Aug 9th, 12:00 AM

A Blockchain-Inspired, Multi-Layered Transaction Model for Business Process Modeling

Kwok-Bun Yue
University of Houston-Clear Lake, yue@uhcl.edu

A Blockchain-Inspired, Multi-Layered Transaction Model for Business Process Modeling

Completed Research

Kwok-Bun Yue
University of Houston-Clear Lake
yue@uhcl.edu

Abstract

Software development for information systems can be characterized by the development of three kinds of models: (1) business models for business goals and requirements, (2) logical models for the solution design, and (3) physical models for actual implementation. The concept of transactions can beneficially be applied to these models. Past efforts for applying low-level transaction concepts to higher-level models focused on the atomicity, consistency, isolation, and durability (ACID) properties supported by traditional relational DBMS. We propose a novel transaction model that (1) refines the formalism of the ACID properties, and (2) adds blockchain-inspired formalism and properties including statefulness, privacy, and accountability. It supports transaction decomposition to sub-transactions, thus enabling multi-layered transactions. The proposed model is based on our experience on developing a proof-of-concept prototypical Model-Based Systems Engineering (MBSE) model repository using the blockchain platform Hyperledger's Fabric. A use case of the prototype is used as the illustrative example.

Keywords

Business Process Modeling, Blockchain, Business Model, Transaction Model, Model-Driven Software Development

Introduction

Software development can be characterized by the construction of a sequence of models with increasing fidelity for eventual implementation. In Model-Driven Software Development (MDSE), the active uses of models are elevated to a central role in the software development process (Brambilla, Cabot & Wimmer 2017) to examine, specify, design, simulate, implement, test, verify, deploy, and maintain the final systems (Cuadrado, Guerra & de Lara 2014). In information systems, the concept of transactions can be very beneficial for the development and transformation of these models. Business processes commonly involve transactional properties that need to be reliably assured and executed (Garcia et al. 2012).

Past efforts for applying low-level physical transactional concepts to business transactions in higher-level models focused on the atomicity, consistency, isolation, and durability (ACID) properties supported by traditional relational DBMS (Allweyer 2016, OASIS 2002). However, business transactions are more complex, span longer durations, involve more parties, and are usually implemented by multiple physical transactions. The semantic and requirement gaps between business transactions and physical transactions have not been addressed entirely satisfactorily. To bridge this gap, this paper proposes a multi-layered transaction model in which a transaction can recursively be decomposed into sub-transactions. Inspired by blockchain, the proposed Blockchain-Inspired Multi-Layered Transaction Model (BMTM) includes several useful blockchain transaction properties to serve as the basis of its formulation. The paper also re-examines the ACID properties within BMTM.

The remaining of the paper is organized in the following manner. The rest of this section provides a brief background on blockchain. BMTM is based on our experience on developing a proof-of-concept prototypical Model-Based Systems Engineering (MBSE) model repository using the blockchain platform Hyperledger's Fabric. This section also presents the necessary context of the blockchain prototype and elaborates a selected use case, which will serve as a unifying illustrative example for BMTM. The Business

Models, Processes, and Software Development section discusses the context and benefits of how a transaction model may fit in business process modeling and software development processes. The next section discusses the ACID properties and the basic formulation of BMTM. The Blockchain Transaction Properties section elaborates how selected blockchain properties can be used to scaffold and enhance BMTM. We draw our conclusions and discuss future research directions in the last section.

Blockchain Technology (BCT)

A blockchain is a distributed, decentralized, and highly tamper-resistant digital ledger of transactions stored identically in nodes of a distributed peer-to-peer network (Zheng et al. 2017). The *entire* history of all transaction records is stored in blocks cryptographically linked in a chain. Blockchains can be viewed as distributed *state* machines in which the most recent transaction data on all assets represent the current system state (Yue, et al. 2021a). A blockchain transaction changes the system from one consistent state to another consistent state. A block contains the hash of the previous block and a timestamp (Nakamoto 2019). Any retroactive tampering of a transaction will change the containing block, its hash, and thus all subsequent blocks in the chain, resulting in the invalidation of the chain. Consequentially, blockchains can also be considered as write-once, append-only storage systems. This tamper-resistant immutability allows users to effectively verify and audit transactions to ensure authenticity, and thus accountability.

Blockchain was proposed and first used as the backbone of Bitcoin (Nakamoto 2019). Bitcoin uses a simple, Turing-incomplete scripting language that supports only a narrow class of transactions: Bitcoin transfers and creations. Ethereum represents the next major enhancement in BCT by introducing *smart contracts* using a Turing-complete language called Solidity (Wood 2014, Yue 2020). Ethereum can thus be considered as a blockchain *platform* that supports complex transactions using smart contracts, opening up blockchains for wide applications.

