Towards a Shared Ledger Business Collaboration Language based on Data-Aware Processes

--- or ---

What does it mean for Services Research if all parties Share a single Database?

Richard Hull (IBM Research)
In collaboration with:
  Vishal S. Batra, Yi-Min Chee, Fenno F. Terry Heath III (IBM Research)
  Alin Deutsch, Victor Vianu (UCSD)

And drawing on many discussions with: Yao Liang Chen, Pralhad Deshpande, Yuichi Nakamura, Krishna Ratakonda, Jianwen Su, Noi Sukaviriya, Shin Saito, Takaaki Tateishi

11 August 2016 @ ICSOC in Banff, Canada
Version v10
How do organizations collaborate in today’s world?

By exchanging documents, in many cases on paper:

• Trade finance: letter of credit, export documents (eg., SWIFT MT700,...)
• Logistics/Supply Chain: Purchase Order (EDI 850), Load Tender (EDI 204), Tender Response (EDI 990), ...
• Mortgage & Loan processing: many scanned PDF’s
• ...

Are these simply messages exchanged between services?
• No, because they persist, and are referred to at later times
Blockchain (for businesses) will dramatically streamline data/document sharing

- Blockchain provides a trusted repository for holding persistent data
- Blockchain enables selective privacy
- Blockchain will enable deep efficiencies

How will this seismic shift in business collaboration impact the Services Research Community?
One broad area for Services Research contributions:

**Business-Level Language and Framework**

- Blockchain today is programmed using Turing-complete languages such as GOLANG, Java, ???
- Some domain-specific languages are emerging ...

We need

- *Principled approach* for data-centered services & collaborations
- *Domain-specific language* aimed at business users
- *Workbenches for business analysts* to understand, create, test, modify the “smart contracts” that run on Blockchain
- *Foundational understanding* of biz-level “smart contracts”
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
- Artifact-centric paradigm as starting point for Business Collaboration Language
- Research challenge areas
  - Language design
  - Reasoning about artifacts
  - Relationship to natural language contracts
- Conclusions

Caveat

This field is still in its infancy
This talk is mainly raising questions
Example from International Trade Finance

- Suppose that a company in Kenya is exporting pineapples to an importer in Rotterdam ...

- At least 4 parties, often more
  - Exporter
  - Exporter’s Bank
  - Importer’s Bank
  - Importer
  - There may be 10’s of parties

- Kinds of documents
  - Order
  - Letter of Credit
  - Export documents
  - Draft
  - ...

- Today
  - Some documents communicated electronically
  - Other documents sent by air courier

From “International Financial Management” by Jeff Madura
Example from International Trade Finance

- Suppose that a company in Kenya is exporting pineapples to an importer in Rotterdam...

  1. Importer orders goods
  2. Exporter agrees to fill the order

At least 4 parties, often more:
- Exporter
- Exporter’s Bank
- Importer’s Bank
- Importer

Kinds of documents:
- Order
- Letter of Credit
- Export documents
- Draft

Today:
- Some documents communicated electronically
- Other documents sent by air courier
Example from International Trade Finance

- Suppose that a company in Kenya is exporting pineapples to an importer in Rotterdam ...

At least 4 parties, often more
- Exporter
- Exporter’s Bank
- Importer’s Bank
- Importer

Kinds of documents
- Order
- Letter of Credit
- Export documents
- Draft

Today
- Some documents communicated electronically
- Other documents sent by air courier
Example from International Trade Finance

- Suppose that a company in Kenya is exporting pineapples to an importer in Rotterdam...

At least 4 parties, often more
- Exporter
- Exporter’s Bank
- Importer’s Bank
- Importer

Kinds of documents
- Order
- Letter of Credit
- Export documents
- Draft

Today
- Some documents communicated electronically
- Other documents sent by air courier

---

From “International Financial Management” by Jeff Madura
Before Blockchain

- Private copies of collaboration data
  → Disputes can take month+ to resolve
- Private copies of collaboration processing logic
  → Trust is based on binary relationships

With Blockchain

- Single shared copy of collaboration data
  → Disputes can be resolved in a day
- Single shared copy of collaboration processing logic
  → Trust becomes based on broadly visible shared data
Many application areas

