SeqCarbon - Architecture

V1.1, 2025-06-30, Lorand Kedves

1 Executive Summary

The context:

- Increasing awareness of the importance to make all operations of the human civilisation sustainable.
- Agriculture is our global, critical, direct and most significant interaction with nature.
- Science shows that changes in agricultural methods can not only significantly reduce our emissions but also remove CO2 from the atmosphere and put back into the soil from where it was removed by the industrial methods.
  (https://youtu.be/nZMJjTkbhYc?t=755)
- The example of the carbon credit system shows that without fully transparent data management; fraudulent actors can distort any benevolent projects and general trust.

The SeqCarbon project focuses on this data management with both strategic and short-term goals:

1. Understand the motivations of the different players, from the small farmers to multinational food production companies, national / global regulators (official ESG reporting) and experts.
2. Design a fundamental architecture for a reliable, auditable core data collection from all agricultural activities, and transparent indicator calculation / transfer both along the supply network and towards external entities.
3. Build a “proof-of-concept” prototype, a partial implementation of the architecture that fulfils an actual need in a way that the related parties can accept. That is, calculate selected indicators (like CO2 equivalent emission) for a product delivered to a buyer by combining the action data collected at the farm with emission factors assigned to the used fuel, fertiliser, etc.

This document first describes the architecture (2) as the result of the analysis (1), including the overall description of the long-term working solution (technical requirements and business model) that allows all players verify if they find it feasible and would rely on the system.

Then it details the description of the prototype: the necessary components to be implemented, the conditions for validating the operation, and the necessary compromises.

The goal is positive feedback from selected players both on the architecture / long-term vision and on the prototype demonstration; and understand the resources needed for a production-level implementation. This would allow setting up a proper business plan with realistic budget and seek for funding.

2 Motivations

In very dense technical terms, the aspects that directly influence the system architecture.

2.1 Farmers

- Do not want to deal with anything they do not own (e.g. emission calculation).
- Want to decrease the already too much administrative burden: if something can be generated from the collected data, it should be.

2.2 Buyers

- Want to demonstrate emission reduction, already optimised Scope 1 & 2; Scope 3 gives the best ROI, but they have no control over it.
- Do not want to investigate the sources or take the responsibility; currently use external official aggregated databases. Our system must offer proper data protection, reliability and transparency to be accepted.
- May want additional information (not only central ESRS/VSME/? but SAI, LEAF, etc.) on the product with minimal (preferably zero) extra effort.

2.3 EU / Government

- They want control over the content of the report – XBRL offers that both with format and content via taxonomies.
- Bad experience with carbon credit and alike – they require transparency, auditability, reliability.

2.4 Expert Groups

- Need high level granularity and control over the reported data (probably more flexible than XBRL taxonomies would allow).
- They would like to build models and protocols, want to test and verify them – again with minimal effort, high ROI.
- Field measurements are expensive; they would like to combine them with model-based need verification.

2.5 Strategic Goal: real CO2 Reduction

- Agriculture offers a viable method to reduce atmospheric CO2
    - The energy source is the ultimate solar plant: the global agriculture
    - The target is the total volume of arable land from where carbon has been extracted with current industrial methods.
- Measurement is complicated, the change is on the edge of error margins and happen over years, granular data collection and flexible analysis is critical.
- The target methods are labour-intensive, therefore less financially profitable, require local collaboration instead of global competition. On the other hand, they are much better in all social aspects (local communities, governments).
- Requires support from government or big industrial players, but the trust has been eroded by previous initiatives.

3 Prototype

The goal of this project is to create a prototype that

- Demonstrates the data collection capabilities described in the requirement specification and interviews.
- Provides sustainability indicators that the buyers would agree to accept from the production system.

3.1 Data Scenario

A realistic sample of the environment with the following Entities:

- Lab Merchant is a grain merchant Entity that
  - Buys grains from farmers.
  - Stores and manage the goods on it site (store in silos and warehouses, transport among them, merge and split, dry, etc. the Products) while the system collects the sustainability footprint of those actions.
  - Sells to buyers.
- A few (3-10) "typical Lab Farms" with some Parcels, Plantations that produce grains with realistic field Job activity. We calculate the Footprints locally at each LabFarm until they sell the product to the Lab Merchant and transfer the aggregated Footprints. That is Scope 3 reporting over XBRL.
- Two Lab Processors who buy the product from the Lab Merchant and can look up the collected footprint data, then peek into how it was calculated. Processors
  - Get the data they need to integrate to their reporting (primary goal).
  - Can verify it through the system to the original actions (auditability).
  - Can only see Products they bought, not others (data protection).

3.2 Milestones

1. **Static data with reporting services** – "minimal viable product", **Static Demo**
   - LabFarm1, LabMerchant1 and LabProcessor1 are populated with core and calculated data.
   - Users can log in and read all data on the portal.
   - The system generates the Footprint reports for the products sold to the buyers, allows history analysis
   - All interfaces provide proper, transparent access control.
2. **Backend Job management** – "minimal lovable product", **Dynamic Demo**
   - LabFarm2 data and the list of jobs imported from a simple Excel workbook, LabMerchant2 and LabProcessor2 have pre-configured data and Jobs.
   - The users execute the Jobs that create and manage the Products.
   - The system aggregates used resources and calculates the footprints using the Protocol (extended info tables, calculation scripts, output templates).
   - The rest is the same as Milestone 1.
3. **Complete system demo** – ***(not realistic within the deadline)***
   - Users can manage the static data on their property sheets directly (Farm, Parcel, Assets, etc.), create and edit planned Jobs before executing them.
   - The rest is the same as Milestone 2.