Both Bitcoin and Ethereum belong to the class of public, permissionless blockchains open to public participation without any permission. Public blockchains are based on a trustless security model in which participants have no mutual trust (Zheng et al. 2017) and exceedingly strong security mechanism, such as the provision of Byzantine Fault Tolerance, must be supported (Sankar, Sindhu & Sethumadhavan 2017). As transaction data are openly accessible, privacy in public blockchains is limited. Participants are known to only have pseudonymity, not complete anonymity (Yue 2020). Thus, public blockchains are not suitable for many business cases where privacy and intellectual properties are key concerns.

As a result, permissioned blockchain platforms, such as Hyperledger's Fabric (Androulaki et al. 2018), Hyperledger's Sawtooth (Hyperledger 2021), and Corda (Valenta & Sandner 2017), started to appear. Participants need permissions to join permissioned blockchains with varying roles and privileges. These blockchain platforms provide varying membership service provisions to support nuanced privacy requirements needed by many business models and collaborations. Permissioned blockchains are thus used to support private blockchains and consortium blockchains for many business cases.

A Blockchain-Based MBSE Model Repository Prototype

Model-Based Systems Engineering (MBSE) is the "formalized application of modeling to support system requirements, design, analysis, verification and validation activities" (Hart 2015). In MBSE, models are developed using formal and executable modeling languages such as System Modeling Language (SysML) (Friedenthal, Moore & Steiner 2014) and UML (Larman 2012) to allow a large number of collaborating organizations to use the same language and system model to develop modern, large, complex and interdisciplinary systems-of-systems (Ramos, Ferreira & Barceló 2012). MBSE provides many significant advantages but requires effective management of an authoritative Single Source of Truth (SST) MBSE model repository with desirable features such as the provision of nuanced privacy models, decentralized governance policies, immutability, process automation, and performance (Yue, et al. 2021a). BCT can support many of these desirable features. Funded by NASA to investigate opportunities and challenges in applying blockchain on MBSE, our team built a proof-of-concept prototype using Fabric (with a secondary prototype covering a small subset of functionality built using Sawtooth) to implement a number of use cases identified and specified by the MBSE lead of a NASA aeronautical project (Yue, et al. 2021a). MBSE artefacts are stored as blockchain assets in the prototype. To help exploring blockchain opportunities and

resolving many challenges in applying BCT on MBSE, the team developed BMTM. It is thus appropriate to employ a use case in our prototyping experiment as an exemplary example.

A MBSE Use Case, UC1

In a large-scale MBSE project, an SST model repository stores MBSE artefacts: models, sub-models, components, sub-components, etc. A consortium of organizations collaborates on model development and usually has a lead organization, C (e.g., NASA). Participating organizations have nuanced semi-trust relationships with each other to protect their intellectual properties, as they may also be competitors elsewhere. In this use case, named UC1, organization O_1 is responsible for the development of the artefact A (e.g., an SysML model). The most recent *approved version* of A in the repository is V2.1. Organization O_2 is permitted to access all approved versions of A as O_2 may need it to develop other components. On the other hand, A may not be visible to organization O_3 at all. To evolve A to the next approved version, such as V2.2, the associated business process (called BT1) goes through several steps.

1. Internal development by O_1: the intermediate *development versions*, such as V2.1.5.1 and V2.1.5.2 in Figure 1 are accessible only by permitted users within O_1 to protect its intellectual property. Not even the lead organization C can access these versions.
2. Collaboration between O_1 and C: When a newer version of A is deemed ready by O_1, O_1 starts the collaboration process with C so that C can ensure that the newer version satisfies the proper requirements and is correctly integrated. The sequence of *collaborative versions* (e.g., V2.1.6 and V2.1.7) can be accessed only by the collaborators O_1 and C.
3. Approval process: When ready, the resulting version is submitted by C to an approval board (AB) for approval. Changes can be made and the intermediate versions (e.g., V2.2.a.1 and V2.2.a.2) can be accessed only by C and AB.
4. Approval: When A is finally approved by AB, the newly approved version V2.2 is accessible to C, O_1, O_2 and AB, but not O_3.

Figure 1 summarizes the version changes with the organizational visibility of UC1. These varying versions may be scattered (as in the legacy system) or collectively stored in a repository (as in our prototype).

