Marko Junkkari and Antti Sirkka

Formal Definition of Traceability Graph

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 3

TAMPERE 2011

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 3
NOVEMBER 2011

Marko Junkkari and Antti Sirkka

Formal Definition of Traceability Graph

SCHOOL OF INFORMATION SCIENCES
FIN-33014 UNIVERSITY OF TAMPERE

ISBN 978-951-44-8652-4

ISSN-L 1799-8158
ISSN 1799-8158

Formal Definition of Traceability Graph

Marko Junkkari

School of Information Sciences
University of Tampere
Marko.Junkkari@uta.fi

Antti Sirkka

Tieto Finland
Antti.Sirkka@tieto.com

ABSTRACT

Data-centric workflows focus on how the data is transferred between processes and how it is logically stored. In addition to traditional workflow analysis, these can be applied to monitoring, tracing, and analyzing data in processes and their mutual relationships. In many applications, e.g. manufacturing, the tracing of products thorough entire lifecycle is becoming more and more important. In the present paper we define the *traceability graph* that involves a framework for data that adapts to different levels of precision of tracing. Advanced analyzing requires modeling of data in processes and methods for accumulating resources and emissions thorough the lifecycle of products. The traceability graph enables tracing and accumulation of resources, emissions and other information associated with products. The traceability graph is formally defined by set theory that is an established and exact specification method.

Keywords
Data-centric workflow, data model, lifecycle data management

1. INTRODUCTION

In many applications, e.g. manufacturing, workflows are used to model processes and the relationships among them. Processes are widely studied from the perspective of process modeling but seldom investigated from the perspective of the requirements of data models.

Data-centric workflow [4, 8] approach for designing workflows is based on defining how the data is transferred between processes and how it is logically stored. The approach examines how processes transform data and which entities send and receive the data. The main goal of data-centric workflow is to present the data sets in the workflow. In other words data-centric workflows must involve an integrated data model for storing and manipulating data.

In the present study, we develop a data-centric workflow approach involving such a data model. It supports dynamic data management techniques for data refinement such as aggregation, attribute value propagation, and embedded functions (derived attributes). Integration of object (entity) transformation (object division/composition) with other dynamic data management issues gives an advanced approach to analyze data associated with processes, products the processes yields, and their components. In our terminology product is a general term used to refer to objects resulting from a process. An identified (database) object is an entity represented with its properties.

The problem of focusing on the correspondence between database and real world objects (entities) is as old as the databases in general. Chen [9] defined that the entity is a “thing”, which can be distinctly identified. In object-orientation (see e.g. [7, 10, 24, 26]) an object is a thing that has existence *perse*. The problem of these approaches is how identifying of real world objects corresponds to database objects and vice versa. In practice a database object may represent e.g. 1) a single real word object, 2) a set of real word objects, or 3) a mass of material that can be identified e.g. based on its usage at a time. Furthermore, composed objects, having parts organized at several hierarchy levels, have their own manipulation needs (see e.g. [18, 23, 24]). Data modeling and manipulation in workflows share the complexity of composed objects because transitive relationships must be managed. Further a workflow may structurally correspond to a composed object because it can describe the related composing process.

In our approach a database object may correspond to a real life object, object set or mass of material that can be physically identified thorough the part of process chain in which it is participating. This means that physical objects of a patch are manipulated by a single database object if physical objects can not be individually identified. It is worth noting that in our terminology, the object set means the set of database objects, although a single database object would refer to a set of physical objects.

In data-centric workflows, the manipulation of objects requires specific features because objects may be changed into other objects in both the logical (in databases) and physical (real world) levels. Namely, an object may be divided into other objects or several objects can be composed into a single object. This means that object transformation must be modeled. In addition, objects are manipulated in patches that can be divided into subsets. In turn patches may be collected into larger patches. Table 1 summarizes the cases of object transformation.

Table 1. Identity manipulation

Transformation	**Identity**	**Description**
equivalence	maintain	a patch is transferred from a process to another as such
subsetting,	maintain	a patch is divided into subsets for refining
supersetting,	maintain	several patches are collected together for refining
division	change	objects in a patch are divided into several objects
composition	change	objects in patches are composed to single objects

We use the term supply chain, borrowed from manufacturing, to determine all the processes that are participating directly or indirectly in the production of a product. A supply chain is a directed subgraph of a workflow diagram. Structurally, a supply chain consists of supply processes following each other in a partial order, i.e. the result of a process is a raw material for another process. Processes possess properties, such as resources and emissions, associated with the result products of processes. From data-oriented perspective, these properties are accumulated in a supply chain, i.e. the information on the preceding processes of a process is also associated with the process and its products. For example, if we want to calculate used energy of a product, all history (preceding processes) must be taken into account. For modeling this accumulation of products, a process contains two values: ordinal and cumulated where the ordinal value is focused on an underlying

process whereas the cumulated value is aggregated from previous processes. The information from a process to another is sifted by derived attributes.

In data models the intensional (schema) and extensional (instance) levels are typically distinguished. In models of complex structures, such as composed objects, the strict correspondence between these levels is an essential challenge because instances (objects) of an object type may structurally vary from each other. The same concerns many applications of workflows, i.e. the supply chains of similar objects may vary from each other and different data are associated with objects and their components. For management of the structural diversity of composed objects, integrating the intensional and extensional levels with each other is proposed [18]. This allows advanced structural analysis and declarative query formulation thorough transitive relationships [21]. In the present paper, we adopt the idea of integrating the levels into each other, i.e. we do not have explicitly separated workflow schemata and instances. The same concerns the data associated with workflows. This gives a possibility for flexible forming of data-centric workflows.