Details

1 Disclaimer

The following parts aim to cover the “big picture”, a future global data platform for agricultural data management, so that

- long-term strategic goals and their proper representation can be verified,
- production implementation roadmap and resource planning can be started,
- the probable motivations of different business actors can be kept under control, not destroying the potentials in this system.

The document is a technical summary with enough details to drive building the production version, updated during the prototype implementation. The “Business Model” is an initial step to start a more detailed conversation.

Regardless, this document should be used as a technical background for public materials about the SeqCarbon project.

The prototype system is a proof-of-concept implementation of ***some core modules*** and demonstrates a ***selected small subset*** of the required features on a simulated “Lab Scenario”.

Core Meta Item Definitions

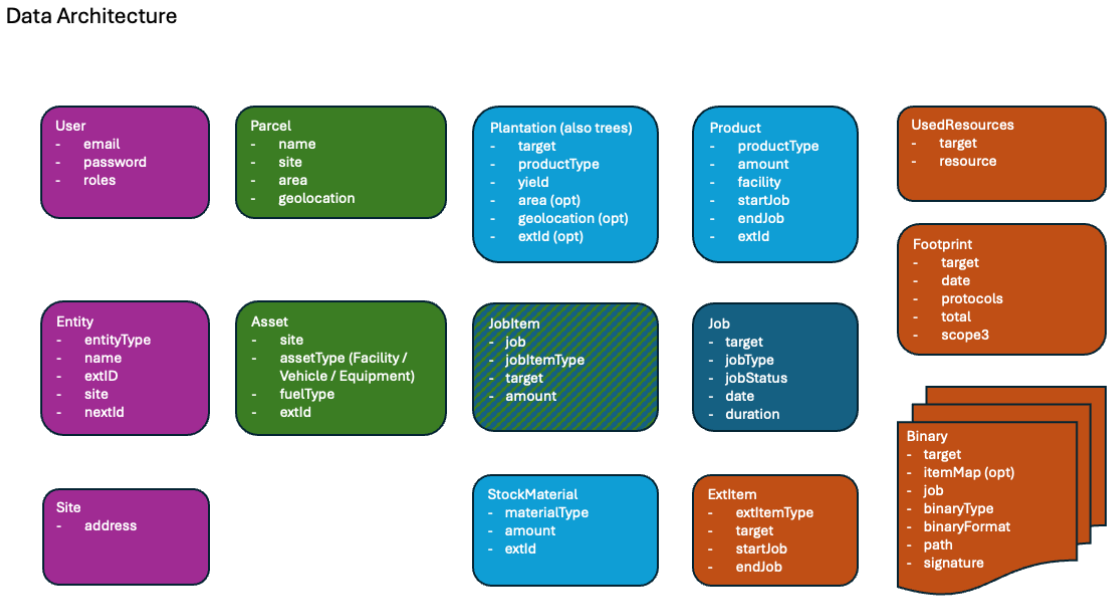

Figure 1 from SeqCarbon_Analysis_20250630.pptx

4 Recommended Business Model

Traditional profit-oriented competitive approach does not work well in this environment; this is demonstrated by the 20+ years in finance. The solution is a layered approach.

The **platform maintenance** requires a non-profit, non-governmental organisation in global monopole position (this is like XBRL International combined with the Arelle / Workiva group). Its sole purpose is to build, maintain and provide free, controlled access to the platform and the fundamental Protocols to all participants, governments, universities, analysts, etc. Just like many other IT foundations, can get revenue from:

- Limited license fees from contracted for-profit national service providers.
- Supporting for-profit entities, academic research, etc. on contracts.
- Donations, grants.

National service providers are profit-oriented, but government connected companies (like the OAMs for ESEF reporting in the EU), one for each country that:

- Work together with the government to integrate the local legal framework.
- Sell the service to clients (farmers, merchants, processors, etc.)
- Maintain the IT infrastructure.
- Manage the Protocol set applicable for that country.
- Provide controlled access to researchers, analysts.

Any number of **Protocol creators** focusing on target areas like SAI or LEAF could provide Protocols in the system. The fact that the Protocol is "open source" and can be used by anyone is not a problem, but a benefit.

1. Any Entity can test their own data against the Protocol and check if they meet the conditions with zero cost.
2. Those who are compliant or think that the remaining issues are worth the effort or need to comply for external requirements, are motivated to buy the license and optionally, contact the Protocol owner for consultancy.
3. The Protocol compliance is valid only if the Entity have a valid licence to use it; in that case the compliance automatically appears in the system, the platform ensures its credibility, continuously monitors the conditions. *Like a blockchain smart contract, but with full transparency and zero extra resource needs.*
4. Neither the Entities, nor the Protocol owners spend any money and time on marketing, research, trials, etc., just use the basic features of the platform.

