Supply Chains and other Complex and Entangled Ecosystems

a sketch toward learning and human agency in a whitewater world

Tom Winans & John Seely Brown

Intended for The American Academy of Arts and Sciences Exploratory Meeting:

Building Resilient and Ethical Supply Chains for a Post-COVID World

Obsolescence of 20th C Business and Technology Architectures

The affordances of technology innovations of especially the last 15 years permit scale in time, geography, cost and efficiencies, and they can be leveraged to manage business as it has been done in the past, but also to do business differently… not simply "differently" as *a slight variance in operations to yield an incremental improvement in unit economics*, but *differently*… as in being a *game changer*

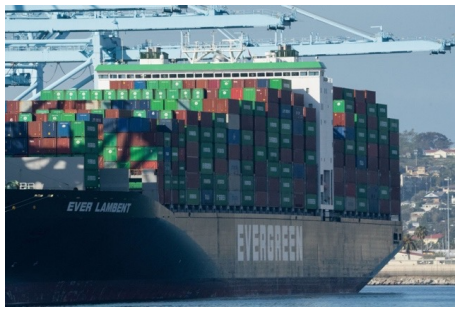
We've done our best to add to our architectural edge rather than fully replace. Whether out of fear, lack of imagination, or willingness to *kick our liabilities as cans down the road*, we take baby steps and hide behind conventional wisdom of risk aversion, never truly leveraging these innovations to understand how to do business *differently* in the 21st C

Our timidity to pursue the future has resulted in inflexible and brittle architectures, fit for their original purpose, but incapable of adapting and extending as one would hope to the next. Yet hope springs eternal… we hope our efforts to evolve last generation capabilities will eventually and magically transform limitations and liabilities to future-proofed assets and differentiators

Even when the limitations are highlighted on a global scale, we seem to be content to deal with first order challenges rather than more foundational/second order changes…

Global Supply-Chain Woes Escalate, Threatening Economic Recovery

Supply-Chain Contracts Get Revamped After Covid-19 Disruptions



Builders Hunt for Alternatives to Materials in Short Supply

Walmart, Target, Home Depot and other large retailers are chartering ships to bypass supply chain problems. Will the strategy save Christmas?

Global supply-chain bottlenecks are feeding on one another, with shortages of components and surging prices of critical raw materials squeezing manufacturers around the world.

The supply shocks are already showing signs of choking off the recovery in some regions.

First Order Problems

WSJ Logistics

Solutions to First order Problems *Usually* are *Temporary and Just Local Fixes*

We create technology and process debt that complicates how we resolve future problems. As we work to get out in front of change, we create patchworks of additional debt because the need for change outpaces our ability to effect it given the limits of our current architectures

We bank on processes we follow when conducting business to make sense for *the foreseeable future*

but then, suddenly, they don't…

<<Pandemic happens>>

A "perfect storm" occurs when many assumptions in our patchwork don't make sense, disentangling and unwinding is an intractable problem, and shock waves are experienced throughout our system

We are here (in 20th C)		but need to be here (in 21st C)	
Pockets of information … local to a single company and in the systems used to run one business No Ecosystem Thinking No architecture environment conducive to any type of sharing and learning	Architectures and communication protocols embed 20th C mindsets and methods Snapshots are rearview mirror in quarters, months, weeks Centralized organizational control model	Information flows become more continuous (*from stocks to flows*) and we must plan and replan, route and re-route All systems need to take a *flows* perspective, not a *stocks one*	*We and our systems* need to be *eco-systemically agile, contextually aware* Organization and system architectures need to be loosely coupled, flexible Goal focused, ability to realize goals in multiple ways Improvisationally

Second Order Problems

Shock waves … *of our own making*

Architectures and communication protocols embed 20th C assumptions, mindsets and methods

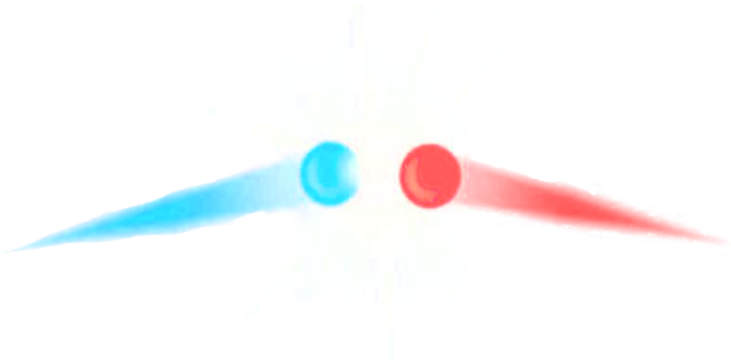
Stock snapshots for quarterly, monthly, weekly, daily performance

Centralized organizational control model to force visibility outward and consistent communication inward

We are here (in 20th C)

To be competitive, we *individually* move faster

But *what if a forcing function requires us to collaboratively move faster*?