- **Trade Finance**
  - Trust between numerous parties, dispute resolution

- **Supply chain/logistics**
  - Non-disputable order tracking, dispute resolution
  - Important to both advanced and developing countries

- **Mortgage processing**
  - Capture machine readable data once; From redundant paper copies to single source of truth

- **Certified Emissions Reduction (CER)**
  - Enabling manufacturers to certify that they are producing product with low carbon footprint

- **Food supply**
  - Provenance from farm to fork

- **Healthcare**
  - More solid, robust basis for electronic health records

- **Education (especially in developing countries)**
  - Accurate, non-disputable student & teacher records

...
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
- Artifact-centric paradigm as starting point for Business Collaboration Language
- Selected research challenge areas
- Conclusions
A highly selective & brief history of Blockchain

- **Bitcoin**
  - Introduces Blockchain paradigm as basis for a crypto currency
  - Sole focus is on possession/transfer of Bitcoins
  - Privacy guaranteed for currency holders
  - Exchanges to trade Bitcoins for state-provided currencies ($, €, ¥, ...)

- **Etherium - a Swiss nonprofit, launched in 2014**
  - General purpose, custom built Blockchain: ~7000 nodes
  - Crypto currency is called “Ether”
  - Framework includes notion of “fuel” or “gas money” - pay for transactions along the way

- **“The DAO” hack**
  - A Distributed Autonomous Organization (DAO) can be set up on Etherium
    - Participants can contribute funding, and collectively vote on investments
  - “The DAO” launched on April 30, 2016, by German company Slock.it
    - By May 27 the DAO at raised $150M
  - An attacker drained 3.6M ether, worth about $70M, by June 18
  - Value of ether dropped from $20 to $13

- **HyperLedger**
  - Launched by the Linux Foundation - Dec 2015
  - 30 founding members, including: Accenture, Cisco, Digital Asset Holdings, Fujitsu, IBM, Intel, J.P. Morgan, R3, SWIFT, Wells Fargo,...
A blockchain provides
1. High reliability
2. Shared single source of truth
3. Trusted
4. Selective privacy
5. Non-repudiable data updates

A blockchain consists in a network of servers
- They may not trust each other at level of individuals

Blockchain network supports ACID transactions
- Consensus algorithm, such as Practical Byzantine Fault Tolerance (PBFT)

Blockchain network supports selective privacy
- Deep usage of encryption technologies
- Selective access to data and service calls
- (Often, the “smart contracts” are broadly visible)
Blockchain 101 (with bias towards Hyperledger) (2 of 2)

Participants (executing on behalf of businesses)

Network of Peers (“Validating Nodes”)

Shared Ledger View

Two types of txns
- Code Deploying
- Code Invoking

A participant can connect to a single peer, and will always see the single shared version of the ledger.

After each round of consensus, each peer holds a replica of the ledger.

• A “chain” of “blocks”
• The sequence of blocks is the shared “ledger”
Blockchain 101 (with bias towards Hyperledger) (3 of 3)

- What makes Hyperledger different?
  - No built-in crypto currency
  - Cost of processing & data storage is not of major concern
    - Smaller number of peers
  - Anticipation of many Blockchain networks - spectrum including
    - Some more public
    - Some more private
  - All of the nodes are white-listed within a Blockchain network
    - Transactors are granted an identity by an issuing authority
  - Modular consensus
    - Consensus algorithms are pluggable

http://www.the-blockchain.com/docs/Hyperledger%20Whitepaper.pdf
Business Collaboration Language logically above Shared Ledger

- Business-Level Smart Contract Language & Framework

Logical Abstraction Separation

Network of Peers ("Validating Nodes")

- Reminiscent of “Physical Data Independence” in databases

- Proof point: [Weber et. al., BPM 2016]
  - Maps BPMN onto Ethereum blockchain

Shared Ledger View

Trxn 1
Trxn 2
Trxn 3

...
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
  - Artifact-centric paradigm as starting point for Business Collaboration Language
    - Business Artifacts and related models
    - ACSI - Artifact-Centric Service Interoperation
- Selected research challenge areas
- Conclusions
Business Artifacts with Lifecycles: A way to factor Business Processes and their data that gives unifying, end-to-end view