This only works with a global platform. It's like the railway network: there is one standard and one network of railroads, maintained by one entity. The service is the same across the national borders, not affected by the competition among the business entities who *use* the rails. With the open platform,

- The maintenance company is only motivated to provide the required features.
- The service providers benefit from the Entity registration fees and depend on their positive feedback.
- Protocol creators can compete for the global audience directly, and benefit from the user license fees.

5 Core Data Collection

5.1 Fundamental Considerations

- The data model is flexible, data structures, possible values, etc. will change over time but the system must be able to handle historic data with obsolete structures and procedures. The system uses a configurable meta description, stores all versions, can recall the related definition for any data.
- The system allows distributed operation because large players may want to run the system on their site for security / integration. All entities must register at the centre to get their unique "SeqCarbon entity ID", item identifiers start with this ID, followed by the item type and a numeric ID unique on Entity level. This solution does not need global locks and still ensure unique identification.
- Separate the known, "tabular" fields for each type, store additional information to any item in "ExtInfo" records that have a JSON "payload". The meta definition is part of the protocol; this allows the maximum flexibility with reasonable performance.
- The transactional data of the system will not grow over time: Jobs and their contexts can be archived when the Product is sold; the performance does not degrade over time. The archived part can be grouped by year and managed as historic data.
- The system never deletes records, only flag them obsolete, can use an ExtInfo item to store more details.
- The active data content is relatively small, larger items (imported data, images proving a fact, etc.) can be handled separately and not used directly.
- It is likely that the active and historic data should be stored differently. The system should keep active data in the simplest form of CSV / JSON / XML files to avoid dependency on external tools like a database (platform independence, zero trust). Cumulative historic data in the centre should use a relational database for easier backup and advanced indexing.

5.2 Organisational Data

The goal is to ensure data and service access control, not to fully represent the actual company hierarchy, or to support historic analysis (does not record role changes and actions by users).

5.2.1 Entity

The entity is a uniquely identified registered member of the SeqCarbon system. This is an abstract item with the following attributes:

- *entityType*: farm / merchant / processor, etc.
- *name*: a user-friendly text.
- *extId*: the ID provider type and value (1-n) that allows external identification.
- *site*: the default physical location (exactly one from the registered Sites).
- *nextId*: internally managed number to provide the last part of a new item ID.

Possible ExtInfo types:

- *Contact*: contact information of the Entity (1-n)
- *Role*: role definition and assignment (0-n). If no role given, all users have full control. The Role attributes contain the list of users having it.

5.2.2 Site

The Site is a physical place that belongs to an Entity with a unique *address*. Used to keep record of internal transportation and optionally local emission calculations, inventory management. (Not used in the prototype.)

5.2.3 User

People who can use the system to work on behalf of an Entity after logging in with a registered *username* and *password* and optionally the list of the assigned *roles*. The User must provide an *email* address used for validation and communication. Federated login (Google or Apple ID) and multi-factor authentication may be added later.

When creating an Entity, one admin User must be created immediately. New Users must select the Entity they belong to, and the admins must accept the registration.

Users can log in to the system by providing their credentials, work on their Entity limited by the Roles they have in the Entity.

5.3 Entity Properties

These items represent anything that the entity "owns": currently Parcels and Assets. The data on this level control the actual operation, directly managed and not logged, and footprint recalculation cannot use them. All attributes that are related to activities are extracted to ExtInfo items that have a validity period; therefore, their actual attributes can be recalled for any given time (calculation validation).

5.3.1 Parcel

Parcel is an area on a map that the farmer uses to grow plants on, it stores its locally unique *name*, fixed *area* and *geolocation,* and the *site* it belongs to. Entities can have multiple Sites with silos, garages, offices, etc.; and operate Parcels from different Sites.

The other end of the scale is a greenhouse as a Parcel, with a few lines for each vegetable, and the arrangement can change in each season. Some of the jobs are related to the whole Parcel, others only to the lines with tomatoes. The solution is that the Parcel is final and covers the whole area, but the Plantation *can* have area and geolocation as well. The Parcel does not change, but the Plantation ends with the season and the new season allows a different arrangement, within the same Parcel. When the parcels are rearranged, that also means making the current Parcels obsolete and create the new ones. Location-based analyses will use the geolocation to find the related Parcels.

Jobs are physically done on a Parcel but generally belong to the Plantation, the used resources are aggregated to the Plantation and then the Product. However, in some cases this is not true like fuel consumption of fixing the irrigation system – that adds to the footprint of the Parcel.

5.3.2 Asset

The Asset is anything that can be used to accomplish a Job:

- a *Facility* like a silo or drier or a conveyor belt,
- a *Vehicle* that uses fuel or electricity to transport product or passive equipment,
- an *Equipment* like a plough or a sprayer.

The list of possible assets is configurable in the main Protocol to ensure that all Jobs that may have sustainability footprint can be covered by the available *assetTypes*.

Assets have *extId* (like a vehicle plate number for identification) and *fuelType* attributes. Although *fuelType* may change (start using biofuel instead of regular), the UsedResource is updated on Job execution based on the current value and is never recalculated.