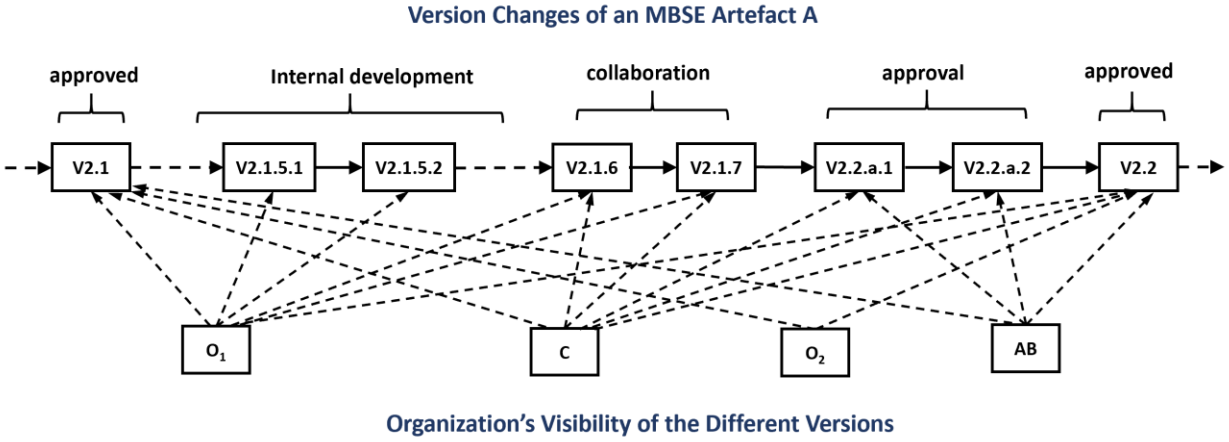

Figure 1 Version Changes of a MBSE Model Repository's Example, UC1

Business Models, Processes, and Software Development

In information systems, software applications are developed to support business functions and processes, which in terms are used to support business goals and strategies. From the perspective of MDSE, the software development process is the construction of a sequence of models with increasing fidelity for eventual implementation (Brambilla, Cabot & Wimmer 2017). Figure 2 shows the usual classifications of these models into a three-layered architecture with business models, logical models, and physical models.

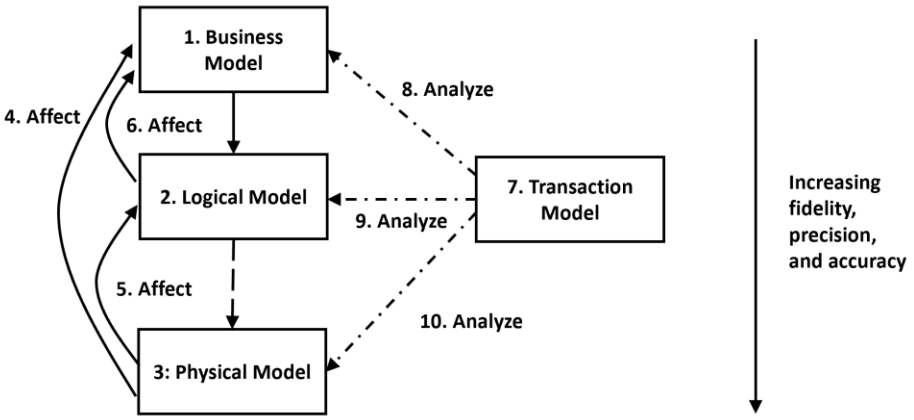

Figure 2 Transaction Models and a Three-Layered Model Architecture

In this paper, we use the term business model (Box 1 in Figure 2) from the perspective of software development to contain the business and requirement aspects of the desirable information systems: from business strategies, business models (Osterwalder, Pigneur & Tucci 2005), business goals (Henderson & Venkatraman 1999), to business functions and process requirements that satisfy the goals (de Oca, et al. 2015). Accordingly, business models are relatively vague and coarse-grained as they focus on high level strategies and goals. For example, in the use case UC1, a business model may call for having a business process/transaction (BT1 above) to approve and update the version of the artefact A, such as from V2.1 to V2.2. The middle logical model (Box 2 in Figure 2) is mainly for the design of solutions. For example, in UC1, a logical transaction (named LT1) for a solution design for the collaboration sub-process between O_1 and C may call for the communications of the MBSE artefact A through email, and the storage and management of the intermediate versions of A locally by C (this solution is used by the existing system.) The lowest physical level (Box 3 in Figure 2) is the actual model used to physically implement the logical model. In the implementation of LT1, C uses MagicDraw (Nomagic 2021), a popular modeling tool, for locally managing the various versions of A in the existing system.

The boundaries of these models may not be clear cut and they can overlap. Many different formal and informal modeling languages (and programming languages and tools for the physical model) may be used. The problem of the transformations of higher-level models to lower-level models (Cuadrado, Guerra & de Lara 2014) can be difficult as each model may have its own goals, constraints, languages, patterns, and best practices. Conversely, features and constraints of the lower-level models can and should also affect decisions in constructing higher-level models (Edges 4, 5 and 6 in Figure 2).

