Synchronized Commerce
Computable Contracts & Ecosystems

Author: Scott Nelson
May 2020

Notes on whitepaper

The use of the words “agreement” and “contract” are very purposeful in this document. The word “agreement” is used to cover both informal agreements and formal written contracts. The word “contract” is used to describe a formal written legal contract between parties.

This document is intended to be around for a while and represents a vision of a future platform, therefore, the document is written in present tense. At the time of writing this technology has been in development for two years. To complete, the platform will require more time and other organizations to contribute to the open source base.

This paper frequently uses the term “ecosystem”. This is meant to refer to a business ecosystem as a network of organizations within a business sector or geography — including suppliers, distributors, customers, competitors, government agencies, and so on that are independently related to the delivery of products and services.

Table of contents

Preface

This paper is part of a series of papers about Synchronized Commerce. These papers describe what it takes to create consistency, symmetry and confidence within commercial activity. This can be done by using a new form of automated contracts called computable contracts. When designed and used properly these computable contracts will enable win / win vs win / lose situations.

Inconsistency and asymmetry, fuelled by fear, ultimately power lies, killing and destruction. Inequity, injustice and the disproportionate pooling of power has rarely led to a good result through the open lens of history. This paper is the result of many years of relevant experience and also comes out of many failures and successes. It holds a small piece of the promise for a better future with greater prosperity for all.

Consistency

Consistency comes from being able to classify things. To classify we need a classification system.

One of the most important realizations in my life occurred at the British Museum exhibition on the History of Classification. This exhibition made the point that all advances in commerce, economics, medicine, science and society had been preceded by an advancement in classification. Whether as a way to classify temperature, time, information, molecules, mass, length, pathogens or value – it does not matter.

For example, saying it's "warm today" is significantly less precise than saying it's 20 degrees celsius. Until we had a system to classify temperature all we had was cold, cool, warm or hot. Depending on where you live 20 degrees celsius may be considered cold, cool, warm or hot. Classification produces clarity and clarity reduces confusion which enables consistency.

Consistency reduces fear and enables trade.

Symmetry

Symmetry of information is produced when we all share the same facts about a transaction. The opposite of symmetry is asymmetry which occurs when one party knows something important the other party does not know about a transaction.

Asymmetry of information is one of the greatest obstacles to economic growth. The lack of symmetry between two parties creates the opportunity for deception, fraud and unfair advantage.

Conversely, any market that has a high degree of information symmetry is highly liquid and efficient. Symmetry results in trust, I can know what you know about facts governing the price, quality, location or ability.

Symmetry reduces fear and enables trade.

Confidence

Confidence is produced by knowing that claims are valid and risks are known. Lack of confidence or worries that can't be resolved produce inaction. Worry literally causes people to freeze.

Risk drives the fear of "what if". What if results in; friction, cost and lost opportunity. What ifs causes us to freeze because we can't tell what the risks actually are. If risk is known, we can figure out how to mitigate the risk. Where risks are known and mitigated we can do more deals and create more value.

Confidence reduces fear and enables trade.

Why computable contracts help

Consistency is a byproduct of computable contracts built from common components. To work these common components require classification models to function. Classification creates consistency. Consistency of information is key to coordination and cooperation within ecosystems.

Symmetry is a side effect of standardized computable contracts. The ecosystems defined in this whitepaper must have standardized computable contracts to work.

Confidence requires the validation of claims, identification of risks and elimination of fraud. Computable contracts that classify risk in common ecosystem processes with settlement finality and identity verifications reduce worries. A reduction of worry increases confidence.

We get to choose

I believe in the existence of a creator who has allowed free choice, even though it can result in horrible things sometimes. He loved us so much He allowed us to make our own choices. Ecosystems driven by computable contracts can enable us to choose what economic world we want to live within, to some degree regardless of location.

As with all new technologies, what we choose to do with it will be either a blessing or a curse. If we choose wisely we will see prosperity. But, if we choose poorly, automation will only enslave us further.

This whitepaper

In this whitepaper I use the word ecosystems because I find people understand it but they are really more like sub-economies that aren't necessarily constrained by a border. That's right, they can transcend countries, cultures, and industries. Payment networks already do this today.

I believe the long term result of this technology will be to provide an ability for more of us to pick the ecosystems we want to be part of and support. These ecosystems will represent different ideas about many things, some of these ideas will succeed and some won't. The point is that the best ideas will be proven by their results.

Technology doesn't fix corrupt hearts but it can make it difficult to lie, cheat or steal.

Seeking blessings for our future, J. Scott Nelson

Overview

A goal of the Sweetbridge Synchronized Commerce platform is to enable a new culture of commerce that is frictionless, low risk and fair:

- Reducing the fear that we will lose, while others win,
- A platform for building ecosystems that foster safer, better and more profitable relationships in business,
- Ecosystems that ensure good actors are rewarded by making it difficult for bad actors to prosper, and
- Commerce that does not need to rely on fear and greed to succeed, with settlement finality and risk management built right into the contracting fabric.

Sweetbridge is an open source platform. Advances in computable contracting[1] are core to this open source platform's ability to create a new culture of commerce. Computational Contracts allow agreements between parties and even laws to be understood, executed and verified by computers.

The Sweetbridge platform has already been adopted by several consortiums and at the time of writing is still under development. Computable contracts are just one aspect of Synchronized Commerce. Identity, claim proofs, accounting, audit, data security and settlement are also aspects of the platform[2].

Though the platform is open source, it allows parties to create applications that are not open sourced. These applications can generate contracts from a model of the parties' relationships and business processes. These contracts can be part of commercialized applications that generate revenue within the Sweetbridge Synchronized Commerce platform.

These applications can be thought of as Apps that generate royalties or be open sourced. These Apps include computable contracts among other things that automate and validate commerce transactions. They can even automate accounting, audits and settlement.

This new technology can validate claims, financials and digital twins of products, assets, rights or risks in real-time. To do this, the platform must understand the agreements that inform the accounting treatments, rights, obligations and risks in transactions. This means agreements must be modeled in a way that computer algorithms or AIs can understand them with a high degree of accuracy.

Historically, areas of commerce with highly standardized and balanced contracts see reductions in court disputes and lower costs of capital. This is true to a greater degree when the contract creates settlement finality. Payment settlement networks, commodity exchange agreements, and ISDA Master Agreements are just a few examples where this has occurred.

Therefore, one of the goals of the Sweetbridge platform is to enable the creation of standardized contracts and legal components for computable contracts that increase settlement finality. This will

---

[1] See the University College London's description of Computable Contracts and Finance https://www.ucl.ac.uk/computer-science/research/research-groups/financial-computing-and-analytics/computable-contracts

[2] See www.sweetbridge.com for other whitepapers and research documentation

enable the platform to be used to create new ecosystem exchanges, marketplaces and other intermediaries to service ecosystems with frictionless commerce.

Commerce is frictionless when accounting, audits, banking, contracting, financing, risk management, settlement and validations can be done in near real-time. Ecosystems can then provide many of the capital formation, liquidity, risk management and optimization of production benefits that are currently only available to major corporations. The result is that these value chains can obtain many of the advantages of a major enterprise.

This is possible because we can now use Artificial Intelligence (AI), Distributed Ledger Technology (DLT), Internet of Things (IoT), machine vision, satellites and new types of scanners as well as personal devices to validate claims in near real-time. By using computable contracts that can model the rights, obligations, roles and state transitions turning on and being discharged within workflows, we can identify what must be verified to be trusted. This automates trust by verifying the process.

By storing these verifications in systems that ensure they can't be faked or hacked we can know what is and is not verified. When we know what is verified and what has historically been reliable we can measure risk more precisely. Risk that a party is lying, cheating or defrauding is therefore reduced.

This means we can enable decentralised value chains to act as if they were integrated value chains as if they were controlled by a single entity. Computable contracts that can be paired with verifications for events around location, quality and quantity are key to this becoming reality.

Therefore, building audits and verification of claims into the platform have been core objectives. These objectives drove the design of the Sweetbridge platform's computable contract design. The desire is to enable value chains to become ecosystems that compete instead of companies that compete. To enable this the Sweetbridge platform has built in ways that let the participants share in the value and benefits created by the ecosystem.

Sweetbridge is even building standardized computable contracts for value sharing that will enable a group of smaller organizations to share in an ecosystem's value. If the ecosystem becomes large enough and diversified enough it should be valued like major companies such as Apple, Alibaba or Amazon. Imagine the global impact of a group of small lot farmers linking up to form an ecosystem with the same economic advantages of a Big Ag firm.

We believe the benefit of acting together can provide a massive incentive to adopt the type of contracting described in this whitepaper. These benefits will be very tangible, things such as lower working capital cost, greater value liquidity, lower cost of insurance and better production optimization. Finally, this technology will lower fraud which now represents more than 6.05% of GDP and which has remarkably grown by 100% in the last 10 years[3].

[3] The Financial Cost of Fraud 2019 - Crowe Clark Whitehill, together with the University of Portsmouth's Centre for Counter Fraud Studies (CCFS)

The current environment

Most contracts in use today are not written to enable modeling by computers. There are five primary reasons for this:

- **Timing** – The technology to model agreements holistically has not existed until recently,
- **Skill** – Lawyers lack the required technical skills and cross disciplinary training to create these new computable contracts by themselves,
- **Motivation** – A lawyer's primary motivation is not to make their contracts computable and dynamic to facilitate frictionless commerce,
- **Business Model** – Law firms make money by selling time not encoding intellectual property in technology that generates revenue from use over time, and
- **Benefit** – Agreement modeling does not benefit the legal profession as much as Corporate GRC (Governance, Risk and Compliance), Audit, Capital Markets and Insurance sectors.

Contracts can't simply be converted

Converting existing contracts into data models will frequently fail to fully represent the intent of a relationship in a way that enables the intent to be accurately understood by computers. Existing contracts frequently contain ambiguous language by design, and this ambiguity requires a dispute resolution process to resolve. They are typically static in nature and fail to fully describe the intent in all relevant situations or define all possible events within the relationship's workflows.

The result is that existing contracts can only be modeled to various degrees of accuracy. This means there are risks when automating existing contracts around rights, obligations and risks. It means that workflows frequently are insufficiently defined to be modeled. For example, the way to deal with common types of failures may be undefined.

As a result, existing contracts can't be completely or accurately automated because today's contracts are designed to require some level of human intervention. This can be assisted with AIs but the cost of training the AI is significantly high. The better solution is to switch to newer standardized contracts designed from data models that lack today's weaknesses.

The Sweetbridge Synchronized Commerce platform is designed to deal with the reality of the legacy agreements that exist today. It can deal with transactions that aren't governed by a contract at all. In that case, some of the verification benefits to users of the platform are simply reduced. The platform can also work with PDFs of contracts even though it can't model them without converting them. Agreements can also be modeled incrementally or partially when some aspects of the agreement can't be or aren't yet modeled.

But, the platform provides a compelling advantage to both users and creators of new standardized computable contracts.

The case for standardized contracts

The platform allows for a simpler approach than dealing with existing contracts or converting contracts. Simply switch to using new standardized computable contracts gradually replacing the

agreements in use today. This is very cost effective because individual entity's legal costs are quite low when using standardized contract templates. This means they leverage the work done by others in an ecosystem instead of hiring a lawyer to write customized contracts, resulting in a much lower cost.

The platform enables ecosystems that already have the skill and can easily absorb the training to author highly standardized and computable contracts.

- Adaptive contracts that are based on the relationship and processes agreed to by the parties.
- Generated contracts that are actually written by the platform based on data models of the actual relationship instead of being written by hand.

This does not mean the parties will not use lawyers but that the lawyers' role broadens and will primarily focus on helping the parties think through the pros and cons of various contract templates and variables.

These standardized contracts are made up of standardized components that are designed by lawyers as well as accountants, industry process experts, computer scientists, economists and experts in ethics. In most cases, to realize the full benefits, a new computable contract requires all of these disciplines.

These contracts are driven from data that models and monitors the relationship as it evolves. Therefore they can even detect when the parties' processes and relationships change, notifying or recommending new Statements of Work (SOWs) or components. This enables computable contracts that are adaptive instead of static.

Lawyers that realize these changes offer an opportunity by adopting a new culture of legal work will likely be more successful than those who ignore the inevitability of these changes. This old culture exists in part because there was no objective way to measure contract compliance. Once that way is created, in the form of computable contracts, it will enable a shift in the motivation of legal work. The goal will become clarity and broad adoption which requires contracts that are fair and balanced, as only these types of contracts will become standards for their industries and use cases. Since the greater success will come from the broadest use of contracts that generate more royalties, authors will have an incentive to make contracts appeal to the needs of the widest audience.

For standardized contracts to be widely accepted they must be fair, creating win / win instead of win / lose. The default master services agreements available on the platform will take this approach. This is possible because they can be built on open source master service agreement components designed to simplify dispute resolution, liability and risk management that reside on the platform. These act like laws in the local jurisdiction that must be overridden explicitly when needed but provide a default set of rights and obligations, risk and mitigation components, and ecosystem dispute resolution processes.

Computable contracts as a business model

Contract component and template authors can earn royalties whenever their standardized contracts or components are used. These royalties are collected automatically and reward the intellectual property work. They also subsidize the ongoing maintenance that is continually needed to keep

these contracts up to date with changes in laws. This eliminates the need for each individual party to pay for legal fees.

Authors can compose new contracts from open source components or share a royalty when they use components created by others. All of this is automatic and done through a computable intellectual property (IP) contract enforced by the platform that governs the use of others' intellectual property.

These standardized contracts are designed to take the cost, frustration and risk out of legal agreements while minimizing the likelihood of disputes. They operate under master agreements that create settlement finality whenever they are used. These standardized contracts are created by templates that can be highly configurable. By using these standardized contracts anyone can gradually replace existing contracts between counterparties over time.

The result is a contract that can be modeled by a computer to a very high degree of accuracy and a new age of commerce. Commerce where:

- Payments happen faster,
- Working capital is cheaper,
- Risks are lower and known,
- Disputes are fewer and simpler,
- Insurance cost less and is built into transactions,
- Accounting is automatic,
- Real-time audits happen automatically, and
- Lying, cheating and fraud are difficult.