5.4 Active Items

These items represent the primary agricultural activities. These items are never edited manually, only the scripts assigned to Job execution can change them to ensure transparency and reliability.

5.4.1 StockMaterial

StockMaterial is everything that is consumed while accomplishing a Job like fertilisers, pesticides, water, etc. The general idea is that the farmer acquires these materials, store some amount for a while, then applies them during a job. This is not an inventory management, *the system only handles the total amount of one type*, as one pool mixing all batches the farmer bought.

The footprint of any stock material consists of two factors:

- Industrially produced materials should have a unique type identifier; for that ID some organisation or the producer provides sustainability indicator values by the amount (tons, litres, etc.). The system's protocol contains these tables, never edited by the farmer.
- All stock materials can have Scope 3 emission values from the transportation and storage of that specific batch from the factory to the farmer's Site.
- When the farmer fills the water reservoir from a local well, the energy used by the pump adds to the total UsedResources "content" of the water.

The StockMaterial record stores the actual amount all the time, keeps the history of all stock changes in an auditable way. The amount can change by

- Executing a "Buy" Job – the cumulated Scope 3 measures are also updated using the incoming batch.
- When part of the StockMaterial is used by a field Job, the amount is decreased – the Scope 3 values are split between the remaining stock and the UsedResources / Footprint of the job.
- It is possible that an amount of the material is lost (spill, evaporation). In this case the amount is decreased but the UsedResources / Footprint remains the same and will be distributed over the jobs.

5.4.2 Plantation

Plantation represents plants growing in a Parcel but handles multiple scenarios.

- For most cases, this is seasonal and repeated over years (preparation, sowing, treatments, harvest); but Plantation can also be a vineyard or fruit trees with much longer life cycle.
- Farmers can use cover crops in parallel with the main one to retain water or protect the main product, so multiple Plantations can exist in the same Parcel.
- Farmers can grow different plants on the same Parcel like in a greenhouse; for this reason, the Plantation can have their own *geolocation* and *area*. In this case, the UsedResources of one Job can be distributed over multiple Plantations in the same Parcel, the ratio is stored in the Job.

Before seeding, Jobs can be done on the Parcel, some of them related to the Parcel, others are preparation of the Plantation. When creating a Plantation on a Parcel, the farmer should review the list of the previous Parcel Jobs and assign Plantation-related activities to it. Note, this does not have to be all, e.g. they start with seeding the cover crop and then the main Plantation, which should inherit the preparation resources.

This also allows partial usage of a greenhouse: half of the area is allocated for an early vegetable and start working with it, later the other half joins. All Jobs on the parcel after harvest belong to the next Plantation (the UsedResources list is not updated after harvest).

5.4.3 Product

The Product represents a given, fixed *amount* of a single *productType* of grains, apples, etc. that is handled together.

Product is created by a “Harvest”, “Buy”, “Merge” or “Split” Job, this is stored as the *startJob* of the Product. Its source may be one (or more for Merge) other Product of given amount; the sum of those must be the current amount. The Product can be found at a *facility* Asset (like a silo, warehouse, drier, etc.). This location can be changed with a Transport Job. Jobs can be assigned to the Product (transport, dry, wash, etc.) and the UsedResources are aggregated to the total amount.

When the Product has no other jobs assigned, then the Merge can continue, registering the additional amounts of Products. Splitting a Merge without other Jobs result two Merge jobs with the proper ratios. If a Product has other Jobs assigned, Merge and Split closes it or them (stored as *endJob*) and creates new Products with the respective amounts.

At the final step, the Product is sold and leaves the Entity. At this moment, its whole creation path is closed, and the final Footprint calculation is executed, the Scope 3 report is created according to the current Protocol. The buyer can be a registered Entity, in that case the access control is available, or can register at a later point to gain direct access to the product data, but the calculated Footprint is public information.

Secondary Products (like straw, branches, etc.) also appear in the process. They may also be sold to other farmers or become StockMaterial and processed the same way.

5.4.4 Using *extId* in Active Items

Any “thing” that is important to be uniquely identified, the type separates them that comes with the related payload definition. Some possible types are:

- Tree: The plantation can contain 100 apple trees, and the yield is 10 tons of apples from all trees. However, if some get infected and cured, it is done individually, so they can become separate items (can even handle their yields separately). Similarly, some trees can be replaced, the new ones don’t have yield for a few years.
- Any individually identified animal (a cow vs 100 chicken) – yield separation for the same reason.
- Product sample, taken at a certain moment either for local examination or to be stored in a vault for reference; or to be sent to the buyer for their analysis.

5.5 Actions

These items register all data related to individual activities.

5.5.1 Job

The Job represents the activity with simple generic

- *jobType* controls the content of the “payload” of the item and selects the execution algorithm.
- *date* is mandatory, can be in the future, *duration* is optional.
- *jobStatus* is initially Planned and the attributes can be edited. Later it can be manually removed or executed, in the latter case the assigned logic performs the Active Item updates and the attributes become read-only.
- *target* is the primary item affected by the Job, like a Parcel, a Product, a Plantation.