Three major reasons making object-oriented modeling (OOM) with UML (Larman 2012) a popular general-purpose methodology are (1) OOM provides a language and methodologies that can apply to all three levels of models, thus facilitating transformations and feedbacks between models, (2) UML provides varying degrees of fidelity to suit different phases of the software process and different levels of modeling, and (3) UML is flexible and extensible for different domains. The concepts of transaction are important in all levels of models, but they usually have different meanings. In the coming sections, we propose a common transaction model (Box 7 in Figure 2) that can be used to analyze all levels of models (Edges 8, 9 and 10 in Figure 2). Furthermore, the model, BMTM, is also flexible and supports varying fidelity.

ACID Properties and BMTM

The studies of transactions related to business processes have been researched for a long time. For example, properties of workflow transactions have been investigated, including those related to the data consistency properties as provided by databases (Georgakopoulos, Hornick & Sheth 1995). The ACID transaction model of traditional relational databases is a physical transaction model in which the DBMS guarantees a set of four properties for concurrency control (Gray & Reuter 1992):

1. Atomicity: all or none of the operations of a transaction are completed.
2. Consistency: the database preserves its consistency after a transaction.

3. Isolation: transient internal states within a transaction should be isolated. The effect of a transaction is not visible to other concurrent tasks until the transaction is completed.
4. Durability: database changes as a result of a transaction completion are permanent.

ACID transactions support effective concurrency control and recovery in the low, physical implementation level, preserving data consistency, and making modeling in the higher level easier. The ACID transaction model can gainfully be adapted in many directions for higher-level models as well as different domains. For example, NoSQL database is an attempt to improve performance and scalability by modifying it to a Basically Available, Soft state, Eventual consistency (BASE) model (Nayak, Poriya & Poojary 2013). Likewise, ACID properties have been used to establish a business-aware Web services transaction model (Papazoglou & Kratz 2006), the Business Transaction Protocol (BTP) for higher logical modeling (OASIS 2002), or a business process model (Garcia et al. 2012).

Since the goals and requirements of higher-level models and domains are different, the low-level ACID properties may need to be adapted, relaxed, or enhanced. For an example, the traceability and correct execution properties are also supported by a transaction model for business process systems (Garcia et al. 2012). Likewise, the *all-or-nothing* atomicity property is relaxed for a business-aware transaction model (Papazoglou & Kratz 2006) in which business-level atomicity criteria can be classified as vital (always included in a transaction) or non-vital (can be replaced by contingent activities when fail).

The ACID model has also been adapted for Business Process Modeling (BPM). OASIS developed BTP to manage business interactions in loosely coupled, asynchronous environments such as the Internet (OASIS 2002). Comparing to ACID transactions, the logical transactions required by the business models usually span much longer durations (e.g., a customer may have 24 hours to complete the purchase of a reserved air ticket) and the partial result may need to be somewhat visible (e.g., for the customer to know that the air ticket reservation has been successful). Thus, the transaction model of BTP is based on relaxed and modified ACID properties (OASIS 2002). BTP transactions distinguish between two kind of behaviors: (1) BTP atomic behaviors in which ACID properties are only slightly relaxed, and (2) cohesive behaviors in which ACID properties are more significantly relaxed to accommodate business needs. These ACID properties of these two types of behaviors are shown in columns 2 and 3 of Table 1. Since isolation is one particular setting value of visibility, Table 1 states 'isolation' as 'isolation/visibility.'

ACID Property	**BTP Atoms**	**BTP Cohesions**	**BPMN**	**BMTM**
Atomicity	Yes	Negotiated	No	Yes, but layered
Consistency	Yes	Yes, but nuanced	No	Yes
Isolation/Visibility	Relaxed, controlled by the service	Relaxed, controlled by the service	No	Yes, but layered
Durability	Yes	Mostly, but some may be volatile	No	Yes

Table 1. Built-in Support of ACID Properties of Some Business Process Models

Note that for BTP atoms, isolation in BTP is relaxed and the visibility of transactional effects are controlled by the service provider. Although this relaxation provides the needed business flexibility, it has two disadvantages: (1) it is too flexible and ad hoc to provide consistency and to serve as a universal guideline, and (2) the service provider can be regarded as a central authority, which is unfavored in the increasingly decentralized landscape. These two disadvantages are more serious for BTP cohesions.

Transactions are also a key concept in business modeling languages such as the popular Business Process Modeling and Notation (BPMN) (Allweyer 2016). BPMN supports a business transaction model in which a transaction is only a special kind of sub-processes depicted with double line borders (Polančič 2013). Comparing to other regular tasks, a BPMN transaction just has stronger relationships and constraints. A BPMN transaction is typically implemented by a number of ACID transactions with the intermediate results visible to selected actors. Specifying ACID properties of sub-tasks of BPMN transactions are done through additional BPMN elements, notation extensions, and/or methods external to BPMN. Thus, as shown in Table 1, there is no intrinsic, direct support of ACID properties of BPMN transactions.