A logical view that is natural to biz-level stakeholders

Order (between Importer & Exporter)

Letter of Credit

Shipment of Physical Goods

Export Documents

Draft (request for payment)

Manages/tracks overall operation of the Order, from creation to delivery

Establishes trust between Importer (Bank) & Exporter (Bank)

Tracking physical shipment

Legal documents holding information about the shipment

Financial contract between Exporter Bank and Importer Bank (may be transferred)
Each Artifact type includes info model, lifecycle model, and roles

- Info model brings together all biz-relevant data about a given artifact type
  - These cut across parties, organizational silos, etc.
  - Provide a common vocabulary across parties, silos
- Lifecycle model shows possible progressions of artifact instance through the business operations
  - Status of Lifecycle is stored in the info model
- Roles have access rights to data & operations
- Biz-level stakeholders can easily query, monitor, use dashboards, and specify rules/policies
There is extensive research on “Data-aware” Business Process

- The use of entities with FSM-based lifecycles appears as early as [K. Robinson 1979] and [C. Rosenquist 1982]
  - Focus is on Information Systems and System Dynamics rather than modeling modern business operations.
- Business Artifacts (a.k.a., Business Entities with Lifecycles) [Nigam+Caswell 2003, Kumaran et al 2003]
  - Use cases - Pharmaceutical, supply chain, manufacturing, finance, je.g., [Bhattacharya et al 2007], ...
  - Systems - e.g., BizArtifact open source system [Boaz, Limonad, Gupta 2013]
  - Standards - impact of GSM approach on CMMN standard [Marin, _, Vaculin 2012]
  - Foundations - e.g., [Bhattacharya et al 2007] [Deutsch et al 2009] [ _ et al 2010] [Calvenese et al 2013] ...
  - Collaboration - Artifact-Centric Service Interoperation (ACSI) [ _, Narendra, Nigam ICSOC 2009]
- Business Objects
  - Very similar to Business Artifacts with FSM lifecycles
  - PHIharmonicflows [Kunzle, Reichert 2011] ...
- Active XML Artifacts model
  - Information Model based on (Active) XML [Abiteboul, Segufin, Vianu 2009]
  - Collaboration [Abiteboul et al 2010]
- Case Management
  - Pallas Athena FLOWer [van der Aalst, Weske, Grünbauer 2007]
  - Adaptive Case Management [Swenson 2010]
  - OMG Case Management and Modeling Notation (CMMN) [2014]
Artifact-Centric Service Interoperation (ACSI) [ _, Narendra, Nigam, ICSOC 2009]

- Our Inspiration: EasyChair
- A “hub” that supports numerous conferences

How to replicate this in arbitrary application areas?
How do we make it easy for services to “understand” what a hub is doing, how it is working?
How do we provide systematic access controls?
Artifact-centric Service Interoperation hubs in a nutshell

- Like orchestration, puts a “hub” in the middle of collaborating services
- Unlike orchestration, an ACSI hub:
  - Establishes a “pseudo-standard” as the center of a collaboration environment
    - Different service collaborations will need to comply to this standard
    - Enables scalability because of distribution of adaptation
  - Primarily re-active; essentially a structured whiteboard
    - Assumes that the services are “self-motivated” and pro-active
    - Permits service-to-service communication
  - Based on Business Artifacts
    - Demonstrated to simplify understanding of business operations
    - A structured approach to data + process + roles
    - Data: Info Model provides skeleton that provides full status snapshot
    - Process: Lifecycle Model provides top-down structure for processes, events
    - Roles: Info Model + Lifecycle Model provides structure for intricate, conditional access control
Example ACSI Hub for Trade Finance

Artifact Types:
- Order
- Letter of Credit
- Shipment Tracker
- Exp Docs
- Draft
- ...