These benefits are so substantial that the cost of developing standardized computable contracts for an ecosystem can easily be justified. This enables consortium based business models which create these standardized contract components and templates to generate income from royalties. These consortiums can be made up of multiple disciplines, institutions and industry participants within a legal jurisdiction or spanning multiple jurisdictions for an industry sector.

By designing templates with the needs of financing entities, insurance companies and professional services firms in mind, a substantial cost reduction can be achieved for these entities. An ecosystem consortium can act as an intermediary to integrate these financial services into the master services contracts to enable significant benefits to all users. The resulting cost reduction and increase in benefits renders a small royalty on each transaction needed to fund computable contracts insignificant.

As an example, a royalty of 10 basis points on £10 billion worth of economic activity is £10M per year. This is more than enough to provide a ROI for investment and cover the ongoing maintenance of standardized contract templates in many industry sectors. The initial ecosystems will likely need to be grant funded by governments with significant participation by researchers and universities. However, once proven models exist within several sectors, grant funding will likely no longer be needed.

An ecosystem-governed entity that resolves disputes could also be funded by these royalties.

What’s in an agreement

Contracts define a set of rules, procedures and understandings between parties in a relationship. The parties in a contract may have roles such as buyer or seller. These roles frequently participate in a commercial workflow made up of multiple states or milestones that change when specific events occur.

Each new state can result in changes to rights or obligations for the parties. Contractually granted rights for one party always result in an obligation for another party. All rights and obligations carry risk, the risk that the counterparty won’t fulfill their obligation if nothing else. Risks may have risk mitigations that are required by the contract to contractually transfer risk to another party.

Agreements may be formal contracts or informal arrangements and both are extensions to the laws and regulations that exist in the legal jurisdictions of the transactions that occur under the contract. Contracts can sometimes be used to override the legal requirements or laws of the local jurisdiction, while in other cases this may be prohibited.

All of this must be accurately modeled in data to correctly understand what rights, obligations, risks and mitigations are in existence on a commercial transaction as it moves through a workflow.

The term computable contract in this document means something much more than a “smart contract”. The term smart contract commonly refers to a snippet of code which is neither smart nor a contract in the legal sense. The phrase “smart contracts” has been widely used in the blockchain space and should not be confused with computable contracts.

A computable contract is a contract that a computer can “understand.” This means a computer can automatically assess whether the terms of the contract have been met, discharged and settled.

Therefore, a computable contract is one that is:

- A legally binding contract,
- A data model that fully expresses the meaning and intent of every clause of the contract in an unabligious way,
- The logic needed to validate and properly determine the state with a contract on a transaction as a result of events,
- A complete description of the data structure, events, workflows and states in the business process that represents the relationship between parties, and
- Anything else needed to make it possible to automate the discharge of the terms in a contract.

The Sweetbridge Synchronized Commerce platform builds on the significant academic work that has been done on computable contracts by universities such as Stanford, University College London and many others. The platform literally generates the written contracts from data models. The contract then becomes a living document that can be quickly updated and compared when changes need to be made.

This means that the contract documents become a user interface for lawyers, courts and legal professionals to use to computerize data models. Documents are the read-only representation, not the source of drafting the agreement. Instead of drafting agreements using the written word,

contracts are designed more like engineers design products by using pre-existing components. When needed new components can be designed and crafted to fulfill new needs.

The lack of formal agreements

The vast majority of business transactions occur without a formal contract between the parties. Business to customer (B2C) transactions don't have a contract between the buyer and seller. We don't sign contracts to eat at restaurants with the owner of the restaurant nor do we sign a contract when buying most things at retail shops, petrol stations or grocery outlets.

However, there may be other forms of agreements that cover some aspects of these transactions:

- On-line transactions frequently require the user to accept a "terms of use" agreement,
- Purchase orders, RFQ responses, bill of ladings or sales orders frequently have terms which cover aspects of the transaction,
- Consumer or business purchases using a credit card or debit card, and
- Laws or regulatory rules that cover aspects of a transaction.

Even business to business (B2B) transactions are frequently conducted simply by a purchase order or sales order without a formal agreement. Many businesses don't rely on contracts with some or all of their customers or suppliers. Instead, they rely on long standing relationships of trust or the law to protect themselves from negative consequences.

In B2C transactions consumer protection laws exist and many regulations were created to protect both B2C and B2B buyers from substandard, defective or dangerous products or services. These legal rules may be in force whether the parties know it or not. Sometimes these can be overridden by agreement of the parties in contracts and sometimes they can't. In many developed countries a primary reason for some contracting is to override the default set of rights and obligations established by local laws.

The growth of intermediary agreements

In recent years more and more B2C and B2B transactions are occurring via intermediaries such as exchanges, on-line marketplaces or e-retailers. The size of the retail e-commerce market alone is estimated to exceed $4tn[4] in 2020. Historically, marketplace intermediaries have been used for hundreds of years in financial services and capital markets such as bond, commodity, payment and stock exchanges.

Intermediaries frequently require users to use standardized contracts or to accept terms of use. As a result, the number of transactions that utilize standardized agreements is growing rapidly. The benefits of these online marketplaces and exchanges are so high that parties are typically willing to agree to these standard terms and conditions in order to access the services.

The value for standardized contracts

Standardized contracts have allowed the creation of new innovative services such as the creation of the world's largest fintech company, Alipay. The reason this is true is that the cost and effort of the

[4] See Statista https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/ on size of retail e-commerce sales worldwide.

analysis needed to become comfortable with the agreements is amortized over a large amount of economic activity. With a larger number of contract users, it is easier to get a high return on investment from the cost of the analysis. As a result insurance companies, financial markets, lawyers and others are willing to invest in understanding the risks associated with financing, derivatives or settlement based on these agreements.

Standardized contracts allow many parties to participate in crafting the contract components and templates. This is not practical with customized contracts unless the value of the agreement is very high. Over time, standardized agreements tend to mature so that they contain more and more of the rights and risk mitigations needed by non-contract parties to be comfortable with their inherent risks when working with one of the parties. This greatly reduces the friction in commerce and creates information symmetry which leads to lower cost and faster execution.

The 96-4 rule in commerce agreements

The Pareto Principle, commonly known as the 80-20 rule, asserts that 80% of outcomes (or outputs) result from 20% of all causes (or inputs) for any given event. If we apply this to standardising contracts then we should be able to standardize 80% of the transactions with 20% of the effort.

However, over a 25 year period while converting tens of thousands of complex commercial contracts into data models the author found that there is at least one more 80–20 that can be achieved by breaking contracts into components that standardize a part of a contract. See Common Contract Components below.

This means that 96% of agreements can be covered into standardized computable contracts for 4% or less of the effort of converting all agreements into standardized forms. This can be calculated as follows:

- 100% - (100% * 20%) = 80%
- 80% + (80% * 20%) = 96% of agreements
- 100% - (100% * 80%) = 20%
- 20% - (20% * 80%) = 4% of effort

The power of contract componentization and templates

Contract templates have been used in many industries in many industrialized nations for more than 50 years. In recent years, web based firms such as LegalZoom, Rocket Lawyer and others have created standardized legal contracts using web based templates for common individual and commercial agreements. Paper or web templates allow individuals, business owners or paralegals to draft agreements in a few minutes without the need of a lawyer to provide specific advice.

By creating standardized computable components for contracts that can display themselves as clauses in printed form we can enable s to participate in the creation of contract templates with minimal support and training. Many law firms and s already use a process of pasting together boiler plate language when crafting contracts. They already use tools such as word processing to populate variables or find and replace text. They are already used to some form of templatization.

However, the effort to create well thought-out components can be significant. Creation of components that provide the full benefits of computable contracts requires multiple disciplines. Law and regulation forms the basis, but computable components require many other disciplines. From accounting to audit, from GRC to ethics, from settlement to risk assessment, from insurance to risk mitigation, from banking to capital markets, all of these areas can be improved by our ability to accurately model a contract.

The link between contracts and economics

Like a well designed appliance or car a well designed contract serves the parties that use it in a variety of ways. Good design can reduce friction and enable valuable benefits by taking into account the other areas of commerce that are affected by a contract. By taking these other areas into account when we create computable contracts we can make businesses and economies more competitive.

We can even do things we might not have thought possible such as automate auditing of financial transactions in real-time, create new types of financial instruments and manage risks systemically. The most important of these new abilities is the creation of decentralized ecosystems of businesses that can obtain many of the benefits of a major corporation but without the need to sacrifice their sovereignty or dilute their equity. Capital formation, risk management and production optimization can be done at an ecosystem level or even at the level of an economy.

The link between contracts and supply chain events

For a computable contract to automate anything it must be aware of key events that trigger state transitions in a transaction between the parties. So if payment is to be made 15 days after goods are received, the event that starts the 15 day clock, “goods received” must be known. Computable contracts have significantly lower value if they aren’t integrated with information about the events that control state changes within a transaction workflow.

Some of these state changes can come from automated sources of information such as IoT devices or third party organizations such as the logistics company that delivers the goods. However, many of these state changes must come from humans or at least be “signed off” by humans. For example, if the payment is not due until 15 days after “acceptance of goods”, then not only the delivery but the acceptance meaning “the goods are in good condition” needs to be known.

Who signed or approved something is so common in business processes that until computable contracts can integrate with systems that know this information they will have less value. Integration with existing systems to obtain this information such as ERP systems may be possible in some cases but in many cases human validations are not recorded in trusted systems.

To address this problem the Sweetbridge platform is designed to allow personal devices to provide verifications of events occurring. These devices are ubiquitous and apps to verify information through pictures, pick lists and other means can easily verify who is using the device via device biometrics. These biometrics can identify the actual party “signing” the verification, their GPS location and the exact time.

Simple apps that can be configured for any verification or parties can be tailored to provide the equivalent of a very difficult to forge signature and state verification. The computable contract can actually configure these Apps to fit the contract event verification requirements in the contract itself. By integrating this with barcodes and other simple ways of capturing information from printed forms this verification process can be made very simple.

Supply chain drift

What a company needs from its supply chain changes over time. The shorter the product life cycles or the more project oriented the work, the more this tends to be true. The more dynamic the business activity or the more customized its product or service, the more the actual product or service will drift from what was thought to be needed at the point of contracting. This is known as supply chain drift.

Certain industries such as construction, electronic products and consulting services experience higher supply chain drift than others but all industries have some form of supply chain drift. The higher the supply chain drift the more likely that some or all of the agreement will no longer reflect what is needed, requested, or required by the parties during the period the contract is in force.

Supply chain drift introduces risk. This risk is reflected in higher cost and greater uncertainty about the ability of the counterparty to perform on time.

Today, industries that experience high supply chain drift typically have a common practice for working risks out. For example, in construction dispute processes or in consulting some form of padding the price or change order discipline is used.

Regardless, these take the form of de facto renegotiation, reconciliation or dispute resolution but rarely get reflected in an updated contract. Most of the time these de facto renegotiations are handled informally by the parties, particularly if they have long standing relationships. This presents a problem for computable contracts as they will no longer model the actual relationship; therefore, activities such as settlement can no longer be automatic.

The solution is to make the computable contracts adaptive based on what is actually happening between the parties. For a computable contract to monitor and automate transactions we must define the supply chain events, product verifications or sign offs, with changes to product or service specifications being updated within the data model. We don’t need to represent these fully in the data model, at a minimum we simply need to recognize change.

As these changes are verified and validated the computable contract can actually detect the supply chain drift. This is something that current paper-based contracts can’t do.

Because drift in the supply chain indicates a change in the relationship between the parties, computable contracts that are linked to the supply chain events can be used to ensure parties are made aware that a change is occurring. This enables more proactive negotiations and the ability for the contract to be adapted to the new reality – contractual position. The parties to the contract affected by the change can agree to the proposed changes creating an addendum to the contract that trues up the contract with the actual activity.

Alternatively, the parties can agree on a dispute settlement process to resolve the economic impacts of the drift after the fact. When this is the case, other parties that are relying on the contract progressing as planned, such as a financing entity, can be automatically notified. This may result in a change in their terms or services and so forth. Computable insurance policies, for example, could change premiums based on the actual risk instead of a blanket area of risk.

Economics in contracting

Finally, because the accounting can be automated as well as the contract, new forms of dispute resolution can be used that are based on financial information about the transaction, project or entities. This can include value sharing agreements that create an incentive system that balances the risk and rewards from supply chain drift events.

For example, in construction a disportionate level of risk to the project can exist from a very small subcontractor or low cost product. Identifying these areas of risk and designing incentive systems into the contracts can be done in a way that does not require dispute resolution. This would create significant incentives based on actual project economics to ensure the whole project comes in on time or under budget. These could reward parties for recovering the critical path timeline or shortening the duration of a critical path item to produce a buffer among other things.

The ability to recognize the contract no longer represents the actual relationship accurately is critically important to monitoring risk –particularly financial and performance risk. Banks, insurance firms and others can draft their contracts to react to risk that supply chain drift introduces within their value chain. This means they can enable risk adjusted terms and pricing.

The data and accounting information available to the computable contract makes the creation of incentive systems based on actual economics possible. In many industries, this will require entirely new skill sets for those creating contracts, as economics is not a discipline most lawyers possess. The involvement of economists in the design of well thought-out computable contracts means these new contracts can focus more on incentive alignment instead of fear of enforcement.

In mortgages alone the effect could be monumental on society, that effect could even substantially mitigate systemic risk in the banking community. Imagine a mortgage that does not use the threat of foreclosure but allows the homeowner to forgo payments in a mortgage. Instead, equity in the home could be used to make the payments without the need to forcefully remove the homeowner from their home and the bank ceases risking selling that home at a loss. This is possible with adaptive computable contracts.

The link between contracts and accounting

Legal and accounting are intricately intertwined with each other. Agreements between parties, whether contracts or informal agreements, inform the basis for proper account coding under GAAP or IFRS accounting rules. Agreement documents are a key part of the work papers used by an auditor, regulators or courts to understand the intent between the parties in a transaction. How a contract is written affects the accounting treatments, tax treatments, legal jurisdiction of the transactions and legal entities responsible for a transaction.

Lawyers frequently take accounting treatments, tax, and audit principles into consideration when drafting clauses to defend the treatment of their client's financial statements or tax returns. Contract clauses can be drafted to create specific accounting treatments, tax or jurisdiction advantages to benefit one or more of the parties. A legal agreement defines when assets or liabilities must be placed on, or taken off the books of a corporation.