In summary, although business transactions may eventually be implemented using ACID transactions, past efforts in applying ACID properties to strengthen business transaction models are not entirely satisfactory. The conceptual gaps between business transactions and ACID transactions are large and they have not been rigorously investigated and bridged. BMTM is an attempt in this direction. In BMTM, a transaction T is characterized by its properties P(T) (Figure 3). These properties may contain the ACID properties as well as other properties (such as statefulness, privacy, accountability, and immutability) that are rooted from other perspectives (such as blockchains) and beneficial to the targeted applications.

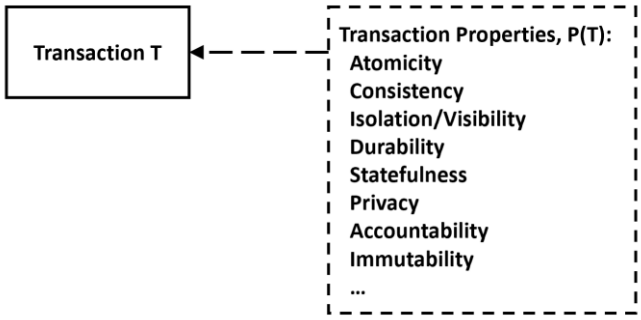

Figure 3 Transaction Properties

In BMTM, transactions can be recursively decomposed into many levels of sub-transactions, each with different properties. For example, in Figure 4, the transaction T1 is decomposed into sub-transactions T1.1, T1.2,...,T1.n, with properties P(T1.1),P(T1.,2),...,P(T1.n) respectively. This forms the basis of a multi-layered transaction model in which a leaf node in the transaction tree may be directly implemented as a single physical transaction, such as an ACID transaction.

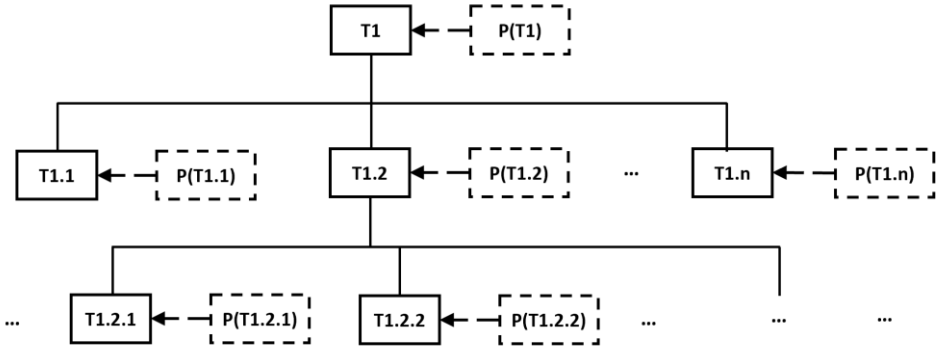

Figure 4 BMTM: A Multi-Layered Transaction Model

In BMTM, the satisfaction of ACID properties is relative to the perspectives of the selected transactions or sub-transactions. As seen in Table 1, although consistency and durability are always maintained, the atomicity and isolation properties depend on the perspective and the actor. This will be further elaborated in the next section.

Blockchain Transaction Properties

As BCT becomes more mature, research works on the transaction models of blockchain begin to appear. For example, there are some efforts on applying the transaction processing perspective to understand blockchains, such as characterizing blockchains with a Sequential, Agreed-on, Ledgered, and Tamper-resistant transaction processing (SALT) model (Tai, Eberhardt & Klems 2017). There are also efforts to implement existing transaction models on blockchains (e.g., Brahem et al. 2020). However, there is a lack of literature on applying the unique properties of blockchain transactions on enhancing the transaction models for software development processes and business process modeling. An exception is the limited recent research works on using aspects of blockchain transaction properties to strengthen existing transaction models in the domain of distributed energy. These blockchain enhancements include a cyber-

enhanced transactive microgrid model for energy trading (Okoye et al. 2020) and a peer-to-peer (P2P), Industrial Internet of Thing (IIoT) direct distributed energy transaction model (Hu & Li 2020). This section presents how blockchain properties are used to scaffold BMTM, which is a general transaction model not limited to a particular domain or physical implementation. (BMTM models may be implemented without using blockchains.) This paper focuses on three blockchain properties: statefulness, privacy, and accountability.