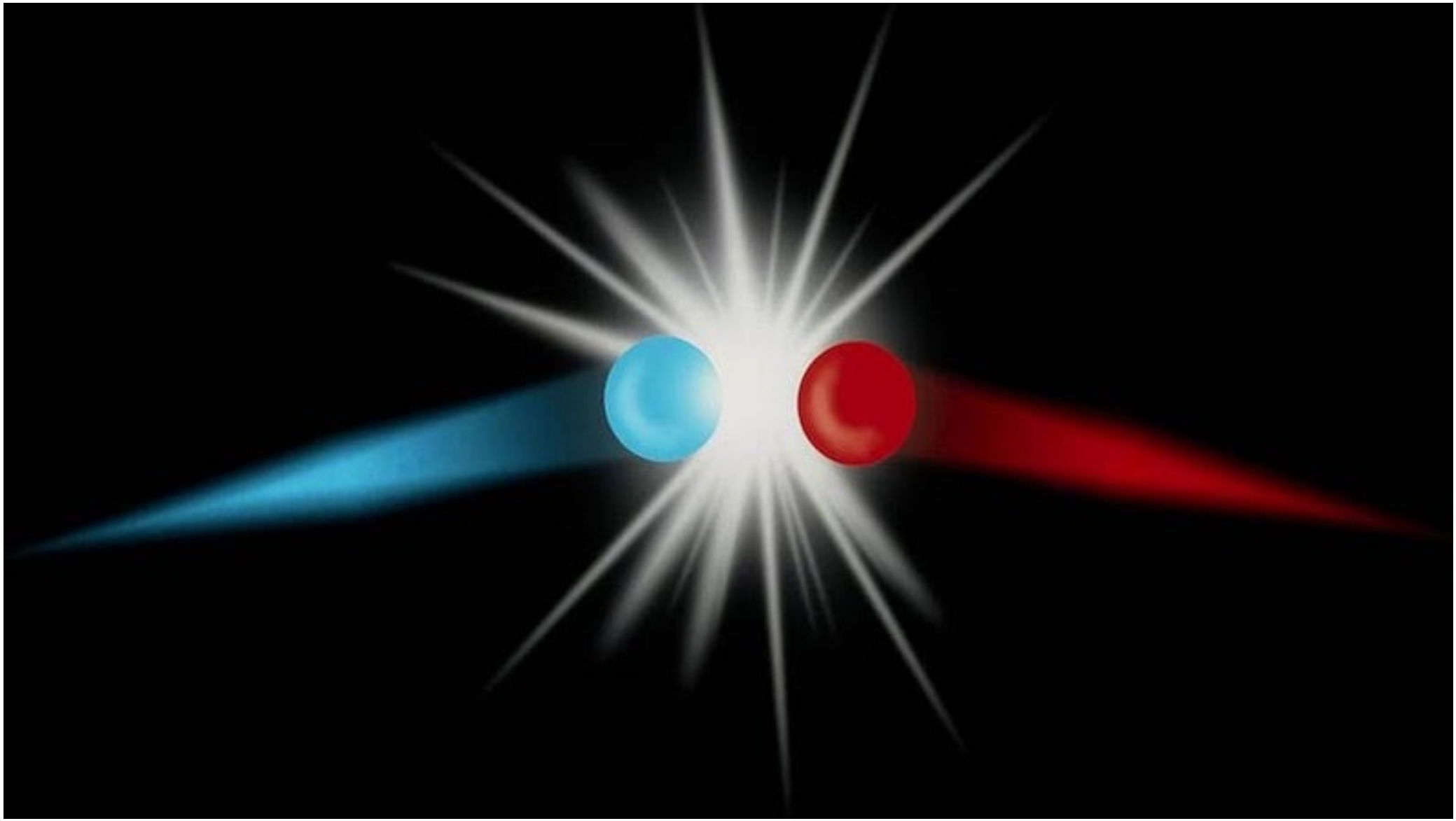


New technical capabilities in networking and communication mean new application and network protocols can be used to status in minute, second, even millisecond-based signaling

We now can look at trends *of flows*

Signals can be viewed globally, correlated for consistent communication to all stakeholders

but need to be here (in 21st C)



We caused these shock waves of our own making. With siloed systems, we have pockets of information scoped to a company … not visible outside of the system that holds it.

If the way we work changes, with 20th C systems we all need to be on the same system to have equal visibility. If "we" belong to different companies, then making integrations between different systems becomes hard to manage as the number of parties grows large. Scaling this becomes even more challenging when we need to accommodate different ways to work…

Communication protocols used to conduct business assume things about corporate organization, processes, etc. These are fixed in software, and they're specific to the time they were conceived. How do they change over time unless the platform allows protocol definitions to be given as data rather than code?

From stocks to flows … We've been looking in our rearview mirror when we look at the past quarter of performance to forecast the next quarter. Same for month and week and day. Our communication protocols give us detail now to milliseconds. When dealing in quarters and months, re-routing or re-planning didn't make much sense. But seeing things sooner gives us time to re-plan and re-route. The knowledge we have in the moment may not be perfect, but it indicates trend, and trend can be used to signal ways to optimize … plans, routes, etc.

Signals viewed globally can mean the different of docking at Port A in Los Angeles, or docking at Port B in Seattle… or it could mean redirecting specific products to another part of the world …

Many of our 21st C methods weren't known in the 20th C…

The systems from the 20th C that we continue to use in the 21st C impose limits on us, then … while we can modify 20th C systems at their edges, we cannot easily bring the edge to the core.

RPAs are not the answer, nor are Autonomous companies

We cannot solve ecosystem problems with business and technology architectures that ignore ecosystems, even if they move with fewer “humans in the loop”

RPAs and smart contracts automate a scripted process – they don’t have a means to be contextually aware or adaptive

Architectures that do not incorporate humans into them ignore the fact that human agency is fundamental to reading and disambiguating context, and disentangling operational systems that are complex, even wicked, on their own

Yet we compound problems by adding to them the friction of insisting on *one way to do things* that is embedded in the systems we use to run our businesses – we require our partners to capitulate or suffer … the days of siloed and standalone business architectures have past

Guard rails previously enforced by a *now obsolete* centralized control model and supporting technologies hardwiring them into place limit the very human agency needed to navigate entanglements and complexities that naturally occur in our world – yet we, with our brownfield risk aversion, insist on dragging these liabilities into the future with the illogical optimism that we can evolve past them, and often you can’t

Our new business and technology architectures need to enable constant improvisation, often collectively done, rather than simplistic and scripted replication (smart contracts, RPAs)

(see glossary)