A contract is an explicit agreement that works in combination with the law and regulations to define the rights and obligations between parties in an agreement. These rights and obligations operate in

a workflow which is hopefully described in the contract. These can trigger accounting changes or changes to risks when events cause transitions to a new state in the workflow.

The rights, obligations, risks and risk mitigations created by contracts must be understood by:

- Auditors to correctly determine reserves or footnote financial statements,
- Financing sources to correctly assess credit risk, and
- Trading partners or intermediaries to assess counterparty risk.

Computable contracts allow for the proper accounting treatment and tax categorization of transactions for all parties to be set when the contract is created and makes them easily modifiable over time as rules change. Since computable contracts are made up of standardized components, updating contracts for changes in accounting or regulatory rules can be automated in most cases. Even the identification of which agreements are affected by changes can be automatic. In addition, components that deal with legal requirements can be locked down so they can't be modified, to facilitate compliance with regional laws.

By embedding the accounting treatments at GAAP and IFRS roll up levels into the computable contract the treatment can be audited and signed off by an auditor in advance of any transactions being processed. The result is that every transaction processed can be audited for accounting treatment in real time. By modeling all parties accounting treatments in the standardized agreement we can tag each transaction with a contract and then have it signed off by an audit firm.

By using several audit firms to do the same audit on the standardized contract we create a computable contract which governs transactions which have already been audited and verified to have valid accounting or tax treatments by external audit firms. The result is a substantial reduction of the ability to commit fraud or mis-state the financial transactions on the books of a company.

As more and more computable contracts are used to govern the transactions of a company an increasing amount of the work in an audit is performed in real-time and at a greater level of granularity than possible in most financial audits today.

The entities that will have the easiest time adopting standardized contracts on 100% of their transactions are SMEs. Many SMEs don't have audited financials today, and even if they do they are likely only done once a year, decreasing the value to banks and other financial parties. The real-time nature of accounting that is possible with Sweetbridge Synchronized Commerce platform enables audited financial statements every day. This can lead to entirely new types of working capital and loan products with very low loan origination cost and real-time risk monitoring. These financing services can be baked into contracts, then utilized based on the users' needs.

Computable loans can enable debt service to be done out of the payment flow in real time, substantially reducing the risk of default while increasing payment data certainty. When these are interconnected in a supply chain the associated risks can be lowered even further. All of this real time information and analysis allows AIs to police network risk and can enable new forms of capital formation, other than bank financing and private equity, for the underserved middle risk areas of capital formation and debt financing.

Financial instruments like community investment bonds and other more creative structures become more realistic.

All of this is possible when accounting and legal can be modeled jointly, but full benefits of standardized computable contracts can't be fully realized without preventing identity fraud.

The link between contracts and Identity

Contracts exist between parties who must execute the contract for it to be in effect. These parties are either individuals or entities. Entities must have individuals sign the contract who are authorized to sign on behalf of the entity or the contract may not be valid. Therefore a computable contract must know that the signing individual represents one of the parties and that the signer has the authorization to bind the entity in the contract.

The identity of a signer must be proven to be genuinely the party signing. Today we use signatures, photo ids, notaries and in electronic environments usernames and passwords, mobile phone text, and apps like google authenticator among other things. When dealing with signers for an entity, bylaws, government company information sites, or board resolutions are frequently used to prove an individual is authorized to sign or bind the entity contractually.

Validating parties in transactions such as auditors, regulatory bodies, laboratories, inspectors, or certification entities are commonly used in commercial agreements to verify information. Weight receipts, customs inspections, material receipts, warehouse receipts and other documents frequently determine when events trigger state changes in contractual workflows.

Employees can call, text, email, or sign on through websites, submitting orders or sending invoices. Yet how does a counterparty know any of this information comes from real sources or authorized parties at those sources?

Increasingly barcodes, automated sensors, satellites and machine vision systems are used to verify information instead of human beings, or to automate the entry of data signed by a human. Data from EDI, EML, emailed excel spreadsheets and other file transfer mechanisms trigger transactions under contract terms and submit invoices. Evidence that these data sources are valid sources and are providing valid information must be proven in many commercial contracts. This means that validations or calibrations of the sensors, machine vision systems, and proofs that the transaction files that trigger computable contract state changes must come from valid sources. This means they must be verified.

Validation of the identity of products can be crucial. Knowing the provenance of materials is vital in many contracts. Knowing that the identity of the producer is genuine or the parts serial number is valid can be very important in some contracts. Computable contracts may need to know how to recognize valid identifiers on materials and products or in subsystems.

Individual Identity theft is a major problem in much of the world with more than 3 million[5] cases per year in the United States alone. A meaningful percentage of contracts are executed by parties that are not authorized by the entity they represent to execute the contract. Contracts are signed by admin staff on behalf of the authorized party faking their signature leading to significant cases of embezzlement and fraud. Automated systems and sensors are increasingly the target of hackers and technology fraud.

The people, entities, authorizations, automated systems, data sources and third parties all have identities which must be verified to enable a computable contract to be signed and transactions to

[5] https://www.iii.org/fact-statistic/facts-statistics-identity-theft-and-cybercrime

be automatically processed without the risk of fraud. Fraud now represents 6.05% of global GDP and has doubled in the last 10 years.[6]

The increase in fraud is partly attributed to the increased leverage criminals have by using technology and the vulnerability of commercial processes as a result of insecure automation.

The Sweetbridge Synchronized Commerce platform can utilize all of the existing methods for identity proof and verification but has new, more secure means of identity verification of people, entities, devices, systems and things as well.

Device and system identification

The Sweetbridge platform supports the use of device certificates to recognize and authenticate devices such as IoT sensors and personal devices such as cell phones or tablets. These certificates can also be used with data sources that can support them to verify the identity of the sender of data through APIs that support OAuth 2.0.

Individual authentication, authorization and delegation of responsibilities

The platform supports authentication of users' identities through personal device biometrics and self sovereign Know Your Customer (KYC'd) identities to tax identities such as a social security number or national tax id within a legal jurisdiction. A computable contracting system must include methods for proof of authorization for employees and representatives of companies. It can also support the delegation of authority to third parties to act on their behalf such as a power of attorney.

Entity identification and proof of authorization

Entities are validated through Know Your Business (KYB) processes that can be shared between entities and audited by both regulators and major accountancies. See our whitepaper on KYC and KYB.

The link between contracts and settlement

Commercial contracts frequently define the settlement processes between entities. Settlement is much more than just payment, it includes:

- The process, timing and terms of payment for goods and services,
- When and how assets are transferred between parties,
- When rights are triggered, transfered or cease to exist,
- When obligations occur and are discharged, and
- When and how risks are transferred.

For computable contracts to automate transactions they must accurately model these settlement processes and provide the means for transfers to occur. The Sweetbridge Synchronized Commerce platform does this by sharing common micro ledgers, which track and assure that settlement state and accounting state changes are atomic between the parties.

[6] The Financial Cost of Fraud 2019 - Crowe Clark Whitehill, together with the University of Portsmouth's Centre for Counter Fraud Studies (CCFS)

This means that an asset can't be on the books of two parties at the same time. It also means that there can't be a difference in the purpose for a payment and the application of the payment on the received books. This ensures precise timing when rights turn on and off, or when their obligations arise or are discharged, simultaneously between all parties in a computable contract.

The importance of settlement finality

Settlement finality is typically used to refer to the finality of payments made through a payment network. By "finality" we mean the transaction can't be reversed even if the parties in the transaction have a dispute, go bankrupt or fail. But this issue is much bigger than payment. This same issue exists when any asset, right or value is exchanged and any obligation is discharged.

Knowing the date and time when an aspect of a transaction can no longer be undone is critical to risk management. If something can be undone there is a risk it will occur. That risk has to be priced into all economic activity as uncertainty. Uncertainty creates fear which reduces the desire to take action.

This is a source of information asymmetry between parties. Information asymmetry always creates fear and risk in economic terms. The greater the asymmetry the more one party can take advantage of another party. The result in economic terms is lower liquidity, higher cost for risk and broader opportunity for profiteering.

Information asymmetry and risk are at the heart of the lack of liquidity in commerce and contribute to the practice of more powerful organizations pushing unfair and predatory contractual clauses off on others. The greater the asymmetry the more each party must look out for themselves. The real loser is our society and our economies which bear the brunt of the cost. It acts as a tax on everyone and increases the cost of trade.

This has been understood by banks and capital markets for centuries. Modern exchanges and payment networks are designed to create settlement finality. These parties use standardized contracts and operate in legal jurisdictions where settlement finality can be created to reduce the risk of transactions being challenged or undone. Modern capital markets, debt markets, credit card networks, derivatives and commodity exchanges could not function without settlement finality.

Derivatives get their name because they are contractual rights derived from other rights. A stock option for example is derived from the right of ownership of a stock. The value of the derivatives market is larger than all other asset classes combined. Derivatives can't exist without settlement finality and they can't be created without their underlying rights having settlement finality.

Few commercial contracts provide settlement finality and very few commercial contacts provide finality of state transition in workflows. This means that the timing of when assets transfer, rights are awarded or obligations are discharged is uncertain. The result is that risk is difficult to assess accurately in real-time as we must wait before we can trust the results. This process worked when we did everything with human intervention and review but won't work when we turn things over to computers to execute.

Finally, settlement finality is hard to create in many legal jurisdictions and impossible to create in some. The value of settlement finality is so high that the Sweetbridge Synchronized Commerce platform is designed for the creation of new types of exchanges and marketplaces, this platform can use the existing capital market infrastructure to enable business transactions to have the benefits of

our capital markets and is one of the primary reasons for creating master services agreements for ecosystem operators as part of the platform.

The link between contracts and laws, regulations and tax

Laws create common rights, obligations, rules or requirements that can be thought of as a common contract that all individuals or entities have with their society within a legal jurisdiction. Regulations add workflows or procedures with additional rights or obligations that act like a master SOWs within a legal jurisdiction defining roles and responsibilities of parties in commerce. Tax rules add a set of obligations, procedures and settlement processes that act like an additional party that is part of each agreement in a contract.

Some laws can be overridden by contractual agreement between the parties while others can't. Regulations rarely can be overridden by contract. Tax treatments are highly impacted by how a contract is written.

In order for contracts to fully express an accurate data model of an agreement they must take into account applicable laws, regulations and tax rules. The Sweetbridge platform is designed to allow for the encoding of laws, regulations and tax rules in jurisdiction master services agreements related to asset classes, industry segments and types of commercial activity.

These jurisdiction master services agreements can only partially model many laws, regulations and tax rules because they suffer from the same weaknesses as current contracts. However, regulatory bodies and government entities could use the same legal component tools to create regulatory enforcement technology (reg tech) that would be able to be accurately modeled in the same way as new computable contracts. The benefit to society would be that rules would be less ambiguous, reporting and regulations enforcement would be automatic.

The effort to create computable laws, regulations and tax rules for society would result in an immense reduction of asymmetry and friction. The effect would be substantial economic growth and greater prosperity for the societies which embrace this approach. The reduction of corruption alone would result in increases in tax revenue without increasing tax rates.

Another benefit would be to make it possible to model the effects of changes with precision and increase clarity for everyone in a society.

Sweetbridge realizes how long it will take to get governments to adopt new computable laws, regulations and tax rules. Therefore, it has designed a concept of ecosystems into the platform which can start to create master services agreements that bind participants contractually through ecosystems' laws, regulations and tax. This allows commercial bodies to forge their interpretation of legal and regulatory requirements into common "laws and regulations" with their interpretation of tax rules into computable contracts.

These bodies can partner with governmental approval and oversight to create micro economies with their own computable laws, regulations and tax collection processes as an early step to make it easier for governments to adopt these new approaches ultimately. This also has the benefit of allowing real commercial players to participate in the formation of these in a real citizen led governance process which is likely to be strongly supported by its participants.

Finally, these ecosystems can even compete with each other or transcend legal jurisdictional boundaries. This already occurs with much of the capital market infrastructure of the world.

Ecosystem as micro-economies

The Sweetbridge Synchronized Commerce platform is designed to allow consortiums to form ecosystems which have their own “laws” (ecosystem master services agreement), “regulations” (workflows) and “taxes” (fees). Computational contracts allow us to create ecosystems that individuals and entities can opt into regardless of legal jurisdiction. These ecosystems can have their own “enforcement” and dispute settlement entities with processes that can include dispute resolution, penalties and even loss of ability to participate.

This is nothing new. Guilds, trade associations, commodity exchanges, stock exchanges and other trade boards have long had some of these abilities. The difference is that computable “rules” and contracts allow us to do this to a level never before possible. They also allow us to do this in such a way that the corruption, so rampant in these bodies, can be made either impossible or transparent.

These ecosystems can create a much more resilient form of commercial structure which would serve our societies far better than major corporations as the pace of change continues to increase. These ecosystems can function with some of the same abilities as economies. They can have shared capital formation via ecosystem bonds or equity much like countries can have shared capital formation via treasury bills. They can have pooled risk management production optimization like major corporations.

Standardized computable contracts and accounting treatments audited in advance would enable the ecosystem to have common banking and insurance functions at significantly lower risk and cost. Trade within the ecosystem could even occur without currency by moving assets directly between balance sheets, greatly reducing the need for working capital and increasing liquidity. Major corporations and countries already trade trillions of dollars in assets annually without using currency. Banks exchange tens of trillions in derivatives monthly without using currency.

Partially self governing utility

The Sweetbridge platform is designed to fuse legal, accounting, and financial services such as audit or insurance into an ecosystem utility. This utility can be governed by the ecosystem itself and be operated as either a for-profit or membership owned entity. The goal is to enable shared infrastructure that reduces the cost of administrative, legal, financial and accounting processes.

By using computable accounting, contracting and workflow processes we can transform trade organizations and turn them into administrative and financial utilities. By giving each member of the ecosystem shares in the utility we can create shared value that has network effects with valuations of a tech unicorn – Apple, Google, etc.