Statefulness

As a distributed state machine, the concept of states plays a central role in blockchain. Many blockchain platforms have operations such as PutState(A) in their APIs to put an asset A (usually represented as a key-value pair, possibly in JSON format) into the blockchain. The most recent value of A becomes a part of the current system state. Thus, a blockchain transaction alters the system state, such as by executing a PutState operation. As distributed ledgers of *all* transactions, blockchains store all historical states in blocks. To improve performance, many blockchain platforms also store a copy of the *current* system state in a synchronized database. For example, Fabric stores the current system state in the World State database, which usually uses LevelDB or CouchDB. An operation such as GetState(A) in the API can be used to retrieve the current state of A much faster through the World State database. The immutability property of blockchain means that there is no update operation such as UpdateState(A) in the APIs. Instead, the updated content of A is stored in entirety to the blockchain by using PutState(A), even when the delta change is small.

As a comparison, even though the whole state may be considered to be the entirety of the current database instance, it is a secondary concept in a traditional relational database. The current value of an asset A in traditional databases is possibly scattered in multiple tables, and is not focally managed. To emphasize the statefulness property, BMTM recommends:

R1. Careful investigation of proper definition and representation of *all* possible consistent states to support the models and processes in Figure 2. In the UC1 example, states corresponding to all different versions of the artefact A in Figure 1 should be defined and specified. This ensures the consistency property of BMTM as shown in Table 1.

R2. Persistent storage of *all* new consistent states. This ensures the durability property of BMTM as shown in Table 1. If blockchains are used in the actual implementation model, the entire history of consistent states is immutably stored. Even if traditional database is used, storing all new consistent states can easily be designed (without the historical states automatically stored.)

R3. Inclusion of state's meta-data. State information of an asset A can be captured as meta-data of A. The meta-data can be stored as an integral part of A to facilitate focused management.

Borrowing from blockchains, state representations can be captured through the specification of two functions on an asset A: WriteState(A) and ReadState(A). (We do not employ the more popular low-level names of PutState and GetState used by many blockchain APIs to avoid confusion.) For example, for the business transaction BT1 in UC1 (update the approved version of A from V2.1 to V2.2) and the logical transaction LT1 (update the collaborative version of A between O_1 and C from V2.1.6 to V2.1.7, in which the intermediate versions do not require the approval by AB) may be modeled respectively as:

BT1: WriteState(A): {A.version: V2.1 -> V2.2, A.approved: yes, ...}
LT1: WriteState(A): {A.version: V2.1.6 -> V2.1.7, A.approved: no, ...}

Both the attributes version and approved status can be considered as state's meta-data and are stored as an integral part of A (recommendation R3 above). There are many formal and informal methods and languages for specifying states to choose from. Meta-data can be defined to satisfy specific needs and varying degrees of details can be captured. This focus on state representations and changes thus provides BMTM with the desirable varying fidelity and flexibility properties of a general-purpose methodology.

Other transaction properties can also be modeled around state representations using ReadState and WriteState, as illustrated by the discussion on the privacy and accountability properties below.

Privacy

Privacy is basically concerned with who can read what. In our example of UC1, it can be formulated through the visibility of the artefact A by various actors/organizations as defined by the returned value of calling ReadState(A). The isolation property of ACID, which serves a different purpose (i.e., concurrency control), can also be understood through having a null visibility of all internal transient states within a transaction when read by other concurrent tasks. To exemplify these issues, Table 2 shows the versions of A returned by ReadState(A) for various organizations in UC1, before, during, and after BT1 (which updates the version from V2.1 to V2.2). It is based on the state change diagram in Figure 1.

Organization	**Before BT1**	**During BT1**	**After BT1**
C	V2.1	V2.1.6, V2.1.7, V2.1.a.1, or V2.1.a.2	V2.2
O_1	V2.1	… V2.1.5.1, V.2.1.5.2, …, V2.1.6 or V2.1.7	V2.2
O_2	V2.1	V2.1	V2.2
O_3	null	null	null

Table 2 Version of A Returned by ReadState(A)

To elaborate, the privacy requirements of UC1 require that O_3 has no visibility of A. BT1 is not atomic to C or O_1 and thus some intermediate states are not isolated from them. Thus, ReadState(A) returns the most recent intermediate states visible to them respectively. However, unlike intermediate transient, and possibly inconsistent, states within ACID transactions that should be isolated, recommendations R1 and R2 ensure designing BMTM states to be consistent and persistently stored. By contrast, BT1 is atomic from the perspective of O_2 and the intermediate versions are isolated from it. Thus, in BMTM, the atomicity and isolation properties are relative to the transactions, privacy requirements, and organizations. As a result, Table 1 describes and qualifies them as "Yes, but layered." However, in the lowest level transactions in BMTM, ACID properties are usually expected to be fully supported without qualifiers to preserve proper concurrency control.