In the present study we define a data-centric workflow model called the *traceability graph*. In general, the following goals are attached to the traceability graph.

1. **Embed logical storing structures of data into a workflow model**. Data associated with processes and products must be stored within data structures from which it can be search and redefined.
2. **Ability to manipulate objects, object sets and their transformation**. In processes, object sets can be divided into subset or be unionized into larger sets. Single objects can be divided into smaller objects or single objects can be composed from other objects.
3. **Support for querying and analysis of data**. The supply chain with cumulated resources and emissions of a patch or single objects can be derived from a traceability graph.
4. **Ability to manipulate the properties of processes and allocate them to different products.** The traceability graph involves the *cumulated values* of ordinary attributes and the concept of the *derived attribute*. The cumulated value is deduced from attributes based on previous processes. Derived attributes are associated with sifting of products from a process to another. The related calculation rules are based on rations of the amount of products in processes and their transforming among processes.
5. **Formal generality**. In formal specification we use set theory that gives freedom to implement the system in different database paradigms, e.g. relational, object-oriented or deductive data models.
6. **Application independency**. The model is not bound to any specific application area. Our sample application is from the forest industry, but the model can be applied to other domains.

The rest of the paper is organized as follows. In Section 2 we present a short survey on workflow models. In Section 3 we aim to motivate our work by introducing different use cases of data-centric workflows. Among them, we focus on the life cycle assessment because the requirements for strict tracing demands advanced analysis that is applicable to other use cases. Section 4 deals with the graphical notations of the traceability graph and an informal introduction for our sample application domain. The used mathematical notational conventions are given in Section 5 and the formal representation of the traceability graph is given in Section 6. The analytic capabilities of the traceability graph are described in Section 7 and in Section 8 we present problems and solutions to share the data of data-centric workflow with several stakeholders. Finally, the conclusions are given in Section 9.

2. RELATED WORKS

For describing functionality (dynamic aspects) of information systems, various graph based methods have been developed. Early data flow languages e.g. [20] were primarily intended for modeling the activity of computer programs, whereas workflow models [3] are focused on modeling physical processes and activities. Historically, the distinction between dataflow and workflow models is vague and some authors [13] use the term “information flow” for describing flowing of all information from a process to another. Nowadays, different types of activities are typically distinguished in workflow diagrams, like sending, transforming and packing of materials. Some modern modeling methods of information systems, e.g. UML [7], contain diagrams for dynamic aspects of programs (interaction diagrams) and modeling of workflows (activity diagram).

In workflows materials, documents and other information are transferred from a process to another [3, 6]. The workflow model of UML is informally defined and its purpose is to map real world activities to the underlying software solution (or visa versa). There are also a number of commercial applications that have a component for drawing workflow diagrams. Their common feature is that they support the illustration of different types of processes.

There are also formal methods for modeling functionality of information systems that are not primarily intended to any specific purpose. For example Petri nets are traditionally used for defining or describing functionality of computer programs, but they are also proposed for exact representation of workflow models [1], [3]. YAWL [2], Temporal logic [5], and Transaction Logic [6] are other good representatives for formalizing workflows. These methods emphasize timing within processes and supply chains. In general, these are very expressive languages that enable not only modeling of the functionality of systems but also other aspects of information systems. For example Transaction Logic is based on F-logic [20] that is a general framework for specifying object-oriented and deductive aspects. Workflow models have been investigated from the perspective of how they support different data-centric aspects [11]. The main aspects include concentrating on process functionality based on the data, i.e. each node has behavior instructions with regard to the data.

From our perspective, timing representation in YAWL, Temporal logic, and Transaction Logic is a secondary feature. Instead we emphasize handling the data aggregation and movement between processes.

Deutsch and others [12] present an advanced data-centric business processes model and its verification. They define several essential primitives such the artifacts schema, artifact instance, and service logically integrated with each other. Our study differs from their approach. First, instead of separated intensional and extensional levels we integrate these composed data structures. Of course, behind our approach may be a predefined data schema, but the data schema can also be derived from the traceability graph. In other words, we do not specify whether a data schema is predefined or derived. Second, we address the explicit object sifting, transformation and derivation of the information thorough of processes.

We aim to find the principal primitives needed to model and manipulate data-centric aspects in workflows in a way that enables tracing of products at different granularity levels. We do not bind our model to any existing data or workflow methods, i.e. we give freedom to apply our model into existing formalisms and systems.

In the related previous studies we have presented only graphical notations of the traceability graph and its implementation in relational data bases [19]. Based on the relational representation we

have introduced possibilities for OLAP analysis [26]. However, these studies were ad-hoc. Now we give a general formal definition for data-centric workflow model.

3. MOTIVATION

The traceability graph is a data-centric workflow model for tracing, analyzing and querying data. Next we introduce some practical use cases and their background for the Traceability Graph.

3.1 Life Cycle Assessment

The product level environmental impact assessment has become more and more important. The main approach for assessing this are international standards of the life cycle assessment (LCA) (ISO 14040 series [16]) and eco-labels (ISO 14020 [15]) and verification (ISO 14064 [17]). ISO has also started to develop standards for Quantification and Communication of the Carbon footprint of a product (ISO 14067).

The life cycle assessment (LCA) is a standardized method for calculating the environmental burden of a product throughout its lifespan, from raw materials to the disposal/recycle phase. The goal of LCA is to compare the environmental impact caused by a product so that the customer can choose the product that causes least damage to the environment. LCA has four main phases:

In the first phase, the goal and boundaries of the life cycle assessment are defined. In other words processes of the supply chain of the product are defined. In this phase also the functional unit is specified. For example a cubic meter of timber, one mobile phone or one tomato.