If a Job refers to multiple Active Items (buying multiple StockMaterial items, Merge or Split Products, etc.), those items with all details (like amounts) appear as JobItem instances linked to the Job.

Transferring a product to another Entity within the system is done by a Job pair.

- The “Sell” Job at the seller Entity closes the product, calculates the Footprint, and allows the buyer Entity to reach out for the history of this Product. Therefore, data access is controlled on request instead of passing data over to the target Entity where this control is lost. The depth of this access (like, can the buyer see individual jobs or only aggregated results) can be configured. Of course, if the buyer is not a registered Entity, there is no “Buy” pair.
- The “Buy” job is created at the buyer Entity and must be executed to create a local replica of the Product or update the StockMaterial list when trading byproducts. Buy job can exist without a Sell pair if the Product or StockMaterial comes from outside the system.

5.5.2 JobItem

0-n items connected to a Job with a simple data content:

- *job* refers to the Job this JobItem belongs to.
- *jobItemType* informs the algorithm how to use this item.
- *target* is the related item, a Product for a Merge, a vehicle for a Cultivation Job.
- *amount* is optional, like the amount of grains bought or pesticides sprayed.

The data in JobItems are used by the Job algorithm, e.g. when the StockMaterial is applied, the related Scope 3 Footprint is updated; the part that belongs to the applied portion is added to the target Footprint. This is a final operation and there is no way to repeat – the Job is executed once.

5.6 Footprint-related Items

The reason for the data collection is to calculate footprint information in a transparent, reliable way. There are multiple footprint related items, they all use a JSON payload to store all data, and they always belong to one Parcel, Plantation or Product identified by the *target* attribute.

5.6.1 UsedResources

Aggregates all amounts of used resources for the target item, like fuel consumption by type, electricity, StockMaterial, etc. This item is updated on Job execution: e.g. a JobItem contains 220 litres of fuel consumption, the Asset contains the Diesel fuel type, so the 220l of Diesel fuel is added to the existing consumption (that may also contain 95l of Gasoline and 130l of Diesel from previous activities like cultivation, harvesting, transportation, etc.)

Important to note that UsedResources are inherited from Parcel to Plantation to Product, so adding them up is not a valid operation. For example, this allows aggregating the fuel consumption for Parcels over years and find that creating a new Site with some tractors closer to some Parcels would be worth the investment.

UsedResources contain tangible direct measurements, are final by nature. The live system should however allow correction items to fix data input errors.

5.6.2 Scope3Footprint

Scope3Footprint is inherited information from the seller, belongs to a Product or StockMaterial, contains calculated emission values (CO2, N2O, etc.). This footprint is final, changes only by Split or Merge Jobs on the target Product: the new Product items inherit the aggregated or proportionally distributed Scope3Footprint values.

5.6.3 TotalFootprint

The TotalFootprint can be calculated for a Product at any given time. Its value starts with the Scope3Footprint, then increased by all emissions related to all items in the UsedResources item. For that calculation, it uses the amount of all identified resources multiplied by the emission factor of that resource, these factors are given in the Protocol. When buying a Product within the system, the initial Scope3Footprint will be the TotalFootprint of the purchased Product.

In a production environment, the Protocol is immutable by the users, but over time (quarterly, yearly) a new version appears. The TotalFootprint always refers to the protocols used in the calculation.

In a research environment, the Protocol administrators can change the Protocol, and the system recalculates all Footprints on the fly. Similarly, in research mode users can clone the data content and change values like replace fertiliser types and recalculate Footprints.

5.7 ExtInfo

The system uses ExtInfo blocks to store any additional information to any item. It contains the entity ID, the type and the period and only one item can be valid by the period for any given entity and type, the "current" is without an end date. When a new ExtInfo of the same type is registered for an entity, the moment is set as the end of the current block and the start of a new one.

The definition of the data content comes from the Protocol for each type and stored as a JSON payload in the block. The block also contains the source of the block (user, automated calculation, independent lab, etc.)

Apart from the previously mentioned applications, ExtInfo blocks can be used to store data for

- Measurements like soil sample analysis, grains moisture and nutrient values, etc.
- Method commitments: the farmer declares that a Plantation will follow externally specified rules (stored in the Protocol). The Protocol can also contain validation expressions that allows continuous monitoring and final certification at the harvest, without manual intervention.

6 Protocol

The Protocol contains everything that can change over time, and for simplicity, the overlapping part of the system itself (except for the technical modules). The complete Protocol must be saved in simple files (CSV / JSON / XML) that is human readable and completely describes the whole operation (the original goal of XBRL).

- **Meta definitions**: types, attributes, possible values, native format: JSON. This describes the "tables" of the collected data items and the structure of the Payload in ExtInfo and Footprint items. The system handles all items by consulting their meta definitions.
- **Info tables**: parameter values for calculations, native format: CSV. These are like
    - The CO2 footprint of different fuels used in different ways.
    - The reported footprint of any manufactured StockMaterial type by mass or volume.
    - The Oil / Gas / Solar / … ratio of electricity at a given location and time.
    - …
- **Calculations**: expressions and scripts that can access data in the system and calculate values. Given in text using the MVEL syntax, stored in XML for multiline

support. The calculation definition contains the entry point (like a Job, a Plantation, Product, etc.), that is the root from which it can browse the data.
- **Templates** for automated text generation (like HTML / iXBRL reports for published footprints) using the MVEL syntax.
- **Configurations** for custom data exports, like how to fill or create an external Excel file required by an external entity (buyer, bank, government).