If more fidelity is needed in a lower-level logical model, the privacy and visibility properties can be further refined by using additional meta-data and constructs, such as in this example by using accessibility matrices for BT1 as the design solution:

Meta-data: state_type: {developing, collaborating, approving, approved}
E.g. state_type of approved: V2.1, V2.2; state_type of collaborating: V2.1.6, V.2.1.7; etc.
Construct: organization_type_accessibility_matrix: which organizations can access what types of states.
E.g. C: {collaborating, approving, approved}, O_1: {developing, collaborating, approved}
ReadState(A) refined to ReadState(O, A, S): returns the latest version of A of state_type S visible to organization O.
E.g. During BT1: ReadState(O_1, A, approved): V2.1; ReadState(O_1, A, developing): V2.1.5.1, etc.;
ReadState(O_1, A, approving): null; ReadState(O_2, A, collaborating): null; etc.

In the physical model, meta-data should be a part of the asset stored. Other constructs may be implemented in different ways. For example, the accessibility matrix can be implemented by using the membership service provision in Fabric.

Accountability

Accountability is concerned with who have written what. By managing an immutable ledger of all transactions, blockchains support accountability exceedingly well. In BMTM, the accountability property of a transaction can be specified to various degree of details by the WriteState function. For example, in the collaboration sub-process of UC1, the logical model may call for a design solution of not using email for communications (as in LT1). Alternatively, it may use a different logical transaction, LT2, to store collaborating versions, such as V2.1.6, into a model repository, with the authorization of both collaborating organizations (O_1 and C) recorded for accountability and auditing. In the coarser level, LT2 can be described by refining the function WriteState(A) to WriteState(O, A) in which O is the list of organizations authorizing the transaction. A proper call may look like WriteState({O_1, C}, A) in LT2.

If more details and security control are needed, the following logical model using additional meta-data and constructs can serve as a possible design example:

Meta_data: consensus_method: {all, majority, …}
E.g. A.conensus_method: all (i.e. all endorsing organizations must authorize a transaction on A.)
Construct: endorsing_organzations(A,S): the organizations that participate in authorizing writing A with state_type S.
E.g. endorsing_organzations(A): {O_1, C}
WriteState(A) -> WriteState(O, A): the organizations O attempt to write A.
E.g. for A with state_type collaborating: WriteState({O_1}, A): failure; WriteState({C}, A): failure; WriteState({O_1, C}, A): success (endorsements of both O_1 and C are needed to write A); etc.

Again, depending on the selected blockchain platforms, there may be various means to implement the constructs and the WriteState function. For example, the membership service provision in Fabric can be used to check whether the consensus methods are satisfied by the submitting organizations, and WriteState can be implemented by multiple-signature transactions.

Conclusion

In this paper, based on our experience in building a proof-of-concept blockchain prototype of a MBSE model repository for a NASA project in Fabric, we present the framework a novel blockchain-inspired multi-layered transaction model with some potential advantages, and illustrate it with ACID properties as well as three blockchain transaction properties: statefulness, privacy, and accountability. There are other blockchain properties not discussed in this paper because of space. Transactions are an important component in business processes and governance. Readers interested in how these transaction properties may interact with various governance attributes may refer to (Yue, et al. 2021b). In the future, BMTM needs to be improved, refined, and further field-tested, which we plan to do in Phase 2 of building a research-and-development prototype that is more production-ready. As a transaction model, BMTM can be incorporated into existing modeling languages and tools. The team is working on extending BPMN notation to accommodate BMTM. Another research direction is constructing a more systematic way to specify state representations that can better be transformed to blockchain platforms. The team also plans to use BMTM to investigate related issues in the business level, such as governance (Yue, et al. 2021b).

Acknowledgement

This work is partially supported by NASA's STTR program: 80NSSC19C0525 and 80NSSC21C0020.

REFERENCES

Allweyer, T. 2016. *BPMN 2.0: introduction to the standard for business process modeling*. BoD–Books on Demand.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., and Muralidharan, S. 2018. "Hyperledger Fabric: a distributed operating system for permissioned blockchains," *Proceedings of the Thirteenth EuroSys Conference, 2018*, pp. 1-15.

Brahem, A., Messai, N., Sam, Y., Bhiri, S., Devogele, T., and Gaaloul, W. 2020. "Running Transactional Business Processes with Blockchain's Smart Contracts," *2020 IEEE International Conference on Web Services (ICWS),* pp. 89-93.

Brambilla, M., Cabot, J., and Wimmer, M. 2017. "Model-driven software engineering in practice," *Synthesis lectures on software engineering*, (3:1), pp.1-207.

Cuadrado, J. S., Guerra, E., and de Lara, J. 2014. "A component model for model transformations," *IEEE Transactions on Software Engineering*, (40:11), pp.1042-1060.

de Oca, I. M. M., Snoeck, M., Reijers, H. A., and Rodríguez-Morffi, A. 2015. "A systematic literature review of studies on business process modeling quality," *Information and Software Technology*, (58), pp.187-205.