In the second phase, illustrated in Figure 1, called life cycle inventory analysis, each process is analyzed and the input and output flows of the processes are defined. There are two basic flow types.

- Elementary flows describe inputs of raw materials and energy resources, and outputs of waste and emissions.
- Product flows describe inputs of products which are an output of other processes, and outputs of products, by-products.

The third phase of LCA is impact assessment where the results of the life cycle inventory analysis are assigned to the environmental impact categories (e.g. Climate change, Ozone depletion and Acidification). For example, the carbon dioxide and methane are greenhouse gases and are assigned to the Climate Change category. The fourth phase is the interpretation where conclusions of the analysis are made.

The traceability graph, defined in the present paper, can be used for tracing and storing the inventory data, the impact assessment and interpretation phase is excluded. Unlike existing methods our model enables analyzing resources and emissions on the single product level – not only average values. Nowadays a common method for calculating the environmental impact is to measure the input and output flows of the whole supply chain during some time period and calculate the average environmental impact for the product (see e.g. [22]).

We have designed our sample system from the perspective of life cycle assessment in order to demonstrate traceability in a forest industry area. We especially aim to demonstrate possibilities that the traceability graph gives for tracing products that are composed from several components, and the components, in turn, have been divided from larger sub-products.

Although we emphasize environmental aspects in our example, the traceability graphs can be applied to other advanced tasks needed in different application domains.

3.2 Origin of Raw Material

The organizations are facing more pressure from consumers and legislators to accurately report the origin of their raw material. However, the complexities of today's supply chains pose a challenge for gathering this data accurately. For example in the forest industry the companies are certificating their product using PEFC[1] chain-of-custody certification. The PEFC chain-of-custody certification is a method for tracing wood from forest to the final product to ensure the wood or wood-fibre can be traced back to certified forest. The certification has two methods of realizing this:

- Percentage based method – the method allows mixing certified and non-certified raw materials taking into account that the percentage of certified raw material must be known. Company can sell as certified the proportion of its production, which equals the proportion of the certified raw material
- Physical separation method – the method requires certified and non-certified raw material to be physically separated throughout the supply chain.

Naturally, the methods can be integrated and improved using the Traceability Graph. The accurate item level information of the origin can be used to create a transparent and trustworthy certification system for origin.

3.3 Recalling Products

The traceability graph enables better and more accurate service for recalling unsafe, defective or hazardous products.

The item level traceability allows manufacturers to recall just the particular products that contain the unsafe elements. For example Toyota recalled approximately 9 million cars in 2009-2010 for pedal entrapment/floor mat problems and accelerator pedal problems. If the reason of recall is for example a faulty set of components, a manufacturer could have only recalled those cars that include the faulty components, not the whole set of manufactured cars in some time period.

The item or patch level traceability can also be used in food industry to avoid total recalls. The traceability graph enables manufacturers, suppliers and resellers to identify the products where some suspected raw material was used and all customers whom the products were delivered. Using the system companies are better equipped to retrieve the affected products and to protect their reputation and brand value.

3.4 Benchmarking

The traceability graph can also be used for benchmarking the processes. By comparing the environmental performance of the same type of processes companies can detect whether the performance metrics of their process are up to the best practices in the industry. This allows companies to notice where they could achieve the biggest improvements in their environmental performance.

[1] http://www.pefc.org/

4. INFORMAL DESCRIPTION OF TRACEABILITY GRAPH

The traceability graph can be used to model the supply processes of products and the data associated with the products. The traceability graph has an ability to manage the transformations of the products. For example a product may be manufactured using masses of raw materials or it can be composed from many parts. The traceability graph also has the ability to allocate the properties of processes to the products handled in processes.

Next we introduce the primitives of the traceability graph, a rough graphical representation, and the sample system used from now on.

4.1 Primitives of Traceability Graph

The traceability graph consists of nodes, edges and their properties. A node describes a process whereas an edge describes a flow between processes. Next we introduce data-centric primitives associated with nodes and edges.

- An *object* is a product unit uniquely identified in a supply chain. It may correspond to a single product or a patch or a mass of material, depending on the precision of the actual traceability system.
- A *product portion* determines a patch of products of a node. It can be associated with a set of objects or a mass of non-identified material. The product portion involves a ratio which is used to allocate the costs (e.g. emissions and resources) of the node to which the product portion belongs.
- An *attribute* of a node describes process information – elementary flows. Input attributes represents input costs, e.g. raw materials and resources, used in the process. Output attributes represents output costs, e.g. emissions and waste, generated in the process. For other information of processes the info attributes can be used. Each attribute has two values: an (ordinary) value for the underlying process and a cumulated value which is calculated from the preceding nodes via edges.

As a whole, a node involves an identity, a set of product portions and a set of attributes. An edge is identified by participating nodes, called start and end nodes. Furthermore it contains the following primitives:

- A *sifted product portion* contains products that are sifted from a process to another. A sifted product portion may involve only some objects from the original product portion.
- *Object mapping* belongs to a sifted product portion. It can be equivalence, subsetting, supersetting, division or composition. For example subsetting means that only a part of the product portion is selected for refining. Division means that products of the start node are divided in the end node. Composition means that products of the start node are components for the products of the end node. It is worth noting that unlike in set mappings the identities of objects are changed in division and composition.
- A *derived attribute* determines the quantity of emissions and raw materials associated with the sifted product portions. It is calculated using process specific rules. For example, the volume or the cost of the product can be used as a factor.