Unlocking Potential

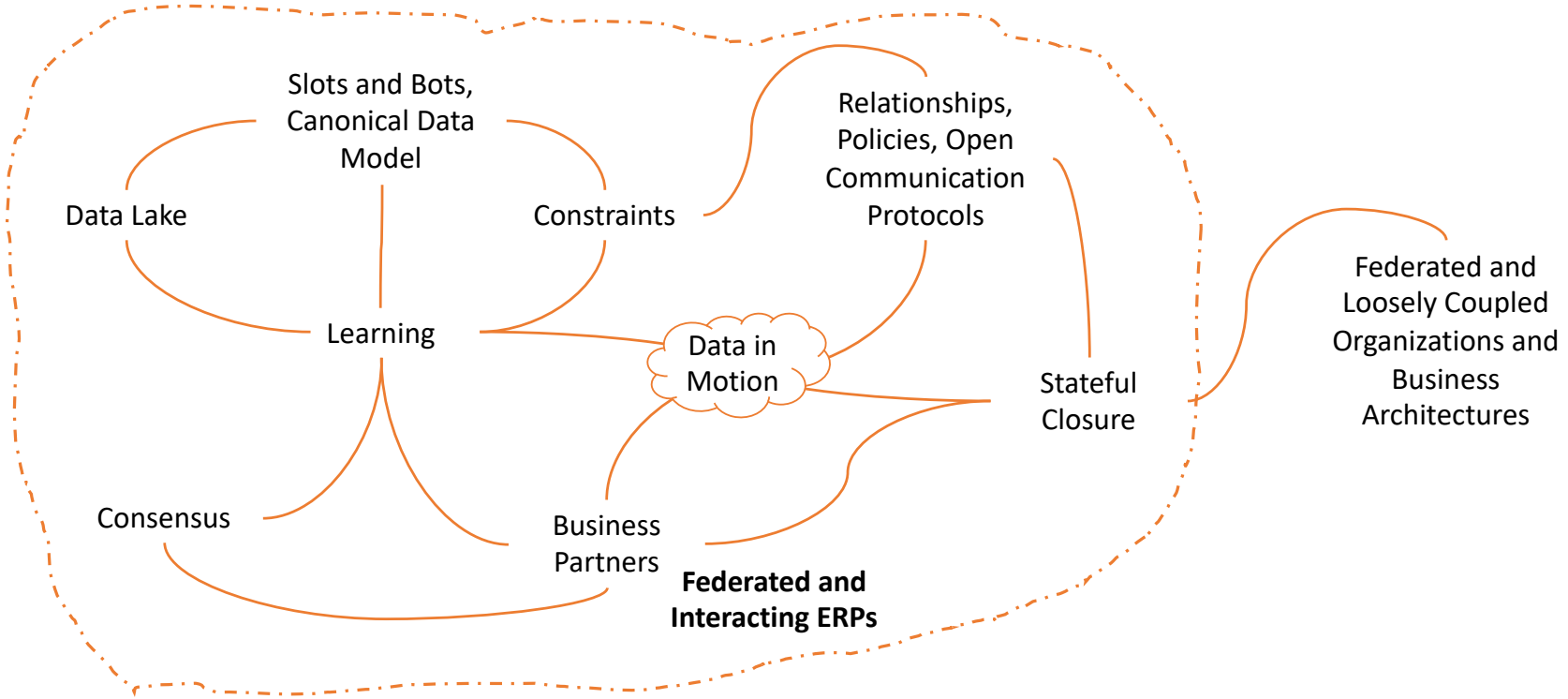
Architectures supporting improvisation at scale, via human agency supported by new types of architecture components, **are prerequisite** to building robust and resilient businesses that can scale *both big and small*, and pivot as needed to survive shock waves

We've arguably already moved into a sufficiently chaotic world that requires ability to re-frame how we work *on demand*. **We haven't the time to codify *how we do what we do* entirely in software** because this takes too long to change… we cannot require our business partners to work the way we do when they work with many partners, not just us…

Our **flows must be *more data* and *less code***, **informed by *data in motion*** and ***shaped by policies as constraints*** *that serve as* ***context-sensitive guard rails***, such that we can *systematically* adjust to change, which is our *new normal*

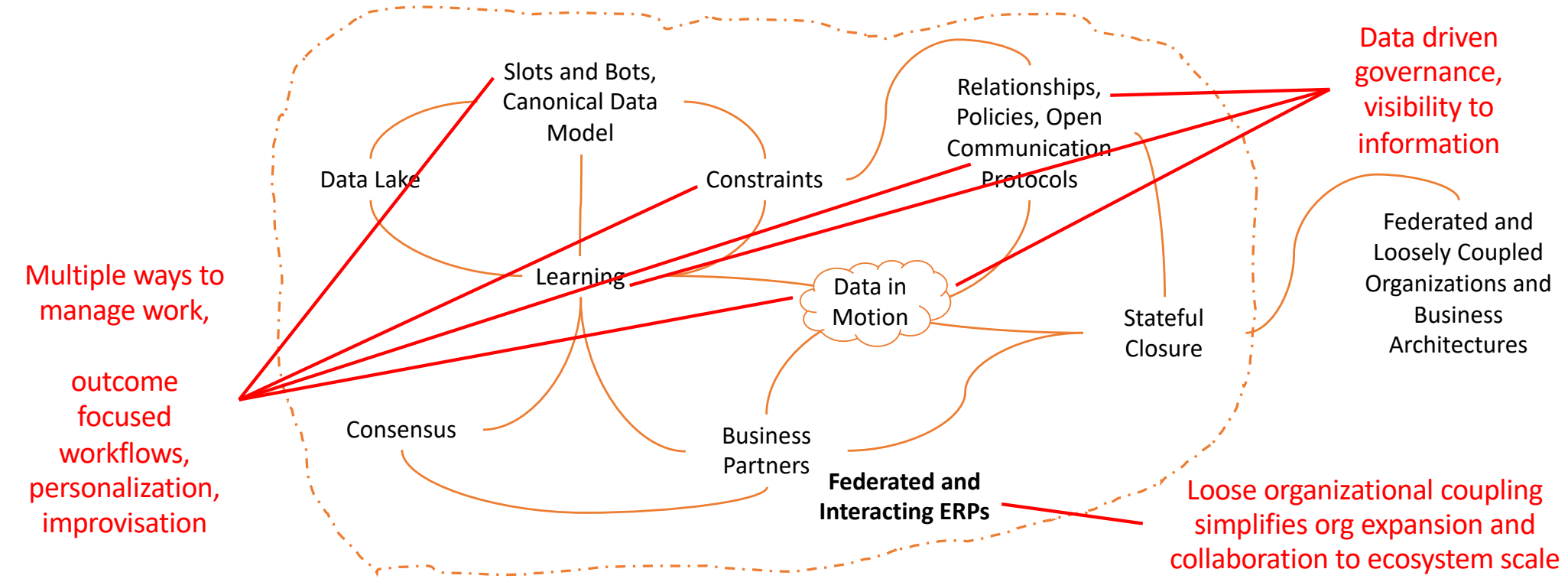
And **we must implement our architectures to bring what we learn from our edge into our core**, where decisions are made, where adapting to change and context *really* takes place, both in our business and our technology

Anything in our architectures **that honors improvisation sets the stage for learning**