Sweetbridge is enabling ownership through something other than equity: a royalty share. We issue the royalty shares based on economic contribution to the ecosystem. Thus we can create a new type of entity that can create new value for its members without changes to equity structure that can result in multiples of the value of their existing organizations. Sweetbridge has tokenized this royalty share into a digital asset that can be traded, creating a new form of liquidity and value for its owners.

Sweetbridge has started test marketing this concept in the construction industry within California. This is being done using economic models that predict the effects of adoption for each entity on their business based on assumptions. The results created a doubling of enterprise value on average for participants. We believe this will be more than enough incentive to change from current processes of contracting to new computable contracts.

Ecosystem links – Contract exchanges

Value chains for any business are complex networks that will cross multiple ecosystem boundaries. This means that special computable contract exchanges will be required to connect the commerce in one ecosystem to the commerce in another ecosystem. Additionally, the same issue exists between legal jurisdictions as differences in laws make it cumbersome or next to impossible to create a single agreement that works under multiple bodies of law.

These problems are not new as they exist today in any legal agreement that spans more than two parties or between legal jurisdictions. Payment networks for credit cards solve both of these problems by separating agreements into layers and forcing a common settlement agreement. The jurisdictional differences are handled by setting up a clearing house that each payment processor contracts with in a single location such as London.

Each party that works with the payment processors legal jurisdiction has a separate agreement. This adds an additional level of abstraction allowing the jurisdictional differences to be separated from the business agreement. The Sweetbridge platform is being developed to enable an ecosystem of ecosystems and jurisdiction-based exchanges to facilitate the conversion of trade between legal jurisdictions and across ecosystem master agreements.

This ecosystem of ecosystems will provide the common backbone for cross ecosystem transactions much like trade agreements provide a common framework for trade between economies. This means that countries could actually use computable contracts and accounting for their trade agreements. The result is that very fine grained inter-country agreements with automatic monitoring and verifications start to become possible. The impact this would have on reducing friction in trade could be quite significant.

The link between contracts and registries

Registries are frequently used to store knowledge about who owns an asset and what rights they have granted. Centralized common registries are used today in everything from property to stocks. These registries always have a scope such as an entity, exchange, or legal jurisdiction. In order for computers to automate transactions in a computable contract they will need the ability to interface with existing and new types of registries.

Common registries are critical to being able to verify and atomically change the ownership of an asset in the real world. The easiest of these to understand is real estate. Contracts related to the sale of real estate must be able to validate current ownership, determine what rights have been granted and what unsettled obligations against the property exist.

Computable contracts must be able to interface with and change the state in these registries to fully automate state change. The platform allows for manual state changes so this is not an absolute requirement but the ability to support registries is much more important than simply interfacing with

legacy manual processes. Stock exchanges, commodity exchanges, and even some automobile registries all have highly automated registries.

The more automated a registry for an asset class, the easier those assets can be used to secure debt and create new forms of liquidity. Currently there are significantly more illiquid assets in the world than there are liquid assets. In fact, if all assets were liquid, had a means to do price discovery, and existed on registries, there would be less need for currency in some business to business transactions.

The reason for this is that assets could be traded instead of one party needing to sell assets or borrow funds to raise the liquidity in currency to purchase another party's asset. Banks do this today with derivatives trading rights and obligations worth trillions of dollars every month without currency. Registries of assets would allow options and other types of derivatives to be used more widely with products and even services.

Registries are also important because they allow rights to be separated from assets which almost always unlocks unrealized value. The right to buy a stock at $5.00 has a value even if the stock is trading at $4.00. The sale of a 90 day option for $0.10 does not decrease its current $4.00 value but it provides the owner of the stock with two ways to monetize the stock, one is to sell it for $4.00 the other is to sell an option for $0.10.

This subdivision of rights is the basis of all derivatives. These always need some form of registry to protect the sale of the underlying asset or right without the right that has been sold to someone else being honored. They also protect the buyer. Registries allow the buyer to discover what rights have been granted on an asset. This ensures the buyer can discover if one or more of an asset’s inherent rights have been granted to another party. For example, the mineral rights having been sold to someone else on a property the buyer is purchasing.

Registries can take many forms but the Sweetbridge Synchronized Commerce platform is designed to allow the creation of asset or right specific registries by asset class within a legal jurisdiction. The reason this must be done at a jurisdictional level is the laws governing rights may not be the same from one legal jurisdiction to the next. Therefore, it is impossible to transfer some assets and rights between legal jurisdictions without taking legal, regulatory and tax implications into account which might require a different computable contract or changes to clauses.

Tradable rights and obligations

One of the most under appreciated benefits of computable contracts and registries (ledgers) is the ability to tokenize assets and rights. By tokenize we don’t necessarily mean a blockchain token, instead we mean a moniker that can be used to uniquely represent the asset’s or right’s ownership. Regardless of what technology is used, there is substantial value in an ability to trade assets and rights on those assets in a tokenized and particularly a fractionalized manner.

The ability to create liquidity through transferable fractions of ownership in rights and obligations can release the same benefits to commercial agreements for ecosystems that derivatives released in financial markets. The ability to hedge and pool risk such as the ability to diversify supply chain procurement in light of Covid-19 would be extremely valuable to decreasing supply chain disruption risks.

In the current legal environment many contracts limit the ability of parties to transfer or subcontract rights or obligations. The reasons for this are many but the effect is that the value available to all

parties is reduced. The Sweetbridge Synchronized Commerce platform is being designed to enable computable contracts with componentized rights and obligations. Computable contracts and ecosystems allow us to minimize many of the reasons for resisting transferable rights and obligations historically.

One of the biggest reasons for this restriction is the belief that one party will be able to realize a significant financial gain without the other party participating in the value or opportunity. Since it is difficult for each party to envision every possible way someone might extract value from a contract in the future, many contracts today tend to be written to require the counter parties approval.

The result of current practices unfortunately is that the counterparties risk on investment in developing an additional value is too great. This comes from a failure by all parties to understand basic behavioural economics. The current practice actually prevents the very activities that would help the parties participate in greater value. The reason this is true is that the counterparty can simply veto or hold the other party hostage to unfair terms.

Almost all capital market infrastructure is dependent on the ability to separate and trade rights on other rights or assets. The ability to do this has actually led to an increase in value of all underlying assets. The conversion to standardized computable contracts is an opportunity to correct these errors and unlock trillions of dollars in new value.

The concern that others may profit on my right or obligation in some unexpected way and fail to share this value or introduce risk can be solved by computable contracts and ecosystems just as they are by exchanges. The ability to do price discovery or write contracts that allow actions within measurable parameters that are validated by external parties enables new abilities to tap opportunity or manage risk.

Imagine a contract with a requirement for the delivery of 200 specific products on a specific date for a construction project. Let’s assume the obligation is for something on the critical path of the construction project. If it is not delivered on the specified date the project will be delayed at a cost of £100K per day. If the obligation for the product on the date is made tradeable and is specified in sufficient detail, a new type of insurance would be possible that would protect the project from being delayed due to items not being delivered on time. If the obligation is transferable, and product progress is tracked based on risk of non-delivery, some or all of the obligation could be automatically transferred potentially at a much higher cost to someone else with the difference paid by the insurance company to prevent a claim payout.

Contracts that enable value sharing can also be created with computable contracts and accounting within the Sweetbridge platform that actually incentivize parties to find new ways to generate value from current and existing assets. See the Sweetbridge whitepapers on the tokenization of assets and rights, the tokenization of assets and the tokenization of rights for a better understanding of the value that can be released in these areas.

The link between contracting and data security

Contracts, more and more frequently, contain clauses on the ownership of data and requirements for data security or access. To make computable contracts with clauses that place rights or obligations on data, the computer model must be able to govern access to data. This means that computable contracts must in some cases have an ability to know what data is “confidential”,

“owned”, etc. This requires a way of modeling ownership rights and access rights to data that can be enforced by contract terms.

This is one of the strongest non-economic arguments for forming an ecosystem data utility that can police data held in trust by one party for another. However, the history of creating a centralized repository of data is not great because it becomes a honey pot target for hackers or compromise. Once data is stolen it can easily be transferred anywhere and to any number of places. Encryption of data helps but still may have weaknesses.

The Sweetbridge platform enables a new paradigm of data sharing. Each party can retain its data within their firewalled control if desired. Instead of sharing data by placing it into the hands of other parties, the platform allows code to visit the data instead. This allows for interesting new types of algorithms that can prove something is true but without revealing more information. For example, a vendor could prove they have 25 units of a product in stock without revealing how many units they have in total.

This also allows computable contracts to enforce control or ownership of data which is required to enable data access and ownership clauses. Policing these clauses would be near impossible without the use of ecosystem data utilities that provide this as a standard service. The cost to retrofit existing data infrastructure would simply be too high.

Contracting model

In the prior section we have presented countless reasons why computable contracts will be too costly and require too much infrastructure to become widely used in commerce outside of shared infrastructure provided by ecosystem utilities. Unless computable contracts are integrated into common commerce infrastructure they will stay relegated to special uses or will be used only to partially automate commerce in the near future. This is why the Sweetbridge Synchronized Commerce platform is designed to provide the infrastructure to make ecosystems work.

Sweetbrdige strongly believes that new intermediaries such as ecosystem utilities will unlock so much value that computable contracts will first be widely adopted within these decentralized intermediaries.

Some centralized industry intermediaries exist today but they are typically weaker industry bodies that offer a fraction of the value of the ecosystem utilities being described. However, these existing bodies could quickly be transformed into decentralized ecosystems with common legal, financial, accounting, data security and insurance provided as a utility. In other cases, ecosystems will emerge in green field areas of commerce which have no legacy intermediaries that must be transformed and will develop from scratch.

The business model infrastructure of internet marketplaces are still relatively nascent so some of these will be able to convert into the new ecosystems. Some of these ecosystems may some day rival an Amazon or Alibaba for economic activity but without being centralized in their control.

The ecosystem utilities are only possible due to the creation of computable contracts which do not need to be centralized, but do need common infrastructure. Like electricity needed common distribution lines, current flow (AC vs DC) and voltage levels to make electricity useful, these utilities need common infrastructure. However, this infrastructure can be provided by networks of independently managed nodes that provide the common services.

These new decentralized intermediaries will be very different from industry intermediaries that extract value and are being displaced in so many markets. The new ecosystem utilities will offer smaller organizations the ability to access some of the benefits historically only available to a major corporation or national economy. They will provide new value only available by synchronizing commerce via common infrastructure that removes cost, friction and risk. Instead of adding cost and fostering information asymmetry these utilities will do the opposite.

For all of these reasons, intermediary vs party-to-party contracting is likely to represent the vast majority of use cases for computable contracts. The Sweetbridge platform is designed to support both. However, a significant focus has been placed on the need for intermediary-based master services contracts that would be accepted as the terms of use by members of an ecosystem.

Platform master services contract

A master services contract provides a set of terms that will apply to all agreements and subordinated contracts between parties. Depending on its design, these terms may or may not be able to be overridden by sub-contracts or SOWs. The SOWs typically define common terms and conditions that will apply to subordinated agreements without needing to be renegotiated. These are

extensively used in corporate contracting when there will be continual contract updates or new SOWs between the parties over time.

The Sweetbridge platform intends to provide an open source Platform Master Services Contract that can be used by various industries and jurisdictions as a template to create ecosystem master services contracts. The intent is to design these with a broad representation of disciplines to make them as useful as possible. Computable contracts require a different set of concerns to be addressed than the existing contracts in use today.

The need for broader skill sets when creating computable contracts could limit their adoption. Therefore, standardized master services contracts are one way Sweetbridge plans to address this barrier. The other is standardized legal components' provisions described later in this document.

Reference master services contracts

Reference master services contracts are initially being developed for several major industry segments in a few reference juridictions. These master services agreements were selected in the following areas based on existing consortiums and team experience:

- Food in the UK and US
- Construction in California and the UK
- Insurance in the UK and US
- Manufacturing supply chains in the UK and US
- Commodities in South Africa
- Payment processing in the EU, UK and US

These areas provide the initial reference contracts that will be used to model the creation of master services agreements in other industry areas and legal jurisdictions. The initial master services agreements will be designed for computable ecosystem contracts or party-to-party contracting. However, the Sweetbridge project is also designing master services contracts for party-to-party computable contracts.

Master ecosystem contracts

Computable ecosystem contracts are master services contracts and terms of use for the legal entity that represents an ecosystem who can act as a trusted intermediaries within the network of ecosystems. These entities can be compared to exchanges and central clearing houses that exist in the capital markets. The difference is that anyone with the ability can operate an ecosystem because the Sweetbridge platform is decentralized in its settlement and accounting activities.

These computable contracts are analogous to credit card network contracts except all parties in an ecosystem will share common master contracts within a legal jurisdiction. This means that parties contract with the ecosystem node under the master services contract. Much like using Amazon or Alibaba would require each party to accept a master services contract in their terms of use. These ecosystem legal entities collect fees and generate profit by providing common services needed to create a higher level of settlement finality, lower risk, shared data security, greater access to capital, etc. Ideally ecosystem entities would be non-profit or member run organizations.

These ecosystem entities will collectively run the distributed registries that are needed to verify and manage tokenization of rights or the creation of derivatives for their ecosystem. This will allow entities within the ecosystem to work together without the need to negotiate most of the non-transaction specific terms. Because most existing B2C and B2B transactions don't have contracts it should therefore be relatively easy to transition to ecosystems under standardized computable contracts for a specific industry.

Common statements of work

Common statements of work are being designed for use with either the ecosystem or common master services contracts to define a unique work product or service. These SOWs are made up of workflows, states, rights and obligations that reflect common practices within a culture and industry. The objective is to use pre-built components to construct new SOWs.

Statements of work primarily define requirements, processes and rules. There are only a limited number of processes and rules in any industry because the parties working together can't do everything uniquely. For multiple parties to create efficiency within coordinated activities they must use common processes and operate under common rules. This means there are relatively few actual processes and rules within any industry and culture combination.

Product or service specifications are also highly standardized in their structure, but not necessarily in their specifics. The very nature of product and service differentiation means there is almost an infinite number of specific requirements. However, there are shockingly few methods of classification systems that can be used to describe a product or service. This means there can be a high degree of commonality between the components that make up SOWs within an industry, but a wide variety of SOWs themselves.