4.2 Graphical Notations

The graphical notation for the traceability graph is a rough level description about processes where data-centric primitives are mainly hidden. However, it gives a framework for information transferring among processes and illustrates object transformation (division and composition) between processes.

In the graphical representation, a node is illustrated by a circle, and an edge by an arrow – i.e. we follow the traditional illustration for graphs. Three kinds of edges are notationally distinguished:

- *Plain edge* is presented by a plain arrow and it determines that objects are transformed from the start node to the end node as such. This covers equivalence, subsetting and supersetting mappings of objects.
- *Division edge* represents that the objects of the start node are divided in the end node. In other words, several objects in the end node are mapped to one object in the start node. This is illustrated by a double head arrow.
- *Composition edge* means that objects in the start node are components for an object of the end node. In other words, several objects in the start node are mapped to one object in the end node. This is illustrated by an arrow having divided start.

4.3. Sample System

In terms of our example from the manufacturing and forest industry we demonstrate the use cases presented in Section 3. In Figure 1 a simplified production of glued laminated timber is illustrated. The processes included in the example are felling the trees (harvesting), sawing logs to boards, drying the boards and jointing the glued laminated timbers from boards.

The first phase of production has three instances (nodes), describing the amount of daily harvesting. The movement of the products is illustrated from to left to right in the graph. For example, the products resulted from the harvesting nodes are transferred to the sawing node, a double headed arrow illustrates that in the sawing nodes, the objects (logs) from harvesting nodes are divided into several objects (boards).

Between Sawing and Drying nodes the objects are not changed. This is illustrated with a plain arrow. The mapping between the object sets in these cases can be equivalence, subsetting or supersetting. In other words the objects from the preceding node can be moved as such, or only part of the objects can be moved, or all the objects can be moved from the preceding node together with some objects from other nodes.

In the gluing nodes the objects from the drying nodes are composed as an object of the gluing node. In other words objects from the drying nodes are components of objects in the gluing nodes. An arrow with a divided start illustrates this. The emissions and resources associated with the products of the drying nodes are accumulated to a new set of objects in the gluing node.

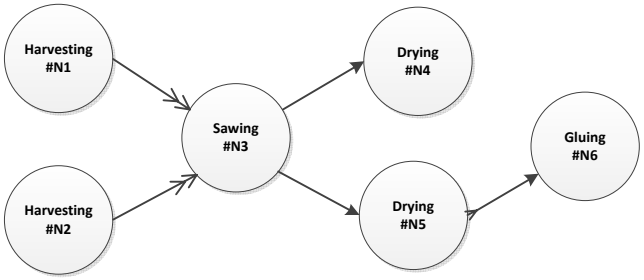

Figure 1. Sample Traceability Graph.

A supply chain of an object can be viewed as a network of processes that are associated with the product during its manufacturing. Using the information of the traceability graph it is possible to track the supply chain of the object throughout its entire supply chain and allocate all the information related to those processes to the object.

Given the running example, we are tracing the environmental burden of the glued laminated timber that belongs to the Gluing #N6 node. Then the supply chain of this glued laminated timber is the processing history of the objects. In Figure 2 the colored nodes are the processes that constitute the supply chain of the paper roll. For example, the glued laminated timber has participated in the nodes Drying#N5, Sawing#N3, Harvesting#N1 and Harvesting#N2. This sub graph of the total traceability graph is the supply chain of the glued laminated timber.

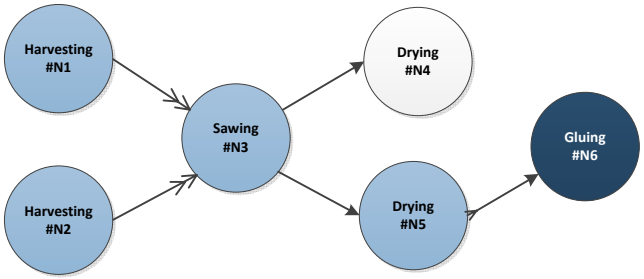

Figure 2. Sample supply chain for the object in node Gluing #N6

In the related formal example we demonstrate the precision of tracing that the tractability graph serves. Before that we introduce the used notational conventions for the definitions.

5. NOTATIONAL CONVENTIONS

Standard set theory is used for representing the traceability graph. Next we introduce only those notational conventions which have widely used alternative representations.

- *Tuple* is an ordered sequence of elements represented between angle brackets. For example ⟨a,b,c⟩ is a tuple.
- If t is a tuple and x its uniquely labeled member then t.x refers to x in t. For example if t = ⟨a,b,c⟩ then t.b refers to the second member of t.
- If it is not necessary to refer to a member of a tuple the underline space can be used. For example in 3-tuple ⟨_,x,_⟩ the first and last members are not referred.
- The power set of the set S is denoted by P(S)
- Cartesian product between sets A and B is denoted by $A \times B$.
- Mapping *f* from a set X to another set Y is a 2-place relation ($\subseteq X \times Y$) denoted by $f: X \rightarrow Y$. X or Y may be a set consisting of sets, e.g. a power set.

- If R is a 2-place relation $R \subseteq X \times Y$ then the *domain* of R ($\{x \in X \mid \langle x,y\rangle \in R\}$) is denoted by *dom*(R), whereas the *range* ($\{y \in Y \mid \langle x,y\rangle \in R\}$) is denoted by *rng*(R).