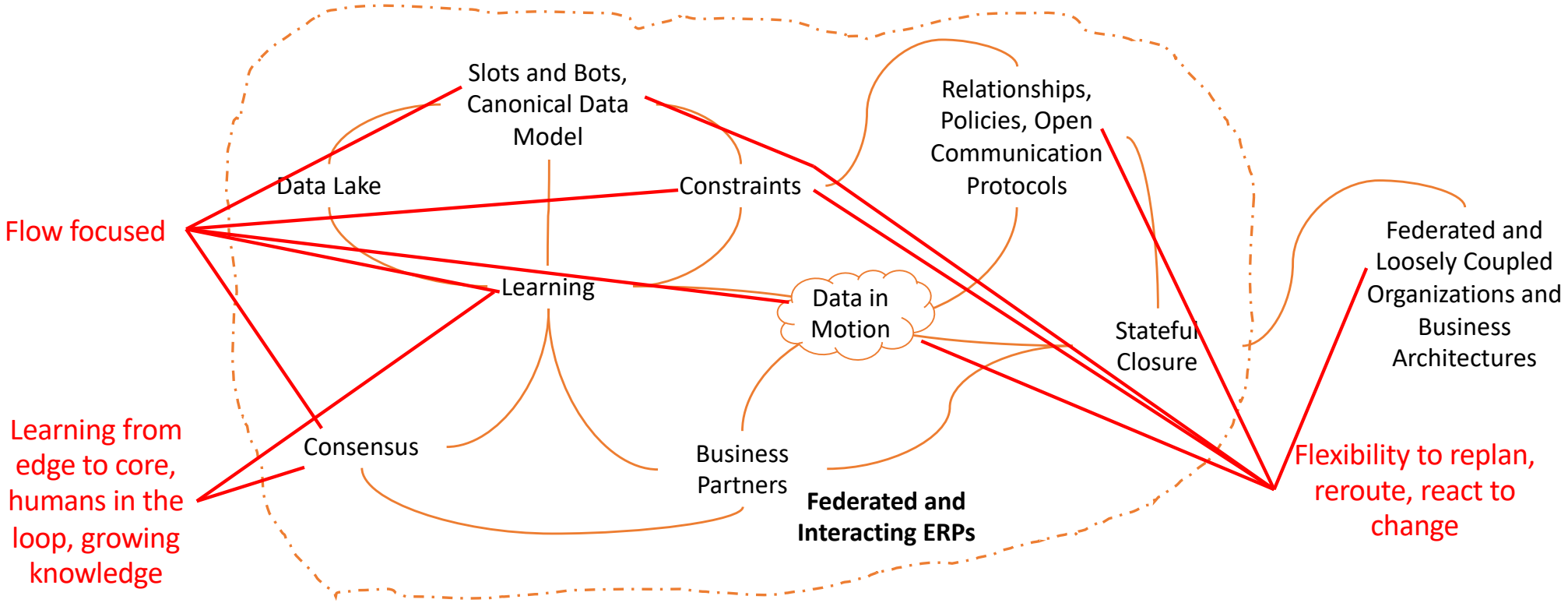


Second Order Solutions | A Sketch

Second Order Solutions

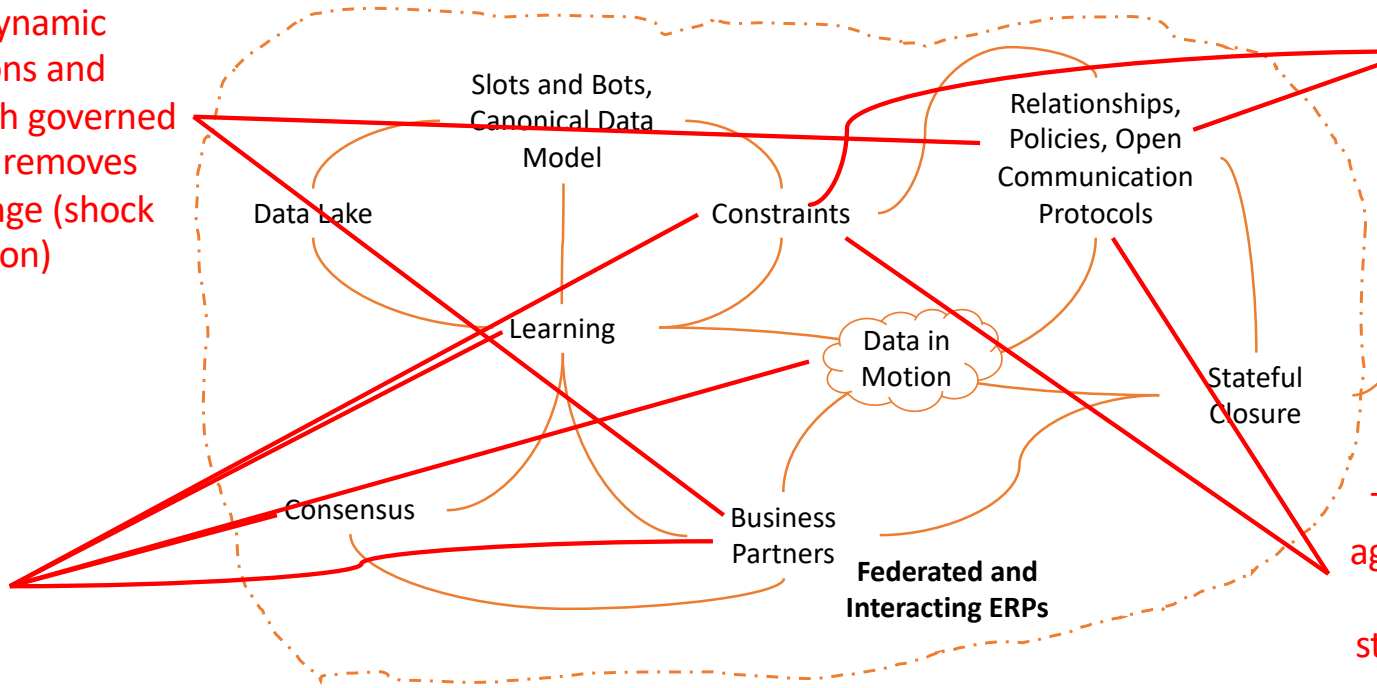


Second Order Solutions



Second Order Solutions

Business partners as role players in dynamic collaborations and interactions, with governed participation, removes friction of change (shock absorption)



Guard rails around Trust Parameters (trust but verify)

Federated and Loosely Coupled Organizations and Business Architectures

Trust via constraints and agreed policies rather than organizational control structures that don't scale

Consensus as a framework for points of view on validity, fairness

21st C Learning Architectures for Highly Entangled Systems

(e.g., today's Supply Chains)

20th C architectures snapshot current business ecosystem and organizational architecture thinking, but it does so in a way that hardwires it into code and makes it difficult to evolve with change. 21st C architectures must not fix or snapshot knowledge to a moment in time as the time value of truth vis-à-vis how we evolve and improve ways to do work shortens ever more frequently

Requiring policies and constraints as structured data enables architectures to curate knowledge that grows over time both generally and contextually. And exposing decision points and guard rails that can be reached via *data-in-motion*-styled pipelines provides the means for new knowledge to traverse edge to core of our architectures, and from core to edge (see forthcoming paper)

Whereas 20th C architectures are challenged to evolve with frequent change, 21st C architectures must more formally manage and apply knowledge that grows and specializes because the pace of change will only increase, and our systems must be adaptive throughout

We must implement architectures in ways that allow them to take in and apply new knowledge…

(This all might seem superficial. If interested in some of the technical depth implied, please see the notes to this slide)

Learning is fundamental and core… constructs that are fundamental to applying learning and knowledge in an architecture must traverse from edge to core and back

Learning is not just AI/ML

Snapshots of change lead to learnings (hence importance to capture policy, etc., as data)

Learning and its application are fundamental to supporting trust in architecture

Anything honoring improvisation leads to learning

Learning is action. A learning architecture and an architecture that facilitates learning are not the same.

