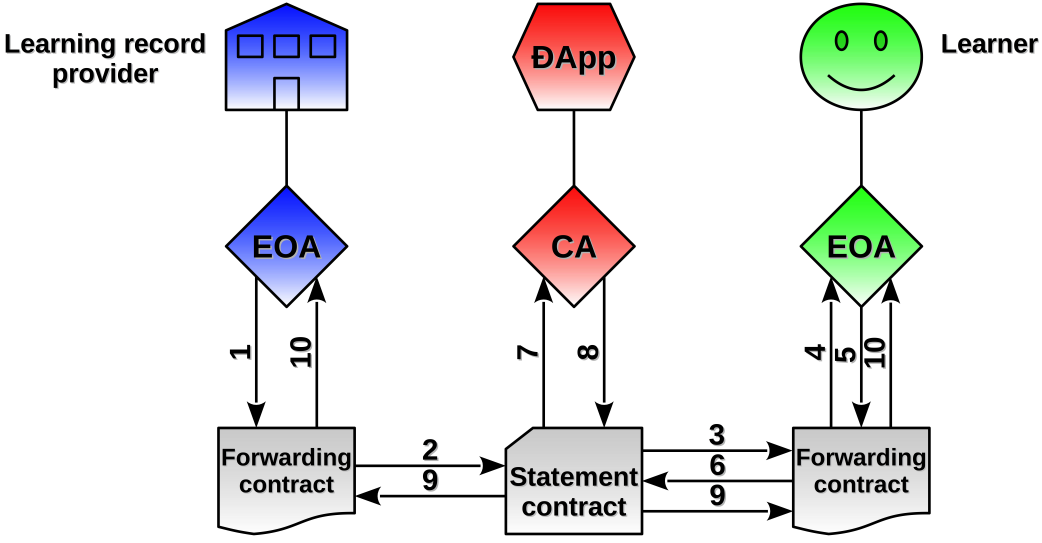
Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., & Chen, S. (2016). The Blockchain as a Software Connector. *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), Venice, Italy,* 182–191. doi:10.1109/WICSA.2016.21

Yedlin, R. (2017). What's a ĐApp? [Web article]. Retrieved from https://www.stateofthedapps.com/whats-a-dapp

Zhang, E. (2016). A Byzantine Fault Tolerance Algorithm for Blockchain (White Paper). Retrieved from http://docs.neo.org/en-us/node/whitepaper.html

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *Proceedings of the 2017 IEEE International Congress on Big Data, Honolulu, USA,* 557–564. doi:10.1109/BigDataCongress.2017.85


Learning record provider
ĐApp
Learner
EOA
CA
EOA
Blockchain
....
Block # i
Block # i+1
Block # i+2
Block # i+3


Learning record provider
ĐApp
Learner
EOA
CA
EOA
1
10
7
8
4
5
10
Forwarding contract
2
9
Statement contract
3
6
9
Forwarding contract