A (directed) graph is a pair (N,E) where N is a set of *nodes* (vertexes) and E is a set of *edges*. Nodes and edges are represented by set theory as follows:

- A node is represented as a tuple $\langle$Node-id, P_1, …, $P_n\rangle$, where Node-id is the identity of the node and P_1, …, P_n are the properties associated with the node. For brevity, Node-id can be used to refer to the node. Thus, the notation Node-id.P_i refers to the property P_i in the node having the underlying identity.
- A directed edge is represented as a tuple $\langle$Node-id$_S$, Node-id$_E$, P_1 ,…,$P_n\rangle$, where Node-id$_S$ and Node-id$_E$ are the identities of the start and end nodes, respectively. P_1 ,…,P_n are the properties associated with the edge.

6. FORMAL REPRESENTATION OF TRACEABILITY GRAPH

In the traceability graph each node describes a process where resources are needed or new costs (e.g. environmental impacts) emerge. A node involves a set of attributes and a set of product portions. These are the properties of the node. An edge describes division, composition or transferring of products portions. Each edge possesses a set of product portions which are shifted to the following process. In an edge neither new resources are needed nor new costs are emerged, i.e. ordinary attributes are not associated with an edge. Instead, an edge may involve *derived attributes* that describe portions of previous product portions and attributes. In other words, *sifted product portions* and derived attributes are properties of an edge. Next we define our model in detail.

Products which are identified physically and logically in the application domain are called *objects*. For logical identifying each object of interest possesses an identity that is either an integer or a string (code used in the application domain at hand). The set of possible identities of an application domain is denoted by ID. In our sample domain ID = {id1, id2, …, id6000}.

In processes different products are manufactured or manipulated. Products can be divided in different portions based on their types or manipulation needs. The product portion is defined as follows:

> **Definition 1**: *Product portion* is a tuple $\langle$P-Name, C, ID-set, R$\rangle$ where P-Name is the name of product, C is the amount of the portion, ID-set is the set of object identities in the portion, and R is the ratio of the portion related the underlying total amount of the products.

In Definition 1 the ratio R is calculated by some application specific method based on e.g. the weight of the product portion related to the total weight of products in the underlying process, or used time related to total time needed in the process. For a product portion associated with other than objects, ID-set is empty and the portion is manipulated as a mass without interest on individual products.

In following list we have some example product portions related to harvesting:

- $\langle$PineSawLog, 350 m^3, {id1, …, id1000}, 0.60$\rangle$
- $\langle$PinePulpWood, 200 m^3, ∅, 0.30$\rangle$
- $\langle$HarvestingWaste, 20 ton, ∅, 0.10$\rangle$

The products PinePulpWood, and HarvestingWaste are manipulated as a mass, i.e. they do not contain object identities. Harvesting waste is manipulated as a product because it can be used to bioenergy. PineSawLog has a set of identities for logs ({id1, …, id1000}), or there are one thousand

logs. This portion has also the biggest ratio value (0.60) which means that it involves 60% of costs of the underlying process.

An attribute describes some information bound to a process. The attributes are divided into the three category based on their nature as follows:

1. *Input attribute* describes costs, used materials and other resources needed in a process. For example the used fuel is an input attribute. In the present approach the input attribute has a numeric value.
2. *Output attribute* describes other matters than products that a process produces. For example, a process may produce some tons of CO-gas. The output attribute has a numeric value.
3. *Info attribute* contains other data or documents associated with a process. The info attribute has a set value.

An attribute involves two values: one for the underlying process (ordinal value) and another for the previous production chain (cumulated value). The cumulated value is derived from previous processes based on the given rules. These are defined after the definitions of primitives needed for them. Until that (Definition 6) we use examples where ordinal and cumulated values are the same.

Next, the attribute is formally defined.

> **Definition 2**: *Attribute* is a tuple ⟨A-Name, T, V, W⟩ where A-Name is the name of attribute, T is the type of the attribute (∈ {input, output, info}), V is the ordinal value of the attribute, and W the cumulated value of the attribute.

In the following list we have some example attributes where the ordinal and cumulated values are same.

- ⟨Diesel, input, 100 liters, 100 liters⟩
- ⟨CO2, output, 300 kg, 300 kg⟩
- ⟨CompanyCode, info, {111}, {111}⟩

In the example “Diesel” is an input attribute describing that one hundred liters diesel is used (V value), “CO2” is an output attribute describing that the process caused 300 kilograms of carbon dioxide emission, and the company code 111 indicates a manufacturer associated with the process.

Now we are able to define the process node involving product portions and attributes in Definition 3.

> **Definition 3**. *Process node* (simply node) is a tuple ⟨Nid, N-type, P-set, A-set⟩ where Nid is the identity of the node, N-type is the type of the process, P-set is the set of product portions and A-set is the set of attributes associated with the node.

A harvesting node (Nid = 1) involving product portions and attributes is given below. From now on we mark by boldface numeric attribute values and ratios needed for cumulating forthcoming values in the supply chain.