The Protocol is modular and can refer to each other, the reference contains the version. The system stores all versions of all Protocols ever published and can provide it in their native format. For current operation all members must use the one "active" version, and all data items (and exported reports) refer to the used Protocols.

7 Backend "Master" Server

7.1 Active Data

This is a 3-tier application, semi-stateless server: when a user starts interaction with an Entity, the backend loads whole active graph into memory. Assumptions:

- The active data segment is always limited to the current state and the ongoing processes with around 10k items pure data each.
- If the entity counts and concurrent actions would require too much memory (not likely), regional splitting is a possible solution.

The system does not have session-level state management, sends the requested parts to the client, receives committed changes and updates the entity graph in a locally synchronised process. It updates the persistent store within the synchronised block; and as the entity graphs are generally disjunct, no collision is expected. A special case is an Entity interaction (buy / sell action, neighbour Job registration), in this case both Entity graphs are locked, and the shared update is executed in the same transaction.

The Entity graphs can be released from and reloaded into the server memory anytime, regardless of the active clients. The system can prefer keeping the big ones in memory all time (expecting more frequent calls) or even optimise management by learning usage patterns to improve performance.

7.2 Persistent Store – File System

A fundamental non-functional requirement is Zero Trust: all parties should be able to review their own data outside the system. Active data will be stored in simple, human readable (CSV / JSON / XML) files, a SeqCarbon client can run on an external machine, load Active Data from this store and execute all operations including repeating calculations using the Protocol.

The prototype version only uses File System store and JSON format, because the implementation is simpler and the performance benefit of CSV is minimal.

7.3 Persistent Store – Database

File System store is not enough for the real SeqCarbon portal, it should handle thousands of clients, support proper backups, and allow handling historic data as well.

The core object definitions are simple for this reason: they will be stored in a relational database in tables for quick access through indexing.

Regarding the customisable types (ExtInfo, Footprint, etc.), those tables will be indexed by the owner item and type (queries locate the interesting ones very quickly). Even free databases like MySQL or PostgreSQL supports JSON fields and allows accessing their content.

Naturally, the SeqCarbon master database must be in sync with the client actions, configured for high-performance queries and updates, the historic data can be stored on other servers for less frequent updates. Similarly, either the whole or a target segment of the database can be replicated to other servers for near-online, complex analysis without decreasing the master server performance.

8 Frontend Server

In the prototype, the same server executes Backend and Frontend services. For performance and security reasons, the Frontend (accessible from the Internet, multiple instances for speed, read access to the store) and Backend (only visible from the Frontend servers, single instance, with write access to the store) separation may be necessary.

The Frontend can read the database, may cache Entity graphs in its local file system and respond to client queries until a change is committed. In that case, it forwards the update to the Backend that ensures synchronisation and proper transaction handling; the Frontend forwards the response to the client.

This is also important because the system will use server-generated simple HTMLs in the first time to maximise compatibility; this is a typical horizontally scalable Frontend task.

Another important function of the Frontend is access control: user login, filtering response data, and access control to services, although the Backend executes all access checks again on every transaction. Regarding the login, the role management is local; the authentication also local for the first time, later external providers (Google, Apple, etc.) or even MFA may be included.

9 Data Access

9.1 Native

All data (and the Protocol configuration) in their default (CSV, JSON, XML) format, as they are stored in the file system. This format is used for

- caching inside a server,
- between the Frontend and Backend,
- when an entity runs SeqCarbon locally, between their server and the main portal,
- when entities download their data for local management or validation,
- between any future SeqCarbon native client and the main portal.

9.2 JSON:API

JSON:API is an open, standard protocol to access data in a system. If any external tool wants direct access to the SeqCarbon data but without dealing with the native formats, they can use this format and protocol. -> https://jsonapi.org/

9.3 Templated text

The system uses the MVEL engine that allows creating text file templates with “callouts” to data. The official exports like XBRL (both the pure data XML for direct, automatic Scope 3 communication and the customisable, formatted iXBRL / XHTML variant) will be created using these templates.

Theoretically, any external text-based standard can be supported by these templates, probably by extending the data access API or adding services to the used Protocol definitions. With some practice, entities can create MVEL templates on their own, from simple data exports (some special CSV) to monitoring dashboards.

9.4 Generated HTML GUI

The primary user interface of the system is based on server-generated HTML fragments, the skeleton is an MVEL template that the engine fills with the current data content. The template itself uses the meta definition of the data (to generate columns for the table fields, rows in a property sheet, etc.), then iterate the data (items or properties) to access the value and write into the HTML fragment.

The fragments also interact with the server, like a grid with add / delete row buttons post to the server, it creates or deletes the target row in the item, generates the new fragment and sends it back to the client. In this way, the first version has no client-side dependency, can run on any platform that has a web browser.

The drawback is the lack of immediate input validation. Idea: a button with validity states (green tick, yellow triangle, red X) that turns to grey ‘?’ on any change, pushing posts the current content to the server and gets the response that includes the new state or even a list of messages.