Ecosystem statements of work

Ecosystem statements of work (SOW) serve two purposes. The first is to provide a library of common business specification methods, processes and rules supported in the ecosystem for ecosystem contracts. The second is the creation of ecosystem regulations. By creating standardized workflows, states, verifications, and certifications an ecosystem can create comparable products and services with a minimum level of quality or functionality.

Therefore, ecosystems can use ecosystem SOWs to create quality, data, reporting, labeling or other requirements that level the playing field and define minimum levels of operation within an ecosystem. This means that ecosystems can ensure members can be trusted to meet these levels of quality or performance. These can also be used to level government mandated requirements between legal jurisdictions. A food ecosystem in the EU might want to do this for members with organic produce that need to be able to be sold to both the EU and UK post Brexit.

Ecosystem SOWs can also be used to provide centralized financial services such as low cost receivables financing or supply chain financing to ecosystem members. By incorporating the requirements of the financing party, ecosystem SOWs can be added to any agreement to provide working capital.

Ecosystems would also use common SOWs for ecosystem services when the Ecosystem acts as a utility or provides common services to members.

Ecosystem interchange contracts

Ecosystems need a means for allowing members to trade with other ecosystems or within the same ecosystem across jurisdictions. Ecosystem interchange contracts define how contracts that span ecosystems are handled. They are notionally similar to a trade agreement between countries, defining the rights and obligations of the ecosystem to another ecosystem. These contracts are written between ecosystems, not between the parties contracting the transaction.

For example, an ecosystem contract between parties in the food industry in one country's ecosystem may need to source food from parties in another country's food ecosystem. The two ecosystems can contract together forming a greater level of trust and settlement assurance between the parties. Each food producer then contracts with their local ecosystem instead of with each other and the ecosystems rely on the contract between the ecosystems to handle the transaction.

By isolating the legal jurisdictional issues and any dispute resolution from the commercial agreement, the parties can deal within their legal framework locally. The computable contracts only need to manage issues within one legal jurisdiction. Issues such as counterparty risk on payment can be managed by the ecosystems.

Ecosystem fabric components and contracts

To facilitate transactions between ecosystems the Sweetbridge network will provide an ecosystem of ecosystems – the network ecosystem. This network ecosystem will act as a shared utility providing a common fabric or net for all ecosystems much like shipping containers provided a shared logistics infrastructure for international trade. This ecosystem's purpose is to provide shared contracting components, utility services and data services for all ecosystems to simplify their establishment and allow them to interact with each other.

Ecosystems can therefore contract with a network ecosystem to handle interactions with other ecosystems as a default. This prevents every ecosystem from having to contract with every other ecosystem. Instead, they only need to contract directly with another ecosystem when it results in an advantage not available in the common infrastructure.

The network ecosystem will also provide a global governance and dispute resolution process between ecosystems. This can be done across legal boundaries similar to how it is done in payment networks or via maritime law in trade today.

Finally, the common network ecosystem makes it an ideal place to provide shared computable contract components to the widest possible number of use cases. Of course this is only true when they don't need to be industry or culture specific. Components developed for the network ecosystem will therefore have the highest potential use, which should also mean that their royalties will be the lowest.

These economics should incentivize substantial development of components for the shared fabric or network, spreading the cost of development and maintenance over the largest number of transactions.

Contract templates

Writing custom computable contracts for each agreement between parties will be impractical in the vast majority of cases. Therefore, the Sweetbridge platform uses contract templates. Contract templates can allow anything from simple fill-in-the-blanks types of templates to AIs that create contracts using a robust set of answers to questions.

Regardless of the complexity, contract templates are parameterized contracts that use wizards to create the contract. These templates allow the parties to provide parameters, answer questions or select options. These parameters, answers and selections allow the computable contract to generate a written contract from a data model. The printed contract is not the computable contract. Instead, it is the user interface used by lawyers and courts. It expresses the computable contract in a way these parties can understand.

Templates are created by using computable contract components. The skill to create a new template varies depending on its complexity but in many cases is something a lawyer should be able to do with a limited amount of additional training. The user interface to populate the template can be automatically created by the platform without the need for programming.

However, in some cases a customized wizard may be needed or advisable. This is particularly true for templates designed for users who will not grasp the implications of some of the answers to questions without more information. Though the platform will generate a default wizard, a custom-built wizard can always be created by directly writing to the template parameter APIs. This enables very robust customization wizards built on AIs to be used when desired.

Summary

Computable contracts, particularly when paired with ecosystems, offer the possibility to significantly change the world we live in and how we engage in commerce. They provide an opportunity for us to revisit the original social contracts that underpin our economies. They will allow us to create a new form of legal entity and ecosystem that members can own. An ecosystem where the benefits from the value members create by participating are shared with the members instead of being extracted by third parties.

If done well these ecosystems offer the promise of a new level of economic freedom. Brought about by a new allegory for a city: a virtual city that has its own micro-economy. An economy where each participant shares in the ownership earned through co-operative participation.

The creation of this open source platform, the Intellectual property within the thousands of its components and the ecosystems themselves will take years and require many parties working together. This is a big change, but it is the next logical step (for technology) beyond social media and internet shopping. Incorporating or enveloping the digitization of law, regulation and the whole of commerce.

We are very aware that this is a grand vision. As founders, our goal is to be the spark that lights the fire of transformation in the economic sector. We are an ever expanding group of people and organizations who have a like-minded vision of building this platform as a gift. That’s why the platform will be fully open sourced and open to all.

We are creating a new way of doing commerce based on win/win instead of win/lose. Commerce where ecosystems compete but organizations cooperate. Ecosystems where firms can come and go, but value is preserved for the participants.

Why? Because, there is a better way.

Discover the power of together.

Sweetbridge.

Appendix A – Computable Contract Implementation

Computable contracts are generated by computable contract templates. These computable contracts no longer take configuration and can be used to execute transactions between the parties. Contracts can be printed but ideally are not signed on paper but using a Sweetbridge digital signature on their personal device. The printed form of the contract is the user interface to the computable contract for lawyers and court officials.

These compuntable contract templates are created from reusable contract components. The components are assembled via a simple Component Composition Language (CCL) to create these templates.

Common contract components

To reduce the work of creating computable contract templates the platform is being designed to support standardized legal components. These components can be bound together to create computable contracts without requiring the specialized skills needed to create the components themselves. Componentization allows for much greater diversity of agreements to be created without composing new computable contracts from scratch.

This is particularly important when linking these components with other areas of commerce such as accounting, security, settlement, supply chain events, data, etc. By creating reusable components that come pre-integrated or with pre-build interfaces to the things the computable contract must interact with, we substantially reduce the work while increasing the value of the components.

To facilitate this integration, common components must publish interfaces for data they require or provide.

Ecosystem components

Ecosystems will frequently need to create ecosystem customized components that are pre-integrated with ecosystem platform functionality, such as supply chain events. These components can be created from scratch, but most of the time can simply wrap standard network ecosystem components such as workflows, with integration to unique ecosystem events.

Because Synchronized Commerce provides a common platform for ecosystem participants, components only need to be written and then implemented at most once per ecosystem. For example, an ecosystem might contract with a set of logistics providers to provide logistics services to the ecosystem. These providers might provide shipment status messages to an ecosystem event monitoring service. A workflow component that monitors the shipment pickup and delivery for contracts with suppliers might be integrated with these events. Therefore, as long as these services are used, the events would trigger state changes in the shipment within workflow components.

By collectively utilizing a common set of logistics service providers, the ecosystem participants have no integration work they need to do to receive these benefits. For the logistics providers they obtain a collection of customers with a single operation, administration and information interface which lowers their cost. Both the ecosystem and the logistics service providers win and both can benefit from the lower cost.

Parameters

Parameters are formally defined input variables or data structures to components. Each component can accept zero or more parameters. Each component must provide the name, data type and a sample printed use of the parameters in the component.

Definitions are used to provide a printed definition of a parameter to a component. For example, a parameter seller might be defined in the print definition of a contract as “Gardener”. By defining the parameter as the definition the platform knows to use the word “Gardener” instead of the parameter name “seller”.

Component hierarchy

The following section outlines the various types of common components within the platform. Each component type can be combined with only certain other types of components. For example, State Components can only be applied to Workflow Components.

A hierarchy exists within categories of components in the Sweetbridge platform. This hierarchy is based on the natural hierarchies that exist within good contracting practice today. Therefore, each type of component can be applied only at specific levels of the hierarchy. A Contract Term that says something takes effect at a specific time and ends on a specific date can not be applied to a Contract Role component because it makes little sense.

The meaning of a Contract Term is that a Contract, SOW, Workflow or Clause only applies during a period of time. The meaning of a Role is to define the part a party is playing within a Contract, SOW, Workflow, or Clause. There is no valid need for Roles to change within a Contract Clause, parties can change but not the role the parties are playing. It would be very difficult to use language to write a single non-ambiguous clause with a role that changes in the middle of the clause. If done, it would imply some kind of change to the clause. It is less ambiguous and simpler to use two clauses that apply at different time periods.

The hierarchy structure can be thought of as looking like this:

- Master Services Agreement can have
    - 1-n Clauses
    - 2-n Roles
    - 0-n Contract
        - 1-n Clauses
        - 2-n Roles
        - 0-n SOW
            - 0-n Clauses
            - 0-n Conditions
            - 2-n Roles
            - 0-n Workflows
                - 0-n Conditions
                - 2-n States
                    - 0-n Clauses
                    - 0-n Conditions
                    - 1-n Events

        - Conditions
    - 0-n Rights | Obligations
        - 0-n Clauses
        - 0-n Conditions
        - 0-n Risks
            - 0-n Clauses
            - Conditions
            - 0-n Risk mitigations
                - Conditions
                - 0-n Clauses

Since components need to be able to be overridden under various circumstances, the Binding Language is designed to deal with precedence in mind. By default, the Sweetbridge platform uses the “specificity overrides generality” design pattern for order of precedence when it comes to components. This is the same pattern used in object oriented programming.

That means that the Term defined at the highest contract level of a master services agreement does not need to be redefined at each contract, SOW or Clause under the master services agreement. Components can have one or more attributes which can be defined as fixed, meaning they can’t be overridden. Fixing a component or a component attribute at a specific layer of the hierarchy means the component or attribute can’t be overridden by layers below that layer.

Therefore, if the master services Term uses a period based Term with a start date of Jan 1, 2020 and fixes that attribute, that date can’t be overridden by an SOW to have an earlier or later start. This could lead to a problem if an SOW was added to take effect 6 months later on July 1st, 2020. Therefore, special constraints can be supported by attributes of components that prevent a parameter from being outside of a range or domain.

In effect, the attribute value can be further constrained at a lower level of the hierarchy but can’t have a constraint broadened or replaced. This allows the master services agreement to have a start date in the Term component that is Jan 1, 2020 but a wizard on the contract template would not be able to set the SOW term start date to Dec 1, 2019 because the Master Service term is set to Jan 1, 2020. This prevents inconsistencies between subcomponents within a contract.

Component composition language

Components are bound together using a Component Composition Language called CCL. This composition language allows the output from one component to be used as input to another component. For example the output of the roles component might be a homeowner and contractor but a component in the template might take the parameters of buyer and seller. CCL also allows the components to learn what they should call the parameter in the printed version of a contract clause if not defined by a definition. This means that formal definitions are only needed to override this behavior.

Common component classes

Components are organized into classes. These classes describe the role or meaning of each instance of component class.

Definitions

Definitions are non-computable components that can be added to any other component. They allow parties to define terms used in print versions of the document. These generate clauses, providing clear and detailed explanations of key words and/or phrases used within and for the contract, generated as pronouns starting with capital letters e.g. Buyer or Seller.

They also serve as a translation to remap parameter names used in the computable components. A parameter named “Buyer” might be defined as the “Patient” in a medical contract. Definitions allow the default names used in components to be overridden and replaced with another name in the print version or template parameters.

Definitions are purely for the human users of a component, contract template or contract. The defined terms in a contract can therefore be programmatically generated using static analysis of the component parameters and their parameter definitions.

Intent

Intent components are special non-legal computable components that can be added to any other component. They allow the parties in a contract to describe the intent of the legal language and behavior of the component. Intent is used both in human based dispute resolution processes and component documentation, but does not affect the behavior of a contract or component.

Contract roles

Contract roles are the roles of parties in a contract. This component simply identifies the roles assumed by the parties. These must be formally defined as they are parameters to many components. As discussed above these can also be remapped to more industry or culturally specific terms in the print version of contracts.

Roles can be applied to Contract Templates, SOWs, Workflows, Rights, Obligations and Risk Mitigations. The default roles of buyer and seller can be used in most two party contacts when the print names can be overridden through definitions.

- **Counterparties:**
    - **Buyer** – The legal entity buying something in a transaction defined in a component or contract.
    - **Seller** – The legal entity selling something in a transaction defined in a component or contract.
    - **Escrow agent** – The escrow or trust agent to use on transactions defined in a component or contract.
- **Third-parties:**
    - **Inspector** – An inspector for something in a transaction that must verify or grade something defined in a component or contract.
    - **Insurer** – A provider of insurance defined in a component or contract.
    - **Logistics service provider** – A logistic service provider to be used at a milestone defined in a component or contract.
    - **Vendor** – A supplier of goods or services defined in a component or contract.
- **Financial parties:**

- **Bonding party** – The party providing a bond in the component or agreement.
- **Financing provider** – A financing provider to be used at a milestone defined in a component or contract.
- **Guarantor** – A party guaranteeing or warrenting something defined in a component or contract.
- **Settlement provider** – The party and legal jurisdiction of settlement activity defined in a component or contact.

Commencement clauses

When used at the contract level, these components create commencement provisions that introduce and identify the parties and their respective functions within the context of the given contract

Term

The effective term that governs when it starts and ends, how to determine if Clauses, Workflows, SOWs, Pricing, Contracts or Contract Templates apply. There are several predefined contract components for terms that are built into the platform:

- **Default** – a default term that says this contract applies whenever a transaction occurs between the parties.
- **Event** – this contract applies to all transactions related to a specific event such as a conference.
- **Project** – this contract applies to all transactions related to a specific project.
- **Transaction** – this contract applies to a single transaction.
- **Duration** – this contract applies to a date range that starts at a specific point and optionally ends on a specific date.
- **Periods after** – this contract applies from this date through x periods after the last transaction.
- **Combination** – a combination of inclusion and exclusion of the other terms using boolean logic (And, Or, & Not).