An architecture that facilitates learning is one that:

Helps to capture and curate policy, preference, and personalization as structured data, understandable by people and machines, to guide the conduct of work, whether automated, performed by humans, or both;

Helps to apply what has been learned in work actions (from edge to core back to edge); and

Captures context relating to where knowledge is applied to both enrich it and to learn, in the platform, where knowledge application might be appropriate or not

A learning architecture is one that leverages context to internally learn and improve

Metadata capturing interaction contexts and corresponding outcomes (we've not mined metadata in systems we've built in the past)

Parameter tuning (hyper-parameterization)

Significance and relevance in data sets and constraints on them

Dimensions of a learning architecture:

- Constraints on data and on workflows both in systems and between people
  - Policies, business rules, constraints, etc., can be implemented as data (not code, which would require code to be updated in software releases, which resembles 20th C ERP systems)
  - These don't evolve and learn on their own, but they are simpler to modify and release
  - Constraints expressed as data make it possible to personalize, even to preferences
- AI/ML classification and identification
  - Narrow AI applications can be used to recognize and clarify
  - These are statistics-based models where data truly is code
  - These models can more easily evolve than code and be distributed more easily as well. They evolve with

data collected over time, potentially with data enrichments as well

- Canonical Data Model/Data Lake Curation
    - Being data driven will require data curation, i.e., schema compliance, data enrichment, etc.
    - Further, data analysis and ETL need to happen while data is in motion, not after it may be persisted to some database, because the time required to persist it and transform it might negatively delay decisioning.
    - We use the term Canonical Data Model to reference the definition of data sets that can be used as arguments to AI/ML models, etc. They are schema-compliant by the time they're presented for analysis (common structure, common semantic model, and dynamic/runtime constraint compliant). They can be persisted later (after use), or they can be published as events, etc.
- Interactions – constraint-driven workflows that focus on outcome, not process
    - Constraint-driven workflows represent families of directed graph workflows … they produce the same outcome, but they implement slightly differently (or maybe radically differently). As the number of partnerships increase, and with the impossibility to integrate between multiple parties at scale without tremendous cost and without tremendous difficulty due to the need to evolve code through changes in business, platforms, etc., we need to be able to orchestrate flows without "drawing graphs", as graphs typically behave similarly to code (republish with modifications, etc.). Constraints can be expressed as data that can be interpreted by code (interpreter code doesn't change as frequently as the data it manipulates).
    - Because constraint driven flows are not code, they can be personalized to specific combinations of role players, which might be part of multiple companies. Constraint-driven flows actually simplify technology interactions between multiple parties. The ability to personalize but still enforce common constraints on outcome simplifies setting up and tearing down work groups.
    - Note also that workflows formed using constraints simplifies integrating policies, preferences, and other constraints that form the guard rails on outcome into workflow.

These are some dimensions of learning, not all. Nothing above mentions domain/context or intention… these, too, are fundamental to constructing an architecture that can learn at the edge, take what is learned into the core, and then apply what has been learned in decisioning, classification, etc.

Consensus – more than just a blockchain construct

Consensus in blockchain applications validates what is written to *a block* via proof of work or stake… while blockchain may prove useful in certain areas, consensus can be leveraged even further

As an architecture pattern, it represents a framework for applying learning via points of view to validate, judge as fair using algorithms (analog to AI/ML algorithm ensemble)

Consensus as an architecture pattern is a framework to apply learning in specific contexts, and it is foundational with policy in forming guard rails of trust (trust, but verify)

Apropos to more than Supply Chain

The preceding sketch of architecture capabilities applie**s** to next generation ERPs *in general*

> They are foundational to multi-country insurance platforms developing in Asia where challenges of country-specific regulatory compliance, carrier and line of business preferences, and service request processing rules are expressed as structured data – rather than code – to evolve more easily with change and variations by country. This is a form of personalization at scale
>
> Supply chain and logistics platforms add to these capabilities, e.g., insurance has less need of real time, but Supply Chain, IIoT, and OT (Operational Things or Technologies), obviously add requirements of real-time, weather, etc., underscoring the importance of data and ecosystem

Federated organizations won't be supportable *at scale* unless policy and governance can be expressed as structured data

We view a supply chain implementation to be an implementation of a complex system that can entangled in its own right (without our technology entanglements added). So what we've discussed to this point is actually bigger than supply chain in many respects.

Supply chain also adds a time dimension that some large ERP platforms don't have. Supply chains are impacted by parts demand and weather and fuel costs, etc.. All of these can be represented by signals that need to be correlated together to trigger re-planning, re-routing, etc.

These are bigger than "single company" concepts … "a Global Supply Chain that shares global warehouses and inventory" represents an interesting extreme to which the architecture we're discussing here could go.

Implementing a new ERP, especially as Greenfield, may be critical but not be as risky as you may think

With risk aversion as *a justification to kick architectural thinking down the road*, we often insist on dragging brownfield liabilities into the future with the illogical optimism that we can evolve and modernize them more cost effectively that starting from a *clean slate*…

But consider what *clean slate* means in the 21st C:

> *Did you know you can use AWS Connect to implement the IVR (interactive voice response, voice automation in a call center) side of a call center's self service capability?*

Also consider a strategy to implement what you have place today using a new platform that gives you the flexibility to change in ways discussed here

> Mimicking what you have can avoid the shock of *new* to your organization, but can also position for change…

And consider the cost of not rethinking both business and technology architectures:

- 20th C architectures hardwire policies, processes, and context in the platforms based on them. These are fit for a specific purpose, but don't accommodate change, personalization, or ecosystem
- Brownfield compromises agency by hardwiring business logic and constraints into software that cannot be modified in situ. Deviation causes exceptions that often must be manually managed (a problem if these exceptions grow in number)
- In-built and prescriptive workflows result in inability to easily replan and re-route as change occurs… especially if the platform is to be *context aware* and learn in that context
- We cannot solve ecosystem problems with business and technology architectures that ignore ecosystems

Cloud services are increasingly powerful. They are more than simply direct analogs to open-source frameworks… they're big things like "Call Centers" and speech to text services, whole integration frameworks with which to transform data, and AI/ML platforms.