- The participating services do not have to be artifact-centric
Illustration of Lifecycle & Info Models

Order

- Request from Imp
  - Terms & Cond’s from Exp
    - Accepted by Imp
    - Under Revision
      - Abandoned

Lifecycle Models

Info Model

Not shown:
- Roles
- Access rights

Letter of Credit

- Requested by Imp
  - Submitted by ImpB
    - Under Revision
      - Rejected by ExpB
        - Abandoned

- Accepted by ExpB

- Abandoned

- Importer, Exporter, Importer Bank, Exporter Bank, Total Price, Export Tariff, Line items, Fee on Importer
Separated lifecycle models are natural backbone for access to services

Order

Letter of Credit

Imp
ImpB
Exp
ExpB

Importer
Importer Bank
Exporter
Exporter Bank

Request from Imp
Terms & Cond's from Exp
Accepted by Imp
Abandoned

Accepted by ExpB
Abandoned

Requested by Imp
Submitted by ImpB
Rejected by ExpB
Under Revision

Copyright © IBM 2016
ACSI Views: Limiting what can be seen

**View for Exporter:**
Hiding negotiation between Importer Bank & Exporter Bank

**State Condensation:**
Exporter sees one state that is a “collapse” of 3 states

**Attribute Projection:**
Certain attributes are hidden from Exporter’s view
ACSI Windows: Limiting which artifact instances can be seen

Window query: \[ \{ (p, a) \mid \varphi(p, a) \} \]

Examples:
1. Importer sees all of it’s Orders
2. Importer Bank sees all Letters-of-Credit, Export Docs, and Drafts for which it is providing or receiving credit
3. Harbor Master sees all Shipments coming in or out of harbor
ACSI service controls: Limiting what participants can do

- E.g., for Importer Bank, we can specify in detail which attributes can be created, read, updated, ...
- Permissions change based on state you are in

Create
Read
Update
Delete
Append
Execute

- Requested by Imp
- Submitted by ImpB
- Rejected by ExpB
- Under Revision
- Accepted by ExpB
- Abandoned

Importer, Exporter, Importer Bank, Exporter Bank, Total Price, Export Tariff, Line Items, Fee on Importer
Artifacts and ACSI: Providing a robust starting point for a Business-Level Collaboration Framework for Blockchain

- There is also substantial research on the ACSI paradigm
  - Cf. EU-supported ACSI project (2010 to 2013)
- Systems - Biz Artifact (open source)
- Foundations

But ... We cannot apply them “out of the box”

- Conceptual models: Blockchain restrictions - e.g., synchronous service calls
  - Operational perspective
  - Contractual perspective, including legal and natural language
- Systems: Mapping onto Hyperledger, Ethereum, etc.
- Collaboration/Choreography: Very relevant
- Verification: Different questions
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
- Artifact-centric paradigm as starting point for Business Collaboration Language

Selected research challenge areas
  - Language design
  - Reasoning about artifacts
  - Relationship to natural language contracts

Conclusions
Requirements on Business-level Smart Contracts Framework

Solution Language

- Intuitive for Business-level users to create and understand smart contracts
  - Example users: Business Analysts, Trade Specialists, Financial Analysts, Supply Chain Specialists, ...
  - Holistic way of representing key business objects, including data, lifecycles, rules, roles
- Linkage between **Legal Contractual perspective** and Operational perspective
- Linkage to **existing standards, e.g., UBL, SWIFT, ...**
- Intuitive support for adding variations into existing smart contract specifications
  - Including modifications to business object data, lifecycles, rules
- **Modularity & Composability**
  - Intuitively natural ways to do “plug and play”, and to substitute portions of a smart contract
  - Note: in the future, smart contracts will be created by different organizations and mashed together
- Access Control & Privacy features - specified at business level
  - For data
  - For invocable operations

Solution Development & Administration

- Visual editor
- Enable **rapid development & modification of production-level solutions**
  - Use a fully interpreted paradigm for execution of smart contracts
- Design, develop, deploy, test, refine
- Version management

- Ricardian contracts appear relevant
- Emerging CLACK language [Clack et al 2016] aimed at this challenge
- Artifact types can serve as natural composable modules
- Data & lifecycles provide further modularity

The BizArtifact system [Boaz et al 2013] for artifacts included
- Visual editor
- Fully interpreted implementation of artifacts
- Administration framework
Working hypothesis for a Business Collaboration Language (BCL)