Scope

The jurisdictions, entities or business activities the contract or SOW covers. Scope can be applied to Clauses, Workflows, SOWs, Pricing, Contracts and Contract Templates. There are several predefined contract components for scope that are built into the platform:

- **Legal Location Description:**
  - **Address** – 1 to n physical street address locations to which this component applies.
  - **Assessor's parcel number** – 1 to n assessor’s parcel numbers.
  - **Surveyor’s description** – 1 to n surveyors descriptions.
- **Geo location:**
  - **Geographic areas** – 1 to n geographic areas, cities, counties, states, provinces, or administrative subdivisions for a role to which this agreement applies.

    - **Region** – A combination of inclusion and exclusion of Geographic areas using boolean logic (And, Or, & Not) for a role location. For example, “deliveries in New York State excluding New York City”.
    - **GPS box** – 1 to n GPS boxes for a role location to which this agreement applies
    - **Legal Jurisdictions** – 1 to n Legal Jurisdictions of a role to which this agreement applies.
- **Entities:**
    - **List of entities** – 1 to n entities in a role to which this agreement applies.
    - **Subsidiaries** – 1 to n subsidiaries of a role to which this agreement applies, such as a list of explicit subsidiaries of a holding company.
    - **Partners** – 1 to n join ventures, consortiums, SPV or other business partnership structures in a role to which this agreement applies.
- **Alliance:**
    - **API** – a customer, supplier, referral identified by a call to the “is name a”, “is location a” or “is source a” API for a role. For example, alliances are heavily used in internet referral programs such as click ad agreements.
    - **List** – a customer, supplier, referral identified by a look up in the “is name a” or “is location a” list for a role.

Workflows

Workflows define processes between parties, their states (or milestones) and determine when rights or obligations turn on or are discharged. All contractual relationships have some form of formal or informal workflow though sometimes it might be hard to see this when reading a contract. Also, contracts frequently have highly ambiguous or only partially defined workflows.

However, to be a computable contract the workflow must be articulated in a precise manner. Workflows break down into a set of possible states or milestones. Transition from one state to another is controlled by events. These events can be as simple as a statement by one party that we are now in x state, or it can be controlled by a sophisticated set of notifications from multiple sources about the status of something complex, such as a container being shipped around the world.

Even an informal workflow can be broken down into states that are stated to exist by one or more parties, if nothing else. For example, the work is “started” or the work is “done” may be nothing more than a party indicating that a state is currently true – i.e. “done”.

Since many state transitions and workflows are informal, the Sweetbridge platform provides a configuration driven App for human triggered events. This App allows the parties to define the state of a workflow from their personal device. This approach makes it easy to start using computable contracts and slowly migrate to IoT devices, satellites, ERP systems etc. that provide event information automatically. Even in highly automated environments it allows properly authorized humans to override the state to deal with cases where automation fails.

Workflows can be built out of existing workflows by adding or removing states and events in a similar fashion to subclassing in object oriented programming. Workflows can also trigger other

workflows. Over time Sweetbridge expects a very substantial library of useful mini-workflows will emerge.

States

In a computable contract workflow, states may be either informational or contractual. Contractual states always have rights with corresponding obligations for the counterparty. There are four types of states:

- Informational states used to communicate progress between parties, but triggers no change in rights/obligations but records a desire to know or share the information,
- Milestone states are special contractual states that represent settlement finality points, where one or more rights have ceased or obligations have been discharged,
- Initiation states are states that cause a right and obligation to come into existence, and
- Exception states that place the transaction into an exception condition

State changes are triggered by events that cause a state transition to a new state. One of the most valuable opportunities when moving to computable contracts is the ability to formally define “fail paths”. A fail-path is a state transition which handles a failure event by handling or raising an exception.

Exception states

In contracts today fail-paths most commonly trigger exception states. In most contracts an exception state is a dispute resolution process defined in the contract, but in practice this rarely occurs. The reason the contract rarely defines something that is actually done is that the legal dispute resolution process requires a lawsuit or arbitration. Lawsuits, arbitrations and mediations are high risk, costly and time consuming events which turn resolution over to relatively disinterested parties who have unaligned incentives. These incentives mainly have nothing to do with a fair resolution needed to maintain a good working relationship. Bad relationships are generally bad for business so neither party is quick to trigger the contracted dispute resolution process.

Therefore, most exception states are ignored or are handled in some informal negotiation between the parties. How these ignored exceptions are handled varies by culture, and even the same process may look different from one culture to another. The most common actual exception process is some form of the “since you did this, you owe me that” pattern of resolution. In this pattern, exception states are ignored by both parties for a while, then at some point one of the parties will trigger a tallying and valuing that will be done in some manner. This can occur at the end of a project, in a quarterly business review, ad hoc, or when one party needs a favor from another.

There is then a formal or informal process between the parties and in some cases this will involve another party to horse trade based on each party's list of the other parties' exceptions. These exceptions are always valued in some way and the parties reconcile or agree on a method of compensation.

In some cultures, such as Japan, this process is typically initiated by the party that owes the most as a way of saving face. In other cultures, such as the Middle East, this is a process that is typically initiated by the party that owes the other party the least when a favor is needed.

Regardless, the designing of computable contracts offers an opportunity to automate many of these fail-paths to prevent exceptions. Common dispute resolution processes like the “since you did this,

you owe me that" pattern can be automated so that the tallying is automated and the negotiation process is prescribed to occur on a periodic basis.

This enables the informal process to be turned into a formal one that matches the culture and actual process used today. By doing this we can turn a hidden unquantifiable relationship asset or liability into a measured relationship asset or liability. This asset does not have to be valued until the dispute resolution process is completed, but for the computational contract to work it must be handled. Otherwise, some level of the relationship dynamic is not controlled by the contract workflow and the actual rights and obligations are not known.

Acceleration clauses

Acceleration clauses are the ultimate fail path in many contracts and should be avoided wherever possible in computable contracts. The reason for this is that they make an obligation, such as debt, due immediately if a party defaults on other contractual obligations, such as payment obligations. These clauses are frequently used today in debt financing contracts to force a party into a negotiation or legal default on the contract. This means they always require human dispute resolution and so they cause an exit from the computable processes.

Therefore, these should be used carefully as they can create a systemic failure across multiple obligations. In master services agreements and Sweetbridge architected agreements these are typically avoided and more graduated failure paths that can be automated are recommended.

Default clauses

Default clauses are subcomponents or state components. These clauses typically define the state and events that trigger the ultimate fail-path within a workflow. They are used for breach of contract. They may also provide a sub workflow for remedies or defined rights of the non-defaulting party.

Events

Event components trigger state transitions and can be anything from one of several automatic events, like a date, to a business process event requiring rich data. Automatic events are things such as calendar events, timer events, or period based events – e.g. A payment being due 15 days from receipt of goods. Each business process event definition produces an APIs that can be called by an external process to trigger the event – e.g. provision of a weight certificate, laboratory grading and receiver per bale of seed cotton on an order.

Data provided on events may require data validation or may have data type restrictions. Missing, incomplete and invalid data being provided to events is quite common. Event component data validation and exception handling is one of the most important aspects to enabling computable contracts to function properly. Decades of experience has demonstrated that the best designs for event data error handling are to create state transitions and processes to deal with the most likely cases of data error on events.

The platform comes with several default transitions that are automatically added to every state that takes a business process event. The most important of these is a default exception process that always exists. This exception process allows the event to be called without raising a system exception that would stop execution. Instead data errors are returned in a data error structure as the return result of each call to an Event component API. This design prevents a malicious actor from passing data errors as a way of crashing a transaction workflow.

When an Event component data error is not handled by a state transition the platform provides a default state transition. This transition handles the exception by queuing the exception for human review and notifying interested parties. All data exceptions are logged and categorized by state, source, Event component, reason, and valuation rule.

Logging of Event component data errors enables analysis for recurring patterns in data errors to facilitate self healing exception handling and new state transitions based on error history. These enable the ability of the component developer to create data quality clauses that use data error statistics to drive economic incentives. These economic incentives can be used to incent high quality timely information and penalize undesirable data behaviors.

These error statistics also enable contracts that automatically police data quality for informational states that don't drive contract rights but do have economic effects on the parties. This can be a very powerful way of gradually improving data quality and timeliness. The reason this works is that the data provider has an ability to cost-justify improvements that result in higher quality and more timely information. It is also a great way to find the point of diminishing return on investments in data quality across providers.

Since computable contracts are dependent on Event components to change state and no event will have perfect information all of the time, the platform enables any Event component to be triggered manually. A website interface and personal device App are automatically configured when a new event is defined. These are automatically added to the contract interface for all computable contracts.

Every Event component API requires the caller to use an authenticated identity token. This token provides the identity of the individual, device or system calling the API. This means only an individual, device or system can trigger an event that changes state in a contract. Event components can also require n parties to verify the same event before the event will be triggered.

Multiple approval or a maker–checker model can be easily added to any event without the component developer doing anything. This is possible because all events support a "propose then approve" pattern. This pattern is controlled via each party's contract configuration and may be defined or changed at any time in the contract life. For example, an event for receiving goods could be triggered by a receiver in one instance of the contract while in another instance of the same contract a barcode scanner, a receiver and a warehouse supervisor must all approve the event before the event would be triggered.

Calendar events and deadlines

Calendar events are special platform level events that are used for things like deadline clauses. These events trigger deadlines or other special calendar clauses in contracts..

Timers and time clauses

Timer components use one or more events to set a relative point in time to trigger another event, such as payment. These are used to set out the time for production/supply of goods, discharge of obligations under the contract etc.

Time is of the essence clauses

These types of events set out deadlines and stipulations as to either calendar dates or timers and generate time is of the essence clauses.

Rights | Obligations

Contractually rights always create obligations for a counterparty. Rights come into existence at a specific state of a workflow, and are terminated at another state in the same workflow. Obligations are created for one or more counter parties, at the same time, by the same state as the right. Obligations continue to exist until they are discharged at the same time as the counterparty right is terminated.

Rights and obligations can either be transferable or non-transferable. All transferable rights and obligations have a registry that tracks the current owner of the right and the current owner of the obligation. These are tokenized and can be represented on a blockchain or DLT ledger if desirable.

Since the rights and obligations are in a register, this means other rights or obligations can be created off of these rights and obligations. Therefore, rights to be subdivided into other rights or new rights can be created on top of existing rights. The same can be done with obligations. This is one of the most important abilities of computable contracts and the Sweetbridge platform, because it allows the creation of derivative like instruments on assets, goods, and services.

This can unlock enormous hidden value that sits latent in these assets, goods and services today. The size of this partial value is potentially greater than the derivatives market today. The reason this is true is that there are more asset classes today that don't have derivatives than those that do, yet the total value of the derivatives market is enormous. This would provide the same abilities to companies as banks currently have to hedge risk and issue options.

For example, let's say Louis Vuitton wanted to create more revenue. It could sell a right to receive the first handbag produced from each new Louis Vuitton bag line for the next 5 years. This right would be very valuable, and would cost little to create but it would be most valuable if the right was tradable. Louis Vuitton could sell the right to get the first bag and make it transferable in a computable contract but they could not make the obligation transferable because only Louis Vuitton could fulfill the obligation - a Louis Vuitton bag.

States that terminate a right and discharge obligations are milestones. Milestones that provide settlement finality are more valuable than those that don't. This is because settlement finality eliminates the risk that the milestone can be undone.

Being able to know what rights and obligations exist on a transaction at any point in time provides one of the greatest benefits of computable contracting. The reason for this is that the risks of the obligation being fulfilled can be defined in near real-time.

Lien clauses

Lien clauses are a sub-component to a right component. Liens allow for, or exclude a right to retain possession of goods, as security or an interest in a good or asset, such as a lien on a property from a mortgage or a mechanics lien. These always work with a rights registry/ledger within the platform and are tokenized and transferable by default.

Retention of title clauses

These clauses allow a party (usually the seller) to retain its title (ownership) in goods which are being sold, i.e. the seller of goods protects itself against non-payment by the buyer by way of retaining ownership of the goods until payment is received from the buyer. These are used instead of workflow and rights components when no automated process is possible. These clauses are always a sub-component to a Rights component.

Waiver clauses

Waiver clauses are a sub-component of a right component that enables a voluntary and intentional surrendering of a right, benefit or privilege within a contract.

Risks

All rights carry risk, the counterparty risk that the counterparty won't fulfill their obligations at the very least. Today, risks are rarely defined in contracts. However, to access many of the benefits of computable contracts, risks must be identified explicitly. This is not a requirement but real time risk management is only possible when it is done.

There are many reasons to do real-time risk management:

1. To automate company risk analysis for accounting, risk management, business disruption and audit.
2. To enable lower cost insurance which uses real-time risk assessment to determine premium cost based on continuously assessed risk instead of periodic risk assessment.
3. To provide real-time assessment needed for counter parties such as financing entities, auditors, insurance companies, and supply chain partners.

Many risks are common across industries within standard business processes, commercial activity and for specific types of assets. The risk of theft for goods or fire for a building are examples. Each risk needs one or more methods of risk assessment.

Risk assessments

Each risk may have one or more risk assessment methods. These methods break down into three areas of risk assessment, Risk Level, Risk Window, and Risk Magnitude.

Risk level

A risk assessment consists of a means to estimate the level of risk. The Sweetbridge platform converts all risk levels into a rating between 0 to 5. No matter how risk assessment is done the risk must be reduced to a value between 0 and 5. Zero means there is no risk and 5 means the highest possible level of risk exists.

The practice of reducing risk into a single value between 0 and 5 is common in the insurance industry. Each type of risk needs at least one means for estimating the risk level. In property for example, this is commonly done via a risk survey. The risk survey may be done by satellite and or a building inspection by a risk surveyor.

Regardless of the method, a set of facts about the geographical location, a building's environs and the building itself are used by all insurance organizations to develop a property risk score. The facts used by each insurance company or risk auditor may vary and the score each underwriter would assign will also vary. The Sweetbridge platform stores all of these facts for each risk assessment so that different scoring methods can be supported off the same data.