⟨1, Harvesting,
{⟨PineSawLog, 350 m^3, {id1, …, id1000}, **0.60**⟩,
⟨PinePulpWood, 200 m^3, ∅, 0.30⟩,
⟨HarvestingWaste, 20 ton, ∅, 0.10⟩},
{⟨CarbonDioxide, output, 300 kg, **300 kg**⟩,
⟨Diesel, input, 100 liters, **100** liters⟩,
⟨CompanyCode, info, {111}, {111}⟩,
⟨Location, info, {lat 62.87 - lon 22.86}, {lat 62.87 - lon 22.86}⟩}⟩

In a supply chain, information on previous processes (nodes) is propagated to forthcoming processes. This information consists of object identities and the values of attributes. An *identity shift* determines those objects that are transferred from a process to another or a mapping among objects. There are five types of the identity shift. 1. *Equivalence* means that objects of the start and end nodes are the same. 2. *Subseting* means that some objects (but not all) are transferred. 3. *Superseting* means that all the objects are transferred but there also are objects from another process. 4. *Composition* means that several objects are composed to single objects. 5. *Division* means that single objects are auto-identification into several objects. Formally the identity shift is defined as follows:

Definition 4*: Identity shift* is a mapping *M* among identities or the sets consisting of them. The mapping *M* may be:

1. *equivalence*, where M:ID $\rightarrow$ ID and $dom(M) = rng(M)$
2. *subseting*, where M:ID $\rightarrow$ ID and $dom(M) \subset rng(M)$
3. *superseting*, where M:ID $\rightarrow$ ID and $dom(M) \supset rng(M)$
4. *division*, where M:ID $\rightarrow$ P(ID) and $\forall Y \in rng(M)\ \exists x \in dom(M)$: $|Y| > 1$
5. *composition*, where M:P(ID) $\rightarrow$ ID and $\forall y \in rng(M)\ \exists X \in dom(M)$: $|X| > 1$.

In Definition 4, cases from 1 to 3 maintain the object identities, whereas in cases 4 and 5 object identities are typically changed. In case 4, a single object is mapped to a set of object identities, whereas in 5 a set of object identities is mapped to a single object identity.

For propagating information represented as attributes among nodes, the notation of the derived attribute is used. Attribute value propagation rules are based on the types of attributes, i.e. propagation for input and output attributes requires calculation whereas info attributes are propagated by collecting all the data and documents for forthcoming processes. These rules are involved in the definition of the edge below.

A (shift) edge describes the connection between two nodes. An edge involves a set of derived attributes and a set of *sifted product portions*. A sifted product portion describes products that are shifted from a process to another. In order to maintain a product portion and the related objects, an identity shift is associated with shifted product portions. The edge is formally defined as follows:

Definition 5. *Shift edge* (simply edge) is a tuple $\langle N_S, N_E, SP, D\rangle$, where N_S and N_E are identities of the start and end nodes, SP is a *sifted product portion*, and D is a set of *derived attributes*.

- SP is a tuple $\langle$P-Name, C, *M*, Rp$\rangle$, such that there exists $\langle$P-Name,C',ID-set$_S$,_$\rangle \in N_S$.P-set and $\langle$_,_, ID-set$_E$,_$\rangle \in N_E$.P-set. The mapping *M* is an identity shift where objects in $dom(M)$ belong to ID-set$_S$ and objects in $rng(M)$ belong to ID-set$_E$. C is the amount of the sifted product portion and Rp = C/C' is the ratio of the sifted product portion.

- A derived attribute ($\in$ D) is a tuple $\langle$A.A-Name, A.T, DV$\rangle$ where A $\in N_S$.A-set, i.e. A.A-Name is the name of an attribute in N_S and A.T its type. DV is the value of the derived attribute. It is
  $$\begin{cases} \text{A.W, if A.T = info} \\ \text{A.W} \cdot \text{P.R} \cdot \text{SP.Rp where P} \in N_S\text{.P-set: P.P-Name = SP.P-Name, if A.T} \in \{\text{input, output}\} \end{cases}$$

In Definition 5 the sifted product portion is $\langle$P-Name, C, *M*, Rp$\rangle$ where Rp is the ratio of the portion. This is calculated such that the sifted amount C is divided by the original amount C' of the

start node. This ratio is used for calculating the value of derived attributes. A derived attribute is a 3-tuple where the first and second members possess the name and the type of the attribute inferred from the start node. The value of a derived attribute is cumulated as such from the start node if the type of the attribute is info. Otherwise, the original value is multiplied by the ratio of the original product portion (P.R), and by the ratio of the sifted product portion (SP.Rp). Below we introduce these aspects by examples.

In the running example (see Figure 2) there is a division edge between Nodes 1 (a harvesting node) and 3 (a sawing node). This edge describes that 10% of pine saw logs are selected to the sawing node. In the harvesting node (see above) the product portion of the logs possesses the ratio 0.6. This means that the values of input and output attributes must be multiplied by these ratios. Thus, the derived values of carbon dioxide and diesel are $300 \cdot 0.6 \cdot 0.1 = 18$ and $100 \cdot 0.6 \cdot 0.1 = 6$, respectively. The value of the info attribute is propagated as such. The type (division) of the edge means that single objects are divided into the sets of new objects. In the example objects with identities between id3000 and id3200 are balks whereas boards involve the identities between id4000 and id4800. A log is divided into one balk and four boards. In the object level this is described by an identity shift for one log object to a set of objects consisting of one balk and four boards. For example, the identity shift instance ⟨id1, {id3000, id4000, id4001, id4002, id4003}⟩ means that a log object (id1) is mapped to a balk (id3000) and boards (id4000, id4001, id4002, id4003). The related sample edge is represented as follows:

⟨1, 3, ⟨PineSawLog, 35 m^3, {⟨id1, {id3000, id4000, id4001, id4002, id4003}⟩, ⟨id2, {id3001, id4004, id4005, id4006, id4007}⟩, ⟨id3, {id3002, id4008, id4009, id4010, id4011}⟩, …,⟨id100, {…}⟩}, **0.1**⟩,
{⟨CarbonDioxide, output, **18** kg⟩,
⟨Diesel, input, **6** liters⟩,
⟨CompanyCode, info, {111}⟩}⟩