- Leverage Business Artifact / Business Object paradigm
- Information model
  - Nested relations (e.g., represented as JSON)
  - Query/conditions by subset of SQL++ [Ong, Papakanstatinou, Vernoux 2015]
- Lifecycle model - multiple possibilities
  - Acyclic DAGs with rollback (cf. [van der Aalst, Weske, Grünbauer 2007])
  - Finite State Machines
  - Guard-Stage-Milestone/CMMN (see [Marin,H.,Vaculin 2011])
  - Active XML
  - Proclets [van der Aalst et al 200?] / BPMN
  - More declarative, e.g., DecSerFlow [van der Aalst et al 2009], DCR Graphs [Hildebrandt et al 2011]
  - Other ???
- Access constraints
  - Data: should be conditioned on Lifecycle status and data values
  - Lifecycle:
    - Restricted access has not been emphasized in Blockchain literature
    - Techniques from ACSI might be adapted

FSM’s as reasonable starting point because
- Adoption: “Everyone” is familiar
- Natural mapping from Functional Programming formalization of legal contracts (see below)

Use extended Event-Condition-Action (ECA) rules
- Captures the reactive nature of Blockchain smart contract
- While permitting declarative style
Representative standard - UBL 2.1

- OASIS standard focused on Supply Chain use cases
  - Approved by OASIS in 2013 and as ISO Standard in 2015
  - Adopted by/extended by: EU Public Sector; various procurement frameworks (Norway, Sweeden, EU DIGIT, UK Natl. Health Service, OpenPEPPOL (several Euro countries), Port of Hong Kong, Port of Singapore, ...)
  - Supports approx. 45 business process flows and 65 document types
  - For document types: validators, authoring software, parsers, generators

- Focus is mainly on binary relationships/processes, and includes
  - Workflows for order, invoice, shipment, consignment, ...
  - For each workflow, document types for exchanging information
    - The workflows provide the business context within which documents are exchanged

- E.g., “Ordering is the collaboration that creates a contractual obligation between the
  - Seller Supplier Party and the
  - Buyer Customer Party.”

Blockchain allows the documents to be shared . . .

. . . enables the documents to support change through time
Example based on Order with multiple Line Items, each shipped separately

- **Solid arrows:**
  - Transitions internal to an artifact
  - If the triggering event is from “outside” then event is not shown

- **Dashed arrows**
  - Triggering events that go between artifacts

Order:
- Placed → Approved → Completed
- Placed → Approved → Rejected
- Approved → Back-Ordered
- Approved → Shipped
- Shipped → Received
- Shipped → Canceled
- Placed → Created
- Created → Pending
- Pending → Shipped
- Shipped → Received
- Shipped → Canceled

We will focus on one Line Item being Received

If last Line Item has not yet been processed

If last Line Item has been processed

Copyright © IBM 2016
**Targeted example: Info Models and Lifecycle Models**

### Order
- **PurchaseOrderId**: string
- **LineItemList**: [ { LineItemId: string, Price: int, Weight: float } ]
- **LineItemsReceivedSoFar**: [ { LineItemId: string } ]
- **Status**: string

### Line Item
- **LineItemId**: string
- **ParentPurchaseOrderId**: string
- **Price**: int
- **Weight**: float
- **Status**: string

Note: each artifact has an implicit attribute “WaitingFor : { name: string, id: string }“, which is used to record invokes that have gone out but not returned yet. A transaction is not completed unless all the attributes WaitingFor have null value.

This unary relation is used to hold list of all Line Items that have been received by the Buyer so far.

"Status" is used to hold current FSM state.

This FSM should move to “Completed” when the last Line Item has been received.

reportLineItemReceived (invoked from Line Item artifacts)

linItemReceived (invoked from outside)
Illustration: Three ECA rules that string together

This FSM should move to “Completed” when the last Line Item has been received.

Who: Buyer
On: receive lineItemReceived (LinItemld: string, date: dateTime)
Let:
If: self.state == “Shipped”
Then:
  self.state := “Shipped” ;
  invoke reportLineItemReceived (target: self.ParentPurchaseOrderId, ReceivedLineItemId: input.LinItemld, ReceivedDate: input.date)

Who: Buyer
On: receive reportLineItemRecieved (target: string, ReceivedLineItemId: string, ReceivedDate: dateTime)
Let:
allLineItems := self.LineItemList.{LinItemld}
allReceived := union (self.ReceivedLineItems,
  { { LineItemId: input.ReceivedLineItemId } })
If: (self.state == “Approved”) and (allLineItems == allReceived)
Then:
  self.ReceivedLineItems := allReceived;
  self.state = “Completed”;
  invoke Callback (target: input.ReceivedLineItemId)

Who: Buyer
On: receive callBack reportLineItemRecieved (target: string)
Let:
If: self.state = “Shipped”
Then:
  self.state = “Received”;
  invokeCallback (LineItemsStatus: “Finished”)

Test of whether all Shipments have been processed.