9.5 Binary Exports

Primary example is Excel import and export, supported by Apache POI, either by filling configured cells in an externally provided template, or creating a new file. The first version is better for externally formatted and structured, complex exports, the latter is for simple data exports that can immediately be used in Excel, generally containing multiple related worksheets.

Naturally, other binary exports can be added, if the required library is available in Java and the task is configurable.

10 Data Input

10.1 Direct Entry

Manually entering the items of an Entity into the store (copying / editing the data files or populating the database). This can be done while registering the entity into the system after a survey. The prototype will be initialised this way, before implementing the related GUI services.

The important factor is that in this mode, automatic actions don't work, items can be deleted.

10.2 Admin Interfaces

The meta description of the core and customisable payloads allows automated default editor generation using simple MVEL core templates (a generic grid or property sheet that gets the actual meta definition as parameter) for CRU(D) actions, the item updates are executed using the native JSON format. Delete does not remove the item from the persistent store, only makes it obsolete, meaning it is removed and never loaded into the Active Data graph, but remains available in the history. Naturally, this operation affects all items referring to the removed one and should be reviewed.

The admin CRUD editors will be applied for User, Entity, Site, Parcel, and Asset by default; ExtInfo with limitations in the long run as well. All other types are related to Jobs and handled automatically; they are read-only.

10.3 Bulk Imports

The counterpart of Binary or Templated data exports; supports loading data from external systems like GateKeeper. It is essential to have unique record identification in the source to avoid loading the same external item multiple times. The item should contain information about the actual import source in an ExtInfo block (this is another reason to have multiple ExtInfo blocks assigned to any item without restriction). After the import, the process log and the import file must be stored in the Binary Store.

The import process must know the file type, that selects an import logic from a Protocol (may be given by the creator of the file, like GateKeeper). This logic can see the content of the file through a format-specific parser (the data content of a CSV, JSON, XML, etc.) and is responsible for executing data management actions in the Entity, like creating jobs and execute them. The logic is responsible for proper error handling and logging. The platform is responsible for transaction management, all changes are accepted or rejected in one transaction:

- If there is no warning, the system automatically commits the changes.
- Any error means immediate termination and rollback.
- On warnings, the user gets a summary of the operation and can decide to keep or reject the changes.

Scope 3 XBRL data sheets are imported using this feature.

10.4 Job Management

When looking at Parcel, StockMaterial, Plantation, Product items (generally in a list or tree), users can create a Job item related to them, select type and configure parameters (ExtInfo, JobItem) depending on it. The Jobs can be reviewed in their own CRUD editor but editing them is more complicated because of the type-related logic.

Saved Jobs are in "planned" state. At any time, the Job can be "executed", after that the Job cannot be modified. Execution triggers all the logic assigned to it in the Protocols: create the configured Plantation on the Parcel, harvest it and create the Products, etc. Some of the attributes in the payload can only be set on execution as they contain the real-world result of the action that the logic will store in the system.

Job planning allows simulation, and it will be used in the prototype: we set up LabFarm1 with Direct Entry, that contains the planned jobs for the season, then advance the date, automatically execute jobs by their planned date and verify the progress. Another option is bookmarking a state of the entity and return to it by one action, change some parameters and run the simulation (or the planned jobs) again to see the difference.

11 Internal Data Access

This is the API to access data from the MVEL logic that can be an expression, script or template given in plain text. The main idea is that these logics have their own configuration (so they can be configured, probably reuse the same logic with different parameters), and a target item – in most cases that is a Job, so the logic can execute the required changes on the related items; but can be a Product to calculate the footprint or generate files like an XBRL instance.

The API should be simple and with some caution, the scripts can even be executed in a JavaScript client? The MVEL template has different syntax but there is no plan to generate exports in a browser anyways. MVEL templates can also generate custom exports on the server side – probably an advanced feature for clients?

The MVEL actions seem to be relatively limited, react to simple external events, no complex interaction, change listener, etc., therefore, no access and recursion protection is planned.

12 Binary Store

There are final files, like the Scope 3 data sheet of a product. The store (if in the filesystem) is segmented by year with double-hashed folder structure to make archival easier and avoid having too many files in one folder. If the system has a database store, the files should also be stored there in blobs. This is a configuration setting in the backend server, invisible to the other modules. Of course, stored binary files can be loaded form the store by their unique ID.

These files are created once, optionally together with a hash to ensure no tampering. A remote SeqCarbon node must send the hash to the centre even if the file is private, in this way the receiver can check the hash in the centre to validate its content (?)

13 Use Cases

13.1 Public Data Access

There are data segments in the system that are not protected by any access control. These are

- All static part of the web interface.
- Intentionally public exports like some farmers may want to publish data, a very low footprint, achieved certification, etc. and the SeqCarbon portal provides credibility to these results.
- Custom taxonomies created automatically from the Protocols to be included in the formal XBRL reports.
- ...

If the content is generated, should be cached in the filesystem and accessed directly; should completely reside on the Frontend if separated.