Starting "from scratch" is an incredible misnomer … viz. AWS Connect/Call Center as starting point is far from what we'd think to be Greenfield

Often efforts to build next generation cloud platforms turn into fiascos because the effort turns into reimagining the platform at the same time new capabilities are learned. One approach that should be considered is to build with new tools what is done today… The business and organizational impact need not suffer because of trying to do all things new… further, moving to a new platform and new ways to work impacts organization… it isn't just a technical architecture modification.

What does this mean for Supply Chain implementations?

Change is the new normal, and collaboration must scale beyond traditional corporate boundaries to federated and end-to-end partnerships to scale… partnerships that can be set up and torn down on a project basis…

Partnerships spanning corporate boundaries require agreement to conform to explicitly stated policies and regulatory constraints, and global/ecosystem visibility to value chain events, and sharing/ collaboration with respect to demand, etc.

> We can't depend on corporate organizational models and ERPs with hardwired policies / workflows (where the runtime is not introspect-able) to govern work
>
> We must implement technology foundations for trust and learning, and we must replace our process-focused means to govern with an outcome-based means to govern – this will enable interoperability between partners without forcing tightly coupled integrations

We must think in terms of *flows* and *not stocks*, otherwise we create our own (unhinged) belief systems and assumptions that future shock waves will stress, and stress waves may be worse than the incident shock

We must position ourselves to improvise, which quite likely means charting a course for greenfield futures

Key Next-Generation ERP Drivers/Pivots

Ecosystem rather than corporate-only view

Governance via policy and constraints (as data, not code), focusing on outcome (what needs to be done) and not how (a prescriptive dictated workflow)

- Many ways to work—potentially context sensitive/aware
- Multiple organizational architectures, e.g., federated

Improvisation and personalization at scale

- *Just in time* moves to *just in case* –
  the abilities to *read context and improvise*, if need be

Data driven, from stocks to flows

Learning, taken from edge to core

- Including context sensitive applications of trust, not just AI/ML classification

Organizational Jazz

Much of the discussion above is based on our focus on next generation organization and technology architectures which stress skilled listening and collaborative innovation (paper forthcoming)

The models with which we organize our companies and partnerships no longer need to be prescriptive, fixed, frozen to a snapshot of time. They can grow and learn, and people that become expert with them should be able to loosely federate and partner to realize business goals that scale to ecosystem levels

An analogy from music is *experts playing classical music vs. jazz*… applying this to our companies, and considering that *our technology architectures represent our organizational architectures*, our architectural transformation could be described as moving from *classical organization* to *organizational jazz*…

We have been working and writing on how to create a radical new way to support and amplify human agency, having many of the properties of musical jazz (both in form and in player groups) — a model for 21st office work… supporting human agency, improvisations and collaborative and individual learning.

Although a much bigger project than what we sketch here, we are surprised by the amount of technical (mostly) and organizational overlap and synergy between them.

Our architectures must become more adaptive to change. The business interactions conducted through them must be more explicitly shaped and governed with policies and guard rails that can evolve and be context sensitive

The models with which we organize our companies and partnerships no longer need to be prescriptive, fixed, frozen to a snapshot of time. They can grow and learn, and people that become expert with them should be able to loosely federate and partner to realize business goals that scale to ecosystem levels

With the affordances of technology innovations and learning of how to manage global and distributed business relationships, we can transition from scripted and *classic* business, constructed with fixed constraints as in 20th C, to more adaptive organization and technology architectures that honor business form without locking into traditional organization and process, leveraging technology to be more data driven and flow minded

Physical constraints of architecture limit and force work to be conducted in ways that were imagined when the architecture was implemented. Honoring form without forcing it with more rigid constraints gives freedom to judiciously relax constraints when working… for those with experience of when this can be done to realize a goal, without causing harm

Key Takeaways

- Learning by humans and machines is fundamental and core… constructs that are fundamental to applying learning and knowledge in an architecture must traverse from edge to core and back
  - Learning is not just AI/ML
  - Snapshots of change lead to learnings (hence importance to capture policy, etc., as data)
  - Learning and its application are fundamental to supporting trust in architecture
  - Anything honoring improvisation can lead to learning by both humans and machines
- Learning is action. A learning architecture and an architecture that facilitates learning are not the same
- An architecture that facilitates learning:
  - Helps to capture and curate policy, preference, and personalization as structured data, understandable by people and machines, to guide the conduct of work, whether automated, performed by humans, or both;
  - Helps to apply what has been learned in work actions (from edge to core back to edge); and
  - Captures context relating to where knowledge is applied to both enrich it and to learn, in the platform, where knowledge application might be appropriate or not
- A learning architecture is one that leverages context to internally learn and improve using:
  - Metadata capturing interaction contexts and corresponding outcomes (we've not mined metadata in systems we've built in the past)
  - Parameter tuning (hyper-parameterization)
  - Significance and relevance in data sets and constraints on them
- The above is an example of a generative dance between humans and machines
  - This goes beyond basic thinking, e.g., bot, and requires knowledge representation and knowledge graphs