Switch from Line Items to Shipments.
A design method for Artifact-centric Smart Contracts
(adapted from [Nandi et al 2008])

- **Identify Key Business Entities**
  - Identify, name, and describe the key shared evolving Entities in the scope of the project
  - These will sometimes correspond to document types in existing standards
  - Provide clear, unambiguous description of the business intent of each Entity

- **Identify Milestones & Transitions**
  - Identify the key milestones of each Entity in the scope of the engagement
  - Sketch the data needed to enter into each milestone
    - For each Entity, sketch a picture of valid transitions between its milestones
    - For each edge, determine which Roles can invoke that transition

- **Detail Milestone Data Fields**
  - Finalize the Data Groups for each Entity milestone
  - Expand each Data Group into fields
  - Identify any rules and constraints to ensure quality and consistency of the data

- **Define Transition Rules**
  - For each Entity, identify the rules for each transition between the key milestones
  - Also identify the rules that move between Entity types
    - For each rule, include the roles that can invoke it, conditions under which it should be fired, and what data updates it performs

- **Validate / Verify**
  - Perform testing based on variety of examples and corner cases
  - Examine how legacy services will need to be modified in order to use the Blockchain solution.
    (If there will be large expense, consider modifying the solution.)
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
- Artifact-centric paradigm as starting point for Business Collaboration Language

Selected research challenge areas
- Language design
  - Reasoning about artifacts
  - Relationship to natural language contracts

Conclusions
A single collaboration will involve numerous artifact instances, with multiple 1-to-many relationships.

Different artifact types designed & maintained by different organizations.

Why?
- Making artifact types similar to existing standards, e.g., UBL, SWIFT, ...
- Different kinds of concerns for logistics vs. finance

Impact:
- Interaction between artifact types must be understood.
Reasoning about “Choreographies” of Artifacts
(adapting from [Sun, Xu, Su ICSOC 2012], [Su, Sun 2013])

- Suppose that each artifact type is a separate Smart Contract
- In a HyperLedger fabric, communication will be by synchronous service calls

How to develop a framework for specifying desired properties?
- Design “correctness”, auto realization, mechanisms for monitoring, ...
- Key Issue: Need explicit way to model & reason about correlations

- Order
- Letter of Credit
- Shipment
- Draft

Single Draft might pay for multiple Shipments
Two Key Aspects of Choreography Languages

Order

Letter of Credit

Shipments

Drafts

Conditions involving Data:
For each shipment, if Export Docs obtained by Exporter, then a Draft including that Shipment is generated by Importer Bank

If the amount is less $1000 and additional Shipments pending, then delay Draft

Instance-level correlation:
Which artifact instances are correlated during the runtime?
Who sends messages to whom?
Tangent: How might BPMN “Collaboration” model this?

- This is collaboration diagram for a single Shipment mapping to Draft

- In general
  - Grouping of Shipments into Drafts will be random
  - Timing of different Shipments completing, and different Drafts completing will be random

- So, well-formed multi-instance construct cannot be used
Framework for Interacting Artifact Types: Correlation Diagrams

- Two artifact instances are **correlated** if they are involved in a common collaborative BP instance.
  - *Messaging only between correlated instances*
- Correlations for a collaboration are defined in a diagram, with one artifact as the **root** or **primary** process.