13.2 User Registration

Set user id and password; must provide an email and an Entity. The Entity must exist; registering new Entity is the task of a SeqCarbon system administrator. SeqCarbon is a fixed special Entity, the first user can only select SeqCarbon and as it is empty, will become an administrator. If an Entity has administrators, they must accept the user registration, they manage the administrator flag to the other users except their own to ensure that each Entity has administrators.

13.3 Login / Logout

In the prototype, standard login, but the Entity is also on the page, loaded from a cookie. In this way, the user list is under the Entity, easier to handle account collisions. As the first step of accepting the login, the server must load the target Entity for the credentials, and it (in general) stays in memory to support interaction.

The system has a default session timeout for security, or the user can log out manually. In both cases, the session is terminated and the Entity active user count decreased. After the last logout, the Entity gets listed for removal from memory.

13.4 Item Property Page

This is a generated, read-only HTML page. On the top, the properties of the current item are displayed in a key-value table. On the right side, there is a generated QR code for the web address of this page. This QR code can be printed on a paper, label, etc. and scanned to reach out to the target item. Below the property sheet, all items that are related to the current item appear in grids grouped by the item type.

- Every reference to another item (including the first column of the related item grids) are hyperlinks to display Property Page of the current item.
- Some item types have special actions assigned to them: you can export Footprints to XBRL or CSV; or display the Process graph for Products, these actions appear at the top of the property sheet as buttons.

- Jobs in Planned state can be executed; the button appears as the value of the job status. Pressing it leads to the Entity main page, Job grid so that you can continue executing them.

13.5 Product Process Graph

The Product may have a complex history with many Jobs including Merges and Splits that create new Product items. A buyer wants to see the whole story, not only until the last creation Job, but all members and all actions done to each since they were bought or harvested. The Process graph displays all related Product and Job items on the left, can be used as a navigator: clicking an item displays its property page on the right.

Naturally, access control applies here: Users of the buyer Entity can only see the Product that they have bought, while the owner of the Product can see all details.

14 Further Ideas

A collection of ideas, not to be implemented in the prototype phase.

14.1 Measurements

Labs can register to the system as Entities; with the measurement types and configurations they provide (like soil sample analysis). They register the result of a measurement in the system, although the ExtInfo block belongs to another Entity (the owner of the Parcel) because they are configured as the ‘source’ of that measurement.

14.2 Research

Every data access is limited to the actual entity, except for the Buy / Procure Job pair, this is checked when the logic reaches out to data. This applies to the special Protocols as well, like a scoring system that gives scores to farmers depending on their methods: the farmers attach the Protocol to their Entity, and this is how the Protocol logic can access their data and generate a score as an ExtInfo block attached to the Entity again.

However, a research group can also appear as an Entity, create their own special scripts and research projects using them. They can then send participation request to Entities and those who accept them will be listed to the research project and can access the configured subset of their data (including anonymisation or various levels of aggregation, like don’t show individual field jobs, only the aggregated amounts of StockMaterial types). Both parties can break this relationship.

14.3 Data Protection in the Centre

Clients can use RSA algorithm to protect the sensitive part of their data even if that is stored in the centre. This by default affects the Payload field that contains all details of any operation, but they can also configure to encrypt fields in the core data as well. The risk of this way is that if they lose their private key, they lose their data; also, such Entities do not participate in scientific research automatically. However, they can allow access to a selected subset, in this case their client should export that segment into native format and upload to the centre without encryption.

14.4 Tamper Protection

Large players with sufficient resources and safe local (or cloud) IT infrastructure may prefer running their own SeqCarbon server. This is OK either with filesystem or database persistent store, so they can integrate the data directly with their other systems. They send only data to the SeqCarbon centre that they want to make accessible for others, like their Scope 3 sheets or fragments requested by research projects.

The question is, how to avoid local data tampering. The idea is that whenever an item is stored after creating or updating, an RSA hash must be created and both sent to the centre (timestamp, item id and the hash) and stored locally. The same applies to any file export, especially published reports. This way, the centre knows the hash and the time, the auditor can run the creation at the client site (accessing all data) with the centre timestamp and must get the same result as stored in the centre.

14.5 Zero Trust

A Zero Trust environment means “truth” is not something that an authority says, all parties can validate any statement. Reliability, transparency and clear separation of responsibility are essential in this project, as the participants depend on the validity of the data passed over the value network.

Cryptographic signatures and encryption serve as proper guards for data access, but the same must be done with calculations as well, e.g. any party can independently validate that their emission results that appeared on the SeqCarbon portal are calculated as they should have been. The XBRL addressed this goal (taxonomies integrate meta definitions, validation and calculation rules) but the solution is too complicated to be efficient, as the 20+ years of experience in finance demonstrates it.

All calculation-related data (meta descriptions, additional data tables, the calculation scripts and report templates) must exist in versioned packages of externally readable configuration files, nothing in code. These are the Protocols in CSV / JSON / XML files. These blocks are naturally also digitally signed so the user can check if their content is the valid.

The calculation system must be open source and usable for non-profit application – like, verifying my own data and the calculation results. The implementation must have minimal requirement, so that any reasonably modern personal computer can execute – or for more advanced users, even built from the official sources before running, or tracing in a development environment by experts.