In a traceability graph there are two kinds of nodes based on their roles in the graph. Initial nodes have no predecessors, i.e. there is no edge to them. Other nodes possess at least one predecessor. This distinction is essential because attribute values are cumulated in other nodes than initial ones. In initial nodes an attribute has the same cumulated value as the ordinal value. For attributes in the other nodes, the cumulated value is derived from previous nodes via edges. If the underlying attribute is an info attribute, then the cumulated value is the set consisting of the ordinal value and values of derived attributes in incoming edges. Otherwise it is the sum of the value of the ordinal attribute and the corresponding values of derived attributes in incoming edges. Formally, the cumulated value is defined as follows:

Definition 6: Let A be an attribute in node N, V its ordinal value and $S = \{E|E.N_E = N\}$ the set of immediate incoming edges E to N, then the *cumulated value* W of A is

$$\begin{cases} V \cup \bigcup_{\langle \_,\_,D\rangle \in S} DV{:}\langle A,\_,DV\rangle \in D, \text{if } A.T = \text{info} \\ V + \sum_{\langle \_,\_,D\rangle \in S} DV{:}\langle A,\_,DV\rangle \in D, \text{if } A.T \in \{\text{input,output}\} \end{cases}$$

We demonstrate calculation of cumulated values below. Before that we define the traceability graph as follows:

Definition 7: Let N-Set be a set of process nodes and E-Set a set of shift edges, then ⟨N-Set, E-Set⟩ is a *traceability graph.*

Next we present the rest of our sample traceability graph. Node 1 and the edge between Nodes 1 and 3 are given above. In the example there are two harvesting nodes. In Node 1, logs have identities id1,…, id1000, whereas in Node 2 they have identities id1001,…, id1800. Ratios and the values of attributes differ in some extend from Node 1.

⟨2, Harvesting,
{⟨PinePulpWood, 300 m^3, ∅, 0.45⟩,
⟨PineSawLog, 300 m^3, {id1001,…, id1800}, **0.45**⟩,
⟨HarvestingWaste, 20 ton, ∅, 0.10⟩},
{⟨CO2, output, 270 kg, **270 kg**⟩,
⟨Diesel, input, 90 liters, **90** liters⟩,
⟨CompanyCode, info, {211}, {211}⟩,
⟨Location, info, {lat 65.21 - lon 21.36}, { lat 65.21 - lon 21.36}⟩}⟩

From Nodes 1 and 2 one hundred logs (30 cubic meters) are selected to the underlying sawing process. In the example these logs have identities id1 - id100 (edge from Node 1 to Node 3) and id1001 - id1100 (edge from Node 2 to Node 3).

⟨2, 3, ⟨PineSawLog, 30 m^3, {⟨id1001, {id3101, id4400, id4401, id4402, id4403}⟩, ..., ⟨id1100, {…}⟩}, **0.1**⟩,
{⟨CO2, output, **12.15** kg⟩,
⟨Diesel, input**, 4.05** liters⟩,
⟨CompanyCode, info, {211}⟩}⟩

In the edge from Node 2 to Node 3 the values of input and output attributes are calculated such that original attribute is multiplied with the ratio of the corresponding product portion (0.45) and the ratio of shifted product portion (0.1). For example the value of carbon dioxide (12.15) is achieved by the product $270 \cdot 0.45 \cdot 0.1$.

In the sawing node (3) there are three product portions: balk, board and sawing waste. The node has the attributes cumulated from the previous nodes and an additional attribute electric energy. The cumulated value of former input and output attributes is the sum of the values of derived attributes in incoming edges and the ordinal value of the attribute. For example the cumulated value of CO2 is 10 + 18 + 12.15 = 40.15. The ordinal value of the attribute is based on the used electric energy (0.5 kg per one kWh).

⟨3, Sawing,
{⟨balk, 1000 m, {id3000,…,id3200}, **0.50**⟩,
⟨board, 4000 m, {id4000,…,id4800}, **0.40**⟩,
⟨SawingWaste, 5 ton, ∅, 0.1⟩},
{⟨CO2, output, **10** kg, **40.15** kg⟩,
⟨Diesel, input, **0**, **10.05** liters⟩,
⟨CompanyCode, info, {311}, {111, 211, 311}⟩,
⟨ElectricEnergy, input, **20** kWh, **20** kWh⟩}⟩

Next the balks are transferred to a drying process. In the edge from Node 3 to Node 4, the objects maintain their identities and all the balks are transferred (shifted ratio value 1). The balks represent

50% of costs of the previous node, i.e. the values of input and output attributes are multiplied by 0.5.

⟨3, 4, ⟨balk, 1000 m, {⟨id, id⟩ | id ∈ {id3000,…,id3200}}, **1**⟩,
{⟨CO2, output, **20.075** kg⟩,
⟨Diesel, input, **5.025** liters⟩,
⟨CompanyCode, info, {111, 211, 311}⟩,
⟨ElectricEnergy, input, **10** kWh⟩}⟩

The balks are dried in Node 4 which produces 1000 kg carbon dioxide and takes 2000 kWh of electric energy.

⟨4, Drying,
{⟨balk, 1000 m, {id4000,…,id4800}, **1**⟩},
{⟨CO2, output, **1000** kg, **1020.075** kg⟩,
⟨Diesel, input, **0**, **5.025** liters⟩,
⟨CompanyCode, info, {311}, {111, 211, 311}⟩,
⟨ElectricEnergy, input, **2000** kWh, **2010** kWh⟩}⟩

Drying of boards is similar to the drying of balks. The following edge and node represent drying of boards.