- Directed edge indicates creation of Artifact instance(s)
- Cardinality constraints are also defined
- Some syntactic restrictions (acyclic, “1” on root, ...)
- Correlations can also be **derived**
A message diagram defines message types and sender/receiver of each type
- Includes messages from/to the external parties
- “+” means creation of new BP instance

Message may have data attributes
- Path expressions are used to access data contents
Choreography Constraint Example

For each Shipment \( s \),
\[
G \quad \left( \exists \text{Expdocs } D \quad (D\text{.shipment.ID} = s\text{.ID} \text{ and } D\text{.status} = \text{"Received by Exp"}) \right) \quad \rightarrow \quad F \quad \exists \text{message } \mu(\text{Draft, ImpB}) \quad \text{(Type(\mu) = "New Draft Notif." and } s \text{ in } \mu\text{.payload.shipments)}
\]

“For each shipment, if ExpDocs received by Exporter, then eventually a Draft Notification including that shipment is sent to Import Bank”

Expressed in LTL-FO (with some syntactic sugar)
Reasoning with Choreography Constraints on Artifacts

- This field is still in its infancy
- [Su, Sun 2013] study “Realizability”

Given an artifact message diagram and family C of Choreography Constraints …

… is there a family of services that “realizes” exactly the runs that satisfy C?

- Preliminary result: “Yes”, if there are no 1-to-many relationships between artifacts
- Question remains open in the general setting

- What about Verification?

Given full artifact system S and family C of Constraints …

… does every run of S satisfy C?
Verification for artifact-centric models: a representative framework

- Given an artifact-based model \( M \) and a property \( P \), do all executions of \( M \) satisfy \( P \)?

  + Artifact Info models
  + Semantic Tasks (specified using pre- and post-conditions, in spirit of OWL-S)
  + Lifecycle (expressed using rules)

 inconsistency

- The presence of data leads to an infinite state space
  - Verification in general is undecidable
  - Several different approaches to restrict expressive power have been developed

“Books should not ship until after payment”

Temporal + First-Order, e.g., LTL-FO

goals / constraints
Several approaches to verification for data-centric process (see surveys [Calvanese, De Giacomo, Montali 2013], Deutsch, H., Patrizi, Vianu 2014])

- Early results: Focus on “flat” artifacts
  - [Bhattacharya, Gerede, _, Liu, Su 2007] multi-artifact system; PSPACE-complete verification
  - [Zhao, Su, Yan, Qiu 2009] verification of propositional LTL

- Flat artifacts with fixed re-only database
  - [Deutsch, H., Marcus, Patrizi, Sui, Vianu, Zhou, starting in 2009]
  - Strong restrictions, e.g., no dependencies, no arithmetic, focus on $\exists$FO conditions; LTL+$\exists$FO constraints
  - Obtain PSPACE verification by reduction to finite-state model checking

- Recent extension
  - Hierarchical lifecycle models and artifacts with relation-valued attributes
  - [Deutsch, Li, Vianu 2016] Obtain PSPACE verification with reduction to Vector Addition System with States

- Data-Centric Dynamic Systems (DCDS)
  - [Bagheri Hariri, Calvanese, De Giacomo, De Masellis, Felli 2011] and others ...
  - Artifacts range over relations, powerful temporal logics
  - Acyclicity conditions guarantee bounded domain
  - [Belardinelli, Lomuscio, Patrizi 2013] Verification of DCDS in context of multi-agent system

Blockchain context brings a different focus - limited data, access rights, choreography constraints, 1-to-many relationships, ...
Reasoning about views [Abiteboul, Vianu 2015]

- Each participant working on a view
  - Their actions are propagated to the others, according to semantics of the (virtual) global state

- Studying what a participant can infer about the global state, e.g.,
  - As an author, has my paper been accepted?
  - As a PC member, can I tell if a decision was made about my own submission?

- Model
  - Global database is set of relations with keys
  - Each view sees projections, with entire key present
  - Using PLTL-FO - “P” for “previous”

- For fixed workflow schema, reasoning in PSPACE
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
- Artifact-centric paradigm as starting point for Business Collaboration Language

Selected research challenge areas
- Language design
- Reasoning about artifacts
- Relationship to natural language contracts

Conclusions
THESE ARE REALLY SMART PEOPLE, said one lawyer who works with startups. They believe in WORLD-DOMINATION of the engineering class; EVERYTHING can be reduced to an ALGORITHM and LEGAL DOCUMENTS are not going to be SPARED.

From [www.legalese.com](http://www.legalese.com) home page
Legal contracts: what makes them different?