Friedenthal, S., Moore, A., and Steiner, R. 2014. *A practical guide to SysML: the systems modeling language*, Morgan Kaufmann.

Garcia, O., Braghetto, R., Pu, C., and Ferreira, E. 2012. "An implementation of a transaction model for business process systems," *Journal of Information and Data Management*, (3:3), pp.271-271.

Georgakopoulos, D., Hornick, M., & Sheth, A. 1995. "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and parallel Databases*, (3:2), pp.119-153.

Gray, J. and Reuter, A. 1992. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Hart, L. E. 2015. "Introduction to model-based system engineering (MBSE) and SysML," Delaware Valley INCOSE Chapter Meeting, (30), Mount Laurel, New Jersey.

Henderson, J. C. and Venkatraman, N. 1999. "Strategic alignment: Leveraging information technology for transforming organizations," *IBM Systems Journal, IBM Corporation/IBM Journals*, (38), p.472.

Hu, W., and Li. H. 2020. "A blockchain-based secure transaction model for distributed energy in Industrial Internet of Things," *Alexandria Engineering Journal,* (60:1), pp.491-500.

Hyperledger, 2021. "Hyperledger's Sawtooth," https://www.hyperledger.org/use/sawtooth, retrieved 1/5/2021.

Larman, C. 2012. "Applying UML and patterns: an introduction to object oriented analysis and design and interactive development," Pearson Education India.

Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," Manubot, 2019.

Nayak, A., Poriya, A., and Poojary, D. 2013. "Type of NOSQL databases and its comparison with relational databases," *International Journal of Applied Information Systems*, (5:4), pp.16-19.

Nomagic, 2021. "MagicDraw's User Guide," https://docs.nomagic.com/display/MD190SP4/User+Guide, retrieved 1/5/2021.

OASIS (Organization for the Advancement of Structured Information Systems), 2002, Business Transaction Protocol, https://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf, accessed: 1/4/2021.

Okoye, M., Yang, J., Cui, J., Lei, Z., Yuan, J., Wang, H., Ji, H., Feng, J., and Ezeh, Z. 2020. "A blockchain-enhanced transaction model for microgrid energy trading." *IEEE Access,* (8), pp.143777-143786.

Osterwalder, A., Pigneur, Y., and Tucci, C. L. 2005. "Clarifying business models: Origins, present, and future of the concept," *Communications of the Association for Information Systems*, (16:1), p.1.

Papazoglou, M. P., and Kratz, B. 2006. "A business-aware web services transaction model," *International Conference on Service-Oriented Computing,* Springer, Berlin, Heidelberg, pp. 352-364.

Polančič, G. 2013. "Modeling Transactions with BPMN 2.0," http://orbus.blob.core.windows.net/media/2005328/wp0077_modeling-transactions.pdf, accessed: 1/4/2021.

Ramos, A.L, Ferreira, J.L., and Barceló, J. 2012. "Model-Based Systems Engineering: An Emerging Approach for Modern Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, (42:1), pp. 101-111. doi: 10.1109/TSMCC.2011.2106495.

Sankar, L.S., Sindhu, M. and Sethumadhavan, M. 2017. "Survey of consensus protocols on blockchain applications," *4th International Conference on Advanced Computing and Communication Systems (ICACCS),* pp. 1-5.

Tai, S., Eberhardt, J., and Klems, M. 2017. "Not acid, not base, but salt," *Proceedings of the 7th International Conference on Cloud Computing and Services Science,* SCITEPRESS-Science and Technology Publications, Lda, pp. 755-764.

Valenta, M., and Sandner, P. 2017. "Comparison of ethereum, hyperledger Fabric and corda," Frankfurt School, Blockchain Center.

Wood, G. 2014. "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, No. 151, pp.1-32.

Yue, K., "Blockchain-Augmented Organizations, "*Proceedings of the 26th Americas Conference on Information Systems AMCIS*, 5.

Yue, K., Guerra, M., Wagner, H., Thamarai Selvan, J.S., Collector, K., Tang, V., Sikes, M., Sha, K., Kallempudi, P., Mardani, S. & Kasichainula, K. 2021a. "Applying Blockchain Technology on Model-Based Systems Engineering," *AIAA Scitech 2021 Forum,* p. 0093.

Yue, K., Kallepudi, P., Sha, K., Wei, W., and Liu, X. 2021b. "Governance Attributes of Consortium Blockchain Applications," to appear in *Proceedings of the 27th AMCIS*.

Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. 2017. "An overview of blockchain technology: Architecture, consensus, and future trends," *2017 IEEE International Congress on Big Data (BigData Congress),* pp. 557-564.