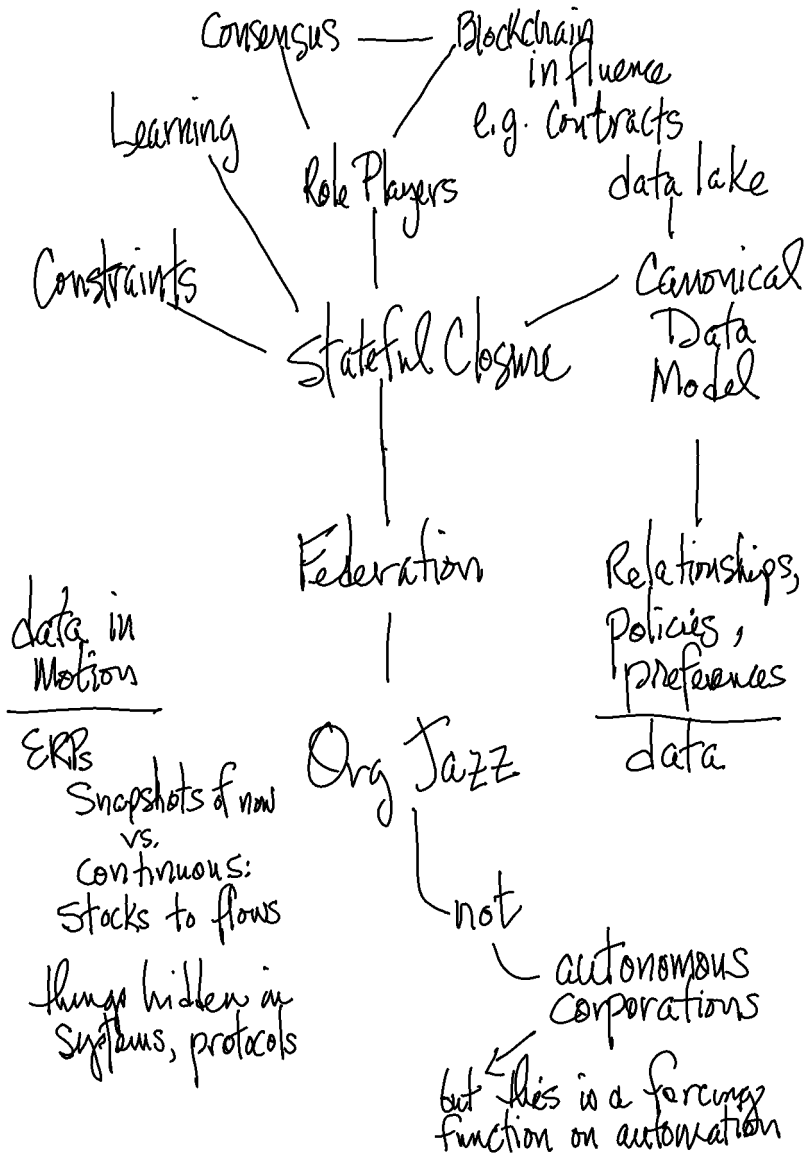
Key Takeaways

- Architecture layering in a 21st C architecture may differ considerably from 20th C layering (user interface, business logic, database)
  - Pipelines provide knowledge conduits in and out of this next generation architecture, establishing foundation of conversational and learning architectures
- Why Whitewater?
  - Captures context and speed that past generation architectures don't accommodate
  - First order problems are that we snapshot in time knowledge and represent it as code… we also hardwire organizational assumptions
  - These are fine when pace of change is slow, and organizational model changes but we color in the lines
  - Otherwise not so much…

Key Takeaways:

Conversation Accelerator

(Day 1)




New
Practice
Levels
Learn
Scale
Trust
Improvisation
Ways
Honor
Firm
Architectures
Listen
Ecosystem
Case
Federated
Guard
Beyond
Control
Act
Vs
Reflections
Optimization
Enhancing
Central
Agency
Time
Rails
One

Acknowledgment

We both have been associated with Li & Fung, one of the largest retail supply chains globally:

TBW worked with Li & Fung to conceptualize a next generation cloud-based Supply Chain and Retail ERP system 7 years ago.

JSB has served as a long-term advisor to Victor Fung starting in 2005, focusing on workplace learning in supply orchestration; he's been a member of the Fung Academy since 2013. JSB also worked with Leonard Lane Managing Director of the Fung Academy and senior advisor to Victor Fung since 2005.

Glossary

Glossary

Term	Definition/Elaboration
architecture	Architecture refers to the fundamental structures of a system, the discipline of creating such structures and systems, and the tools used in so doing. We focus attention on both business/organization and technology architectures as we think of 21st century architectures, and the constraints that each puts on the other. For example, we've seen a centralized control model applied to managing 20th century businesses which was very effective given then modern communication protocols and methods, a pervasive stocks mentality with its assumptions on the pace of information flow and corresponding signal processing. However, network, cloud, and compute technology advances far exceed those in place even 15 years ago, and the affordances of such include leveraging resulting capabilities to organize and architect differently.
autonomous company	An autonomous company, sometimes called a **decentralized autonomous organization** or a **decentralized autonomous corporation**, is an organization represented by rules encoded as a computer program that is transparent, controlled by the organization members and not influenced by a central government (Wikipedia). These terms initially were linked to blockchain technology in which Smart Contracts are used to contract transactions between other autonomous companies, with governing business logic expressed as contracts stored on a blockchain. It is an entity that lives on the Internet and exists autonomously, but also heavily relies on hiring individuals to perform certain tasks that ... automaton itself cannot...". Vitalik Buterin, 2014 This type of company can be viewed to be an extreme of software automation.
brownfield	Brownfield speaks to reusing legacy software platforms as a technology foundation for implementing new software capabilities, and operating within legacy constraints. Building brownfield software platforms in Public Cloud means building new capabilities using the foundation in place, which may preclude use of cloud services at all beyond basic hosting services.

Glossary

Term	Definition/Elaboration
canonical data model	A metamodel to define data elements (their types, static and runtime constraints on them), and data sets made of data elements (and their static and runtime constraints) and data model definitions conforming to the above metamodel (classes) and corresponding (schema compliant) data sets (instances of these data model definitions). Data sets can be used as arguments to web service invocations, arguments to AI/ML algorithms
consensus	Consensus in a blockchain is a process through which validation of data to be stored on the blockchain occurs. The process engages multiple parties, requiring each to come to its own point of view, and a majority position determines accepting as valid or rejecting. The consensus process is similar to ensemble methods in machine learning that combine insights obtained from multiple learning models to facilitate accurate and improved decisions. We apply consensus based on what has been "learned" to date in a system to a next generation ERP, which is required to descriminate in narrow AI applications like classifying fraudulent insurance claim request, but also applying points of view to decisioning strategy pivots in procurement or logistics. We believe that a next generation ERP will have to be able to reason over knowledge as well as classify situations, intentions, etc., so the architectural construct with which consensus can be applied in specific contexts is a fundamental building block in a learning architecture
constraint	A constraint is a condition imposed on a system or set of objects or data elements to meet one or more requirements on the system, objects, or data elements. Constraints in software can be manifest as rules processed by an inference engine, conditions on data imposed by solvers and optimizers, logical constructs with which compliance policies are enforced, or logical constructs that are used to effect a family of workflows (vs. a single workflow defined as a graph, which represents a specific instantiation of a workflow (not a workflow family))