- Binary relationship
  - “Holder”
  - “Counterparty”
- Contract based along time dimension
  - As they move through time...
  - ... people make choices
  - ... result is essentially a new contract
- Contracts are exchanged, combined, traded, ...
- Contract may depend on external “random” variables
  - E.g., exchange rates, stock prices
- A focus of financial industry is
  
  What is the current value of this contract?
  
  - Must incorporate uncertainties of future
  - Various statistical techniques available

On 15 July 2000 you may choose between:

\[ D_1 \text{ Both of:} \]
\[ D_{11} \text{ Receive } £100 \text{ on 29 Jan 2001.} \]
\[ D_{12} \text{ Pay } £105 \text{ on 1 Feb 2002.} \]

\[ D_2 \text{ An option exercisable on 15 Dec 2000 to choose one of:} \]
\[ D_{21} \text{ Both of:} \]
\[ D_{211} \text{ Receive } £100 \text{ on 29 Jan 2001.} \]
\[ D_{212} \text{ Pay } £106 \text{ on 1 Feb 2002.} \]

\[ D_{22} \text{ Both of:} \]
\[ D_{221} \text{ Receive } £100 \text{ on 29 Jan 2001.} \]
\[ D_{222} \text{ Pay } £112 \text{ on 1 Feb 2003.} \]

Functional programming can provide formal abstraction for finance-based contracts

[Peyton Jones, Eber 2000] provides a family of 10 primitive combinators that can be used to formally define contracts

- “and”: if you acquire “c1 and c2”, then you immediately have both
- “or”: if you acquire “c1 or c2”, then you must immediately choose to retain one or the other
- “when”: if you acquire “when <obs> c”, where <obs> is a Boolean-valued observable, then c becomes available to you if/when <obs> becomes true
- “until”: “until <obs> c” acts like c until <obs> becomes true. From that moment the contract becomes worthless
- ...

This functional programming view enables

- Composability
- Formal reasoning about semantic equivalence

Conjecture: a family of inter-related binary contracts can operationalized using an artifact-based Blockchain implementation

Diagram:

1. Functional Programming Specification
2. Multi-Party Artifact Specification
3. Blockchain

- Reasoning about value
- Operational semantics, Reasoning about operations
- Reliable execution
Another perspective on mixing “legal” and “smart” contracts

- Distinction made in CoinDesk by Stark [2016]
  - Smart Contract Code: code that embodies how agents want to collaborate, running on a Blockchain
  - Smart Legal Contract: combination of legal wording and executable code that correspond to each other

- Ricardian contracts: an example of Smart Legal Contract
  - Invented by Ian Grigg [2004]
  - “A digital contract that defines the terms and conditions of an interaction between two or more peers, that is cryptographically signed and verified”
  - It is both human and machine readable
  - Has a unique and secure identifier

Groups like CommonAccord are attempting to create a body of “universal contracts” that can handle essentially all useful kinds of collaborations
Agenda

- Blockchain enables a new level of trust & communication
- What is Blockchain, and why is it useful for Business Collaborations?
- Logical separation between Blockchain mechanics and Biz-level programming
- Artifact-centric paradigm as starting point for Business Collaboration Language
- Selected research challenge areas

- Conclusions
Blockchain: A new technology with growing adoption for Business Collaboration

This raises many of the classical questions from Services Research community . . .

. . . But with a twist:

A persistent shared trusted data store is at the very center of things

- Allows, and forces, a re-thinking of basic services paradigms, such as
  - Orchestration/choreography: is ACSI hub the right abstraction, or something else?
  - Service composition: It’s not just about message/conversation compatibility anymore
    - Will Business Artifacts be the unit of composition, or something else?
  - Service design patterns: How to use presence of data to best advantage?
Blockchain: Operational vis-a-vis Legal/Financial perspectives

Two critical observations:

- The courts will always be the remedy of last resort → legal perspective is always present
- Almost every operational task has a financial aspect → financial perspective is always present

- Brings a new style of challenge to the services community
  - Service composition: Legal and Financial contracts are interlocking, interdependent
    - Do our current paradigms adequately model this?
  - Formal reasoning/verification: We need to address Legal/Financial patterns (among others)
  - Design/Coding style: How will marriage of legal+code be structured, at macro- and micro- levels