Risk levels are separated into open data structures that can be used to support different risk level assessment processes. Different auditors or insurance companies can then create their own risk scoring processes that can be added to standardized risk components. This allows for both standardized and use-case specific risk levels to be supported by the platform.

Each risk assessment method’s data requirements produces an automatically generated API and user interface to collect or view the risk facts.

Risk window

Each risk assessment has a risk window which is the timeframe for the risk frequency and value settings. The size of the window should be limited to the time between possible max value events. For example, if a building fire destroyed a building and it would take a year to rebuild then the Risk window for the building fire risk should be one year.

Risk magnitude

Risk magnitude is the amount of risk within a risk window represented as a magnitude and standard deviation for the risk within the window in value terms. To calculate the standard deviation of risk each risk assessment component must have one or more ways to determine the value of a likely risk event. These are stored as three values, frequency, average value and max loss. These are used to develop a standard deviation for the risk. The “frequency” defines on average how many loss events occur within the risk window.

The “average” defines the value of the average loss per risk event expressed in a decimal value plus a currency qualifier. The “max loss” defines the highest possible loss for a single risk event and is defined as a decimal value and a qualifier.

Risk mitigation

Risk mitigations can be added only to risks. Each risk can have one or more required or optional risk mitigation methods. The reason for defining the risk mitigation within a contract is to express acceptable ways for mitigation of risk between parties. There are four methods for dealing with risk:

- **Avoidance** – Contract clauses that specify or require means of avoiding risks for both financial loss and damage,
- **Acceptance** – With some risks, the expenses involved in mitigating the risk is more than the cost of tolerating the risk. Parties in this situation can use clauses to disclose risk that another party acknowledges accepting or defines a method of monitoring,
- **Transference** – Contract clauses that define how risk is shared or transferred such as the use of insurance, hedging or other pooling techniques, and
- **Limitation** – Contract clauses that require a party to take some type of action to address a risk and regulate exposure. Risk limitation usually uses some risk acceptance and some risk avoidance clauses as well.

By defining risk mitigation in computable contracts the risk mitigation method can be monitored to determine if it is occurring. Alternatively, the risk mitigation method can be automatically performed. For example, insurance can be paid for by the computable contract out of payment processes so that it is always known to be in force.

Computable contracts that manage settlement with integrated accounting within the Sweetbridge platform can also provide new ways to manage many counterparty risks. A simple example of this would be party A owes party B and party B owes party C. Computable contracts tied to settlement through payment can use risk mitigation clauses that would settle party B's commitment to party C with party A's payment to B if party B does not pay their commitment to C.

The above example is a very simplistic but easy to understand example of what can be done. In a complex value chain within an ecosystem these could be extended to include thousands of parties and by linking ecosystems to ecosystems this could be extended even further. Though further study using economic models of this is needed several value chain segments studied in several industries show a significant reduction of counterparty risk in settlement.

This methodology can be further used for counterparty risk in obligations as well. By using computable contracts on real world assets, goods and services we can enable businesses to create derivatives. By trading them between partes in ecosystems, like financial institutions trade them today, we can hedge many counterparty risks using the same methods as banks.

Risk provision

Risk provision clauses are sub-components of risk mitigation components. They are used to regulate the transfer of risk between the contracting parties, or to or from a third party. Risk provisions may also include any related duties such as the requirement to insure goods. Risk provision clauses are a type of risk transference.

Indemnity clauses

Indemnity clauses are sub-components of risk mitigation components. They are used for clauses that set out agreement particulars for compensation or reimbursement by one party for the loss or liability suffered by another party. Indemnity clauses are a type of risk transference.

Conditions

Conditions can be added to all components except contracts, including other conditions. Conditions use logic on parameters to determine if a condition is true. Conditions can be used to determine if other components apply. A condition on an SOW for example can be used to determine if the SOW applies to a specific transaction.

A special type of condition can also be used inside of Clause Components to make the clause language sensitive to other clauses and configuration information in the contract hierarchy. This allows the legal language used in a clause to change when specific configuration values are set or other clauses are present in the contract. For example, if a Jurisdiction indicates the governing legal jurisdiction is in "England and Wales", in the UK the language can be expressed one way, but if it is the state of "New York" in the US it's language can be expressed entirely differently.

Contract configuration

The contract configuration component is a standardized component that is common to all platform components. Therefore, it is not a component someone creates as there is one and only one configuration component.

Contracts that are created from a contract template use contract configuration components to store the configuration parameters within the contract. This configuration is structured similarly to the settings App in a smart device. Each component can read other components configuration and can store its own configuration. The Sweetbridge platform is designed to store these values in an immutable form.

A standard application for reading, defining and updating component configuration when creating contracts is designed into the platform. Creators of components can build additional component or contract specific configuration tools but they must all store their configuration in the standard contract configuration component.

Each component that requires configuration must publish or use a defined configuration interface.

Configuration interfaces

The platform is designed to support standardized configuration interfaces that can be defined in a similar way to abstract classes in object oriented programming. These standardized interfaces operate at either a platform level or an ecosystem level to standardize the configuration of clauses. This enables clause writers to use standardized configuration information but specific clause language.

Dependency tree

Each component that takes configuration must publish a configuration definition and its component dependency tree. A component dependency graph works similar to dependency references in programming languages. It allows the author of one component to require inclusion of other components in any contract that uses the component.

This can be defined either as a specific component or a clause component of a specific classification that supports a specific interface. The reason this is a tree is that order of precedence is implied in hierarchy. Therefore, not only is a specific clause required but it must be enforced at a specific level of the component hierarchy.

Interface enumerations and lists

Many times configuration information is multiple choice or controlled by a list provided by some outside system. The configuration template supports both of these. An enumeration is simply a list of possible valid values. There are two types of enumerations, simple and contextual.

Simple enumerations are nothing more than a list of values and a description of what the value means. Contextual enumerations that are based on another value. For example, if the country is the “US” then the state/province list might be the list of states in the US. However, if the country is the UK the state/province list might be the list of nation’s that are part of the UK, such as England or Scotland.

Lists come from external API calls that follow a standard platform interface where the list must be provided by an external system. This is useful for lists that deal with products or services which are company specific. It also allows values to be verified by external systems such as addresses or company registration numbers in some countries. This could be dynamic or static with a periodic re-evaluation. Dynamic exhibits are common in contracting today which refer to documents that are not part of the contract itself.

Value formats and validations

Configuration information may have value formats or validation requirements to prevent errors in contracting. The valid format of an address, date, phone number or employment identification number are examples. For computable contracts to function, values in variables must be valid. This requires a level of rigger around variables that print-contracts never have.

For example, on a property insurance contract the address of the property must be the address of the insured location not the address of the owner of the property. Yet, a surprising number of these contracts are not filled out correctly. For computable contracts to automatically assess risk for example, the address must be correct. A user interface that forces the identification of the property to be insured via satellite image for example would go a long way to preventing this type of error.

Contract templates

Contract templates are similar to contract forms. In computable contracts these templates can be much more flexible than in a printed form. Templates define all of the components needed with their corresponding configuration requirements.

A computable contract can be generated from a template by configuring the template's configuration using the configuration application and publishing a contract. Additionally. templates can be used to create other templates with some configuration provided, but other configuration left to be defined.

Therefore, a template that takes one hundred configuration parameters could be used to create a new template that only requires four, the names and notice addresses of two parties.

Template wizards

Contract templates can be designed for use by individuals and businesses instead of lawyers. These less sophisticated users may need help in understanding the choices of parameters that templates offer such as various optional clauses. Therefore the platform is designed to support customized wizards that can be published with templates.

Pricing

Pricing components price or value the assets, goods, or services in transactions controlled by a computable contract. Pricing for goods or services on a contract can range from a single number, to a complex series of tables and algorithms, to a price quote via an external system.

Pricing components are built to handle specific forms of pricing or asset valuation.

Discounts

Discount components are a subset of pricing components that calculate discounts after pricing or within the pricing of items in a transaction.

Invoicing

The automation of invoicing in a computable contract enables the contract itself to sign the invoice. In a properly designed contract, this is valuable because it turns the invoice from a questionable quality asset, known as a receivable, into commercial paper.

This is known in banking as a verified invoice, a signed invoice or something similar depending on the country. A verified invoice means that the buyer has no disputes and will pay the invoice on around the date it is due. The credit risk on financing this type of invoice is the lower of the credit risk of the buyer or seller. This means the interest on this financing or factoring this type of invoice may be much lower for smaller enterprises.

This type of invoice is also used in many supply chain financing programs which allow less creditworthy trading partners to benefit from the financial strength of their counterparty. Supply chain financing and non-recourse receivables financing are typically only possible with some form of verified invoice process.

Cost allocation and account coding

There are three reasons a company may not pay an invoice on time that have nothing to do with a dispute:

1. They don’t want to pay because of balance sheet working capital considerations,
2. They can’t pay because they don’t have the funds to use on working capital, or
3. They can’t pay because they can’t book the invoice in the ERP system.

A study covering hundreds of millions of invoices from a wide variety of industries and countries conducted by Trax Technologies, Inc. found a surprising fact. The most common reason for a late payment by a major corporation is its inability to cost-allocate the invoice in the ERP system. You literally can’t enter an invoice in a modern ERP system without valid GL and cost coding being entered. If you don’t know what these are, you can’t enter the invoice, if you can’t enter the invoice you can’t make the payment.

Therefore, the platform is designed to support cost allocation components for invoicing. These range from simple fixed information to complex algorithms based on product lines, transaction specific information, business units or department structures.

Settlement

Settlement components enable computable contracts to use the platform to conduct payment and asset transfer settlement and reconciliation. When combined with invoices created by the computable contract significant value can be unlocked. The combination of invoicing digitally signed by the contract that governs the transactions allows both parties to know that the transaction is valid and will be paid on time.

When combined with ecosystem provided settlement finality, it is possible to create very low risk working capital solutions at an ecosystem level. Just as Ali Pay can use payment history on suppliers and place itself into a position to ensure payment because it handles all settlement, ecosystems can do the same thing. This can reduce working capital cost for the seller while allowing

the buyer longer payment terms, enabling working capital in receivables and inventory to become more balanced with working capital provided by payables.

Incentives

Computable contracts provide highly measured commerce activity. This is a side effect of their use. This enables ability to design fine grained incentive systems into contracts to provide economic incentives instead of simply legal incentives. Rewards and penalties can be designed to create a ROI that causes the counterparty to invest in process improvements. These process improvements lead to better and better performance until a diminishing return on investment is reached. This is very different from using legal penalties, or a breach of contract clauses, which have no upside economically and therefore actually motivate a behavior of doing as little as possible.

Incentive components can be used for everything from powerful loyalty systems that create shared economic value to incentives and penalties for performance. Economic incentives have historically proven significantly superior to contractual obligations. The ability to engineer economics into contracts that adaptively create win / win incentives may be the most valuable, and at the same time most undervalued, ability of computable contracts.

Take a reward system of something like an airline for example. Most major airlines create more enterprise value from their rewards systems than their operations. However, these systems are so poorly designed that they create major liabilities on the balance sheet even though they are worth more than the airline in many cases.

Using computable contracts, Sweetbridge won an award from SAP and Lufthansa[7] by showing how the airlines' point systems could be redesigned to convert a €1.8bn liability into a €1.5bn asset using computable contract royalty shares with zero change in cash flow or P&L. That is a €3.3bn increase in enterprise value.

To showcase this and drive investment in the platform, components and ecosystem creation, the platform has a powerful reward component for ecosystem master services contracts. This component uses the ability of computable contracts combined with settlement, identity and accounting to turn discounts, rebates, cashback and rewards into royalty shares. These royalty shares are tokenized, this provides liquidity in ecosystems that provide their own market making function to create this liquidity by buying back these royalty shares for cash.

These royalty share tokens are automatically given to parties as a reward for helping to grow ecosystems. Both buyers and sellers receive these incentives. Each of these is engineered to create more and more cashback to the owner as the ecosystem grows. This is accomplished by keeping the growth rate of new royalty shares below the growth rate of cash back from discounts, rebates, cashback and rewards.

This cashback comes in the form of restricted cash that can only be spent in the ecosystem. It is real cash, but can only be spent with a member of the ecosystem. This creates a powerful incentive to buy again from within the ecosystem but does not create a liability because it is fully funded from the discounts, rebates, cashback or other incentives already being provided by participants.

---

[7] https://newsroom.lufthansagroup.com/english/newsroom/lufthansa-and-sap-announce-winners-of--aviation-blockchain-challenge-/s/d0e6491f-d942-4cab-a3f4-3463b9a11719

By default, the value of these incentives is set at four times the annual cashback they provide, and liquidity is provided to buy these back from anyone by using some of the cashback to redeem the incentive. Sweetbridge is already doing this in the construction industry. Based on economic models we have projected that a single ecosystem member with $250m per year in revenue will create more than $100m in additional balance sheet value over a 5 year period.

Rebates

Rebate components monitor the economic use of a contract and provide rebates based on the level of economic activity, reaching specific thresholds within a period of time. For example, if the transaction value under contract is between $1M and $2M per quarter, a rebate of 5% on the quarter's activity will occur. However, if the economic activity is more than $2M per quarter the rebate will be 7.5%.

Rebate components can be a simple table of thresholds, as with the above example, or they can be complex based on a combination of activity by product lines, SKUs or services.

Data requirements

Data frequently needs to be passed between parties as part of events, or as a party of the transaction workflow. Data requirement components allow the parties to define what data needs to be provided, when it needs to be provided, the format of the data, and the acceptable transmission methods.

Data required

Data required components define the documents, data files, locations and other information requirements, including things such as data rooms or shared folders.

Data structure

Data structure components define the data structure's syntax and semantics for events, transaction APIs and transaction files. These components define the data structures and attributes that are mandatory, options or conditions.

Data structure components provide contract schedules that define file formats and data structures. These include data structures such as EDI, XML and CSV file structures that are to be used by the parties to exchange information on transactions governed by the contract.

In most cases these components should be designed to validate the data structure, syntax and attribute requirements of transactions as the data is received. This enables dispute resolution processes and incentive systems based on the data, and allows both parties to validate data before and after transmission.

Data transmission

Data transmission components define the data and file transmission requirements, processes including encryption, and transmission methods such as email attachments, HTTPS, FTP, etc.