Glossary

Term	Definition/Elaboration
contextually aware	Context awareness refers, in information and communication technologies, to a capability to take into account the situation of entities, which may be users or devices, but are not limited to those. Location, time, and business ecosystem are examples of contexts or context-related concepts that are fundamental to decisioning, classifying, and personalizing in a modern software platform.
data in motion	Data in motion is a term used to label any digital information that is being transferred from one location to another. It is also commonly referred to as data in transit or data in flight. When the data is finally contained in one location, it becomes data at rest. With modern data communication speed, and considering the life cycle of data, processing data in motion becomes fundamentally important to timely decisioning since the time to persist data and then analyze it could exceed the life of the data element. Internet of Things signals are examples of data whose value may be valid only for short periods of time. Given that these usually come in data streams, persisting each value and then decisioning based on what is safely stored means analysis may be worthless and will be late.
data lake	A data lake is a system or repository of data stored in its natural/raw format, usually object blobs or files. When we think of data lakes today, we often think of storing and managing them in a public cloud because they can grow rapidly in size, and "storage in the cloud" can be cheap (at least in comparison to traditional data storage CAPEX considerations). While data in a data lake is potentially stored in raw form, we consider it important to recognize that the act of curating data is assumed in data lake management -- the process of taking unstructured information to a more structured and even schema-compliant form is fundamental to use of modern algorithmic processing (reference canonical data set in this glossary). It also is important to note that data lake as a kind of data "junk drawer" isn't wrng either: information in a data lake may not all be used today in data processing applications, but it could become immensely valuable over time as we learn more in the contexts where we work. These unused data elements might point to needed investment to further curate the data lake to increase its accuracy or expand its applicability and relevance.

Glossary

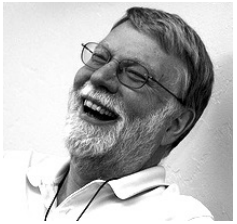
Term	Definition/Elaboration
Enterprise Resource Planning (ERP)	Enterprise resource planning (ERP) is a process used by companies to manage and integrate the important parts of their businesses. Software systems, also called ERPs, have been built as enterprise process integrations used in combination to run companies from a single system. ERP systems are known by many names, including electronic record management systems in the healthcare domain, manufacturing resource planning systems (MRP) in manufacturing domains, workforce and warehouse management systems, customer relationship management systems, and others. ERP platform implementations typically are single-tenant (focused on one company per implementation, including subsidiaries in more complex organization configurations). While configuration in these systems support some personalization at an organizational level, process implementations in a single ERP instance can be viewed as standardized across users.
federated organization	A collaboration between distinct organizations in which each participating organization assumes a specific role and set of responsibilities, and in which all participants pursue a common business goal or outcome. Participants in a federation are governed by agreed policies that effectively are rules shaping how work is effected and managed, and enforcing constraints on outcome.
greenfield	Greenfield speaks to beginning "from scratch", building fresh using current technologies. Building greenfield software platforms in Public Cloud means building software leveraging cloud services, e.g., storage services, compute services, and even Customer Support Call Centers, rather than hosting legacy software in a public cloud but ignoring all other cloud services.
pervasive network	A pervasive network is a large scale network that appears to operate almost everywhere using a large number of networks that self-organize to offer unified services. Here, we simply reference high bandwidth communication services that we can access almost ubiquitously.

Glossary

Term	Definition/Elaboration
protocol	A set of constraints, policies, and regulations can be called a protocol. As used in this document, protocol refers to the definition of ways that participants in a federated organization interact, i.e., how they communicate and collaboratively work together
RPA	Robotic process automation (RPA), also known as software robotics, uses automation technologies to mimic back-office tasks of human workers, such as extracting data, filling in forms, moving files, or performing more complex workflows in response to events published in software systems, etc. RPA enables automated performance of a specific activity sequence--it is process driven, and it is limited to performing human-scripted activities (no intelligence in the RPA, per se).
signal processing	We use the term "signal processing" in architectures as the processing of "system" events over time, correlating them into higher order signals used to trigger system behavior.
stateful closure	In software programming languages, a closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment). In other words, a closure gives you access to an outer function's scope from an inner function. In LISP and many of the modern programming languages influenced by it, closures are created every time a function is created, at function creation time. In supporting programming languages, closures can be unnamed or anonymous -- as such, they can be viewed as unbounded or unanchored to specific object or class hierarchies. Closures sometimes are called *stateful closures* to call specific attention to the fact that they bind together both function and data (closures can be viewed as similar to Objects as a programming paradigm).
stocks to flows	Stocks and flows are the foundation of system dynamics modeling. Stocks are entities that can accumulate or be depleted. Flows are entities that make stocks increase or decrease. Production (which increases stocks) and purchases by consumers (which deplete stocks) are examples of flows. Only flows change stocks. In some sense, flows represent events or actions that transpire and result in changes in stocks. If flow speed is fast, or if the shelf-life of stocks is brief, it can be more important and more accurate to pay attention to flows than stocks to get a sense of current state, and to participate in stock replenishment. Knowledge can be viewed as a stock with a decreasing shelf-life, and knowledge creation can be viewed as a flow. (Power of Pull, JSB et al) Our use of it in this and related documents pertains to being event and data driven and contextually aware -- processing data in motion to decision and work.

Authors

Tom Winans is a software engineer by training and a consultant by profession. He's conducted diligence work and consulted globally in healthcare, insurance and financial services, and other domains. His consulting work focuses on imagining and implementing multi-country architectures that leverage cloud technology innovations into personalized direct-to-consumer engagement at scale, create data pipelines that curate data in motion en route to narrow AI-based decisioning deep into an architecture core as a step toward implementing learning architectures, and orchestrate constraint-based, personalized, and outcome-focused technology workflows needed in more dynamic multi-party interactions and automations.



John Seely Brown is a master integrator and instigator of productive friction who explores whitespace between disciplines and builds bridges between disparate organizations and ideas. In his more than two decades at PARC, he transformed the organization into a truly multidisciplinary research center at the creative edge of applied technology and design, integrating social sciences and arts into the traditional physics and computer science research and expanding the role of corporate research to include topics such as the management of radical innovation, organizational learning, complex adaptive systems, and nano-technologies. JSB is currently a visiting scholar and advisor to the Provost at the University of Southern California (USC) where he facilitates collaboration between the schools for Communication and Media and the Institute for Creative Technologies (ICT). His prolific interests leads him to pursue research on institutional innovation and a reimagined work environment built on digital culture, ubiquitous computing, and the need for constant learning and adaptability. He co-founded the Institute for Research on Learning (IRL). His personal research interests include digital youth culture, digital media, and the application of technology to fundamentally rethink the nature of work and institutional architectures in order to enable deep learning across organizational boundaries – in brief, to design for emergence in a constantly changing world.