Product, project and service requirements

These components define requirements between the parties. When certifications are required these components define what kind and who the valid entities are for the certification process. In some cases these processes may have their own SOWs or Workflows. These components enable parties to sign off on requirements as part of a contract workflow’s events.

The most simplistic of these is nothing more than a shared set of document files that are dynamically attached to the contract as they are generated and approved by the parties. These may be in machine readable and/or human readable formats as needed.

Product requirements

Product requirement components define product specifications including testing or certification requirements.

Project requirements

Project requirement components contain project documentation and specification information. These may be in machine readable or human readable formats depending on the need.

Service requirements

Service requirement components contain service requirements documentation and specification information.

Testimonium and signature blocks

Testimonium and signature block components are used to add electronic signatures to other components, contracts and contract templates. These can use Sweetbridge identity systems using personal smart devices, PDF signatures or paper signatures depending on the need.

Attestation provisions

Testimonium clauses and signature blocks that must be countersigned, witnessed or signed by a notary can also be supported. These create attestation provisions within a contract but can also be used in transactions to sign or confirm an event or document.

Component ownership

The intellectual property (IP) ownership of components is controlled by a licensing agreement which itself is a computable contract. This licensing agreement protects the IP of the authors of the component. New components and contract templates can be created from most existing components or templates.

However, when creating components or dependencies on other components the IP ownership of the individual component creators is maintained. Therefore a template created from an existing template has no IP because it only adds configuration values. Some things are not allowed to be owned on the platform because they do not represent creation of intellectual property that must be rewarded and maintained.

Configuration values, component interfaces, and templates derived from other templates aren't considered IP.

Component use business model

The platform is designed to reward component creators who build and maintain components by allowing them to charge a royalty for their component's use. These royalties are based on a point system that adds up to a total on a contract. Component authors participate in a fee all contracts charge on the transaction value they enable. This is possible because the Sweetbridge platform not only deals with the legal contract, but also the accounting and settlement of all transactions using these contracts.

Royalties are collected for 7 years from the publish date of any component version. This is done for two reasons, one to encourage constant maintenance of components and the second to limit value extraction from components that don't continue to have investment or are static. After the 7 year period components become open source.

Component writers can choose to mark their components as open sourced. Open sourced components do not share in royalties. As a result as more components are made open source the value that accrues to non-open sourced components increases. As this value increases, the benefits to the ecosystem from an alternative open-sourced component increases. The reason for this design is to focus new for profit IP creation on areas that have not been built versus areas that are well represented with lots of components.

Royalty calculations[8]

The royalties for a contract or template can be set by any combination of the ecosystem, component domain or author. The royalty can be changed at any time. However, increases will not affect any contract components that are already in use, only new contracts. Decreases will go into effect on existing contracts for new transactions.

Royalties are broken into two categories, author and points. Half the royalties are distributed pro-rata by component authors and the other half are distributed pro-rata by component points. Component points are meant to represent the level of the investment in the development of the component.

[8] This area of the platform needs further evaluation by behavioral economics and complex system theorists.

Component authors

All of the authors contributing components to a contract or contract template are counted. 50% of the contract royalties' shares are awarded pro-rata based on author. This is calculated by counting all of the components in a contract or template from each author and dividing that number by the total number of components. Therefore, if one author created 10 components and the contract was made up of 100 components they would get 10% of the author royalty shares or 5% of the total royalty shares for the contract.

The authors' shares are calculated when the template or contract is created not when transactions are executed. Therefore, when a component is added to a contract the component author gets credit even if that component is never used by a transaction covered by the contract.

Open sourced components do not receive royalty shares nor do they count in the component total. This creates an incentive for creators to contribute simple components to the open source base. By contributing to the open source base they increase the amount of royalty they receive from non-open sourced components that require significant investment. Over time this should result in a rich set of simple open source components.

The royalty agreement is a computable contract template itself that can always be overridden in a specific use case, like any other computable contract template. This allows the components to be shared with a predetermined way to use other parties IP without needing to negotiate or even contact the other party. This allows IP to be freely distributed and commercialized via a market place built into the platform much like smart device app stores.

Component points

Component points allow the creators of a component to indicate the investment required to create the component. Component points are integers in a range of -1 to 100,000. -1 indicates the component is open sourced, Zero indicates no royalty points but the IP is protected. Components with a value of zero are still counted in the calculation of author shares. Contracts and templates have no max points but all other components have a max points of 100,000. The value of a point is notionally equal to $1 of investment in the development of the component.

The investment in the development of the component is self reported and not audited. Therefore, some authors will overstate the value of their investment while others will undervalue it. However, because the value is published, a higher value will reduce others' interest in using an over valued component within their components or templates. Conversely a value that is too low may cause others to question its ability to be maintained over time, reducing their desire to use the component.

Since an open marketplace exists for components on the platform, multiple authors of similar components can exist. This should result in many overpriced or undervalued components self-correcting over time.

Compound components are built from other components and have their own point value. The sum of all sub components in a compound component can be more than 100,000. When this occurs there is a property on each component except a template or contract that allows one of three options:

1. **Prorate** – Component creators can indicate they will take a prorated value for use of their component without negotiations. The compound component value is set at 100,000 and all

sub components and the authors own points are added and prorated based on the total of 100,000. Therefore, if the sum of sub components was 100,000 and the author of the compound component set their points at 100,000, the total would be 200,000. This means that each point in the component would be set at 100,000/200,000 or ½ its value.

2. **Negotiate** – The component creator can indicate that they will negotiate point values. To negotiate, the authors publish an email address and phone number to contact for negotiation. Component creators can then sign a lower point value waiver which itself is a computable contract to set the new agreed points.
3. **Exclude** – This indicates that the creators are unwilling to let their component be used in a way that would reduce the prorated value of their points being reduced.

Contract royalties

Contract royalties are set by the contract creator, the ecosystem or domain governance process. These are limited by the platform to either a max amount, which can be any amount, a basis point value between 1-100 or the lower of both a fixed amount and a basis point value.

Therefore, the creators of a component might have their component used in contracts that would produce sufficient funds to provide their desired return on investment, or to cover their ability to maintain the component in the future. As a result, the platform is designed to allow component creators to set a minimum aggregate amount on an annual basis from any use. When this is not reached the contract creator can negotiate a custom royalty agreement with the component creator to continue to utilize the authors component.

Right to withhold

All component creators can choose to withhold their components from use by specific domains or ecosystems. This can be done either explicitly or reactively. Because the royalty amount is set on the contract or template by the author when published in domain or ecosystem, the component author has no way to force a specific royalty amount. To create an economic incentive for pricing to be set at a reasonable level, creators can withhold their components from a contract's use with 90 days of notice.

This is designed to create a tension between those that set the pricing and the authors of components that forces a win / win solution. To further incentivise ecosystems to properly price fees on contracts, components can't collect any funds at a transaction level that are not controlled by the transaction contract.

This allows the platform to publish the ecosystem fees and contract fees to all component authors. Since the authors know the amount the ecosystem is making off of the use of its contracts they can choose to withhold the right of use. Conversely to prevent authors from holding ecosystems hostage, contract or template creators can choose to change components to one from another author at any time.

This allows market discovery and value information to be symmetric. It means that pricing has to stay within market bounds or the party out of market will lose. Since there is no supply constraint on software it allows the creation of supply constraints to make a component scarce if pricing drops too low.

Tokenized

All royalty shares from contracts and components are tokenized so they can be transferred to new owners. This sets the model for tokenization of all assets, rights and obligations within the platform.

Component domains

Components can be published into domains or ecosystems. Domains allow a grouping method for components that are designed for specific use cases, industries, geographies, or as a means of organization. Domains can be uncontrolled or controlled.

Uncontrolled domains don't have any validation process. Controlled domains have a formal review process that a component must pass to be rated and go into production. Controlled domains allow a review process to be established under some governance body to manage a minimum level of verification or peer review before publication. These bodies can also set the royalty amount for the contract.

Ecosystem production domains would typically be controlled where test or development domains would typically not be controlled. Any party can also set up their own domain for development.

Testing and modeling

Computable contracts and components are software and software needs testing. The platform is designed to enable both testing and modeling of scenarios.

Testing

Before components can be published in a production domain they must include test scripts that can be used to verify all paths of code execution and all exceptions. These test scripts are used by validation processes to make sure that components are properly verified.

Modeling

This means that scripts can be created to model what happens under a contract based on specific events or conditions. This is a very useful feature of computable contracts and components because what-if scenarios can clearly show parties what would happen if x or y occurred.

Component certification

Today, contacts are reviewed by multiple lawyers whenever parties create a contract and have independent legal representation. Computable contract components must build on this practice and potentially include other disciplines such as technical reviews, accounting, ethics, etc.

Therefore, domains and ecosystems can require production components and contract templates to be certified. This can be done however the domain or ecosystem governance body desires. These certification processes can then be used to set a rating on components or templates that pass minimum requirements. This body would also police components to make sure that authors were not simply plagiarizing components that are created by another author.

For example, an ecosystem might require that accounting components be audited by a financial audit firm to make sure accounting treatments were valid.

Component rating

Component rating can be user feedback based, formal rating by the governance body, or both depending on the governance body or domain decision. Ratings allow peer review processes to provide star ratings based on 5 stars. Formal ratings can use certification icons to represent passing of the component by one or more certification processes.

Information ratings also support reviews with free form text comments. All ratings require a platform identity and identity of the party providing the rating to prevent scam and fake ratings.

Clauses

Clauses add the specific legal language for a wide range of legal requirements. Clauses are organized into classifications that relate to the purpose and semantic of the clause. Clauses are classified as extensible lists of:

- Categories – see Clauses section below for the major categories,
- Types,
- Groups, and
- Sets.

Clauses support translation of the primary language into multiple translations to support internationalization.

Enforceability clauses

Clauses that are specific to a legal jurisdiction which control the type of law and legal jurisdictions for dispute within a court. There are currently three types of enforcement clauses:

- Governing law,
- Jurisdiction, and
- Statute of limitations.

Dispute resolution clauses

The dispute resolution clauses define the type, method, and party to use for dispute resolution between the parties that must be used before seeking the participation of a court. There are four types of dispute resolution clauses:

- Mediation,
- Arbitration,
- Ecosystem Resolution, and
- Automatic.

Liability clauses

Liability clauses control how liability between the parties and on transactions is handled. Who is responsible for what under what conditions. There are currently four types of liability clauses:

- Limited liability,
- Third party rights,
- Specific Liability (Product, etc.), and
- Joint and several liability (Clauses that define and provide for the total liability either collectively or individually of the parties if damages or losses occur.).

Capacity

The party signing has the right and mental capacity to make the agreement. They may attest to the fact that they are authorized to execute the agreement on behalf of a legal entity or another party. There are currently nine types of capacity clauses:

- Power of Attorney,
- Trusts,
- Mental Capacity,
- Real Estate Trusts,
- Corporations,
- Consortiums,
- Charities,
- Government, and
- Ecosystem.

General clauses computable

Miscellaneous clauses which have computable functionality.

General clauses non-computable

Miscellaneous clauses which do not have computable functionality and are only textual.

Force majeure

Clauses that represent unforeseeable circumstances that prevent someone from fulfilling a contract or clause. There are currently two types of force majeure clauses:

- Acts of God, and
- Specific definitions.

Data privacy clauses

Components for data privacy and non disclosure. There are currently three types of data privacy clauses:

- Non-disclosure,
- Individual data privacy, and
- Organizational data privacy.

Data access clauses

Components that define data access and sharing rules. There are currently two types of data access clauses:

- Data access policy, and
- Data sharing.

IP ownership clauses

Components that define intellectual property.

Severance clauses

If any of the clauses become illegal or unenforceable what actions should be taken such as that clause is no longer in effect but the contract is still valid.

Setoff clauses

Clauses that govern rights of set off on claims or damages. There are currently five types of setoff clauses:

- Rights to setoff,
- Contractual setoff,
- Legal setoff,
- Insolvency setoff (organization), and
- Bankruptcy setoff (individual).

Damages clauses

This is a very large area of contracting with hundreds of specific types of damages. There are however six main types of damages:

- Compensatory damages,
- Exemplary/Punitive damages,
- Nominal damages,
- Incidental damages,
- Consequential damages, and
- Liquidated damages.

Authentic version clauses

These clauses detail the language to be used in the event of a conflict/dispute.

Entire agreement clauses

Clauses used to state that the given contract contains the entire agreement between the parties and that there are no additional or supplementary agreements, warranties or representations. This does

not mean that the contract can not be adaptive or modified, only that it must be approved by both parties when changed. When changes are executed a new agreement can become the entire new agreement and depending on design, may or may not constrain any prior agreements.

Exemption clauses

Exemption clauses used to set out any exclusions, or limitations of any duty or liability. Exemption clauses are also part of condition components to express condition logic in written form. Exemption clauses can be used without conditions when they need to rely on human versus computable logic.

Retention of title clauses

These clauses allow a party (usually the seller) to retain its title (ownership) in goods which are being sold, i.e. the seller of goods protects itself against non-payment by the buyer by way of retaining ownership of the goods until payment is received from the buyer. These are used instead of workflow and rights components when no automated process is possible. When a computable solution is possible Rights components should be used with states instead.

Best endeavours clauses

Best endeavors provisions for a party stating it does not warranty a particular matter but undertakes to attempt to achieve such a result that is defined and agreed in an agreement.

Recitals provision

Clauses that provide statements of agreed or understood facts material to the contract.

Reps and warranty clauses

These clauses provide a written undertaking as to the correctness of certain claimed facts material to the contract.

Assignment of rights

The ability to assign rights and obligations is a property of the right or obligation at a specific state. The general assignment language is defined at each right or obligation and can be turned off and on at each state.

Insurance templates

Insurance templates are special contract templates for use in risk mitigation components that transfer risk using insurance. These templates allow standardized ecosystem level insurance coverage to be baked into contracts. This prevents each party from having to provide their own insurance and enables the ecosystem to provide insurance at wholesale instead of retail cost.

Dynamic vs static contracts

Today all contracts are static, meaning once executed that contract does not evolve over time. Computable contracts can actually be designed to adapt over time as laws change or as activity between the parties evolves.