⟨3, 5, ⟨board, 4000 m, {⟨id, id⟩ | id ∈ {id4000,…,id4800}}, **1**⟩,
{⟨CO2, output, **16.06** kg⟩,
⟨Diesel, input, **4.02** liters⟩,
⟨CompanyCode, info, {111, 211, 311}⟩,
⟨ElectricEnergy, input, **8** kWh⟩}⟩

⟨5, Drying,
{⟨Board, 4000 m, {id4000,…,id4800}, **1**⟩},
{⟨CO2, output, **800** kg, **816.06** kg⟩,
⟨Diesel, input, **0**, **4.02** liters⟩,
⟨CompanyCode, info, {311}, {111, 211, 311}⟩,
⟨ElectricEnergy, input, **1600** kWh, **1608** kWh⟩}⟩

Next dried boards are transferred to a gluing process. In this phase we assume that 10% of boards are disallowed, because for some reason they are flawed. This means that the ratio of the shifted product is 0.9. In gluing several boards (say ten) are composed to one glued beam. In the corresponding identity shift ten board objects are mapped to one glued beam object. For example in ⟨{id4000, ..., id4010}, id5000⟩ the identities id4000, ..., id4010 are board objects and id5000 is a glued beam object.

⟨5, 6, ⟨board, 2000 m, {⟨{id4000, ..., id4010}, id5000⟩,
⟨{id4010, ..., id4020}, id5001⟩, ... }, **0.9**⟩,
{⟨CO2, output, **734.454 kg**⟩,
⟨Diesel, input, **3.618** liters⟩,
⟨CompanyCode, info, {411}, {111, 211, 311, 411}⟩,
⟨ElectricEnergy, input, **1474.2** kWh⟩}⟩

Finally, in the gluing process glued beams are composed. There is also an additional attribute ‘glue’ that describes the amount of the used glue.

⟨6, Gluing,

{⟨GluedBeam, 2000 m, {id5000,…, id5500}, **0.9**⟩,
⟨WoodWaste, 0.1 ton, ∅, 0.10⟩},
{⟨CO2, output, **100** kg, **834.454** kg⟩,
⟨Diesel, input, 0, **3.618** liters⟩,
⟨CompanyCode, info, {311}, {111, 211, 311}⟩,
⟨ElectricEnergy, input, **200** kWh, **1647.2** kWh⟩,
⟨Glue, input, 100 kg, 100 kg⟩}⟩

Now we are able to calculate cumulated resources and emission for single glulam beams. For example emissions of single products or a set of products can be calculated.

7. ANALYZING TRACEABILITY GRAPH

In this section we introduce different analyzing possibilities based on our data-centric approach. We assume a traceability graph ⟨N-Set, E-Set⟩ notation behind the formalization.

7.1 Basic Functions for Analyzing Object Structure

Objects, their mutual structures and properties are embedded in the traceability graph. Next, we introduce how they can be derived from a traceability graph.

The predecessors of an object are in special interest because they determine materials and components needed for the object. Immediate predecessors are the nearest predecessors of the object (id) and they can be achieved by the function *i_predecessors* that is defined as follows:

i_predecessors (id) = {id' | E ∈ E-Set ∧ id ∈ *rng*(E.SP.M) ∧ id' ∈ *dom*(E.SP.M):

⟨id',id⟩ ∈ E.SP.M}

where E is an edge, SP its sifted product portion (presented as a tuple) an M the mapping among objects.

For example, *i_predecessors*(id5000) = {id4000,…,id4010}, i.e. glulam beam id5000 is glued from boards id4000-id4010.

All predecessors can be achieved recursively as follows:

$$predecessors(\mathrm{id}) = \begin{cases} S \cup \bigcup_{i \in S} predecessors(\mathrm{i}), \text{ where } S = i\_predecessors(\mathrm{id}), \text{ if } S \neq \varnothing \\ \varnothing, \text{ otherwise} \end{cases}$$

For example, *predecessors*(id5000) = {id4000,…,id4010, id1, id2,id3}, i.e. glulam beam id5000 is glued from boards id4000-id4010 which are sawed from logs id1, id2 and id3.

Successors of an object mean, for example, those objects for that the object has been raw material or component. The functions *i_successors* and *successors* yield the immediate and all successors, respectively.

i_successors(id) = {id' | E ∈ E-Set ∧ id ∈ *rng*(E.SP.M) ∧ id' ∈ *dom*(E.SP.M): ⟨id,id'⟩ ∈ E.SP.M}

$$successors(\mathrm{id}) = \begin{cases} S \cup \bigcup_{i \in S} successors(\mathrm{i}), \text{ where } S = i\_successors(\mathrm{id}), \text{ if } S \neq \varnothing \\ \varnothing, \text{ otherwise} \end{cases}$$

For example, *successor*(id1) = {id3000, id4000,…,id4003, id5000}, i.e. log with id1 is sawn to boards id4000-id4003 and beam id3000. The boards are used to manufacture the glulam beam with id5000.

An object may belong to several nodes in process chain. The function *node*(id) yields all the nodes that the object with id has associated with:

$$node(\text{id}) = \{N.Nid \mid N \in \text{N-set}: t \in N.\text{P-set} \wedge \text{id} \in t.\text{ID-set}\}$$

Among these nodes the fist and last ones are of special interest because the former determines the node where the object is created and the latter refers to the final state of the object. The functions *first_node*(id) and *last_node*(id) return them as follows.

$$first\_node(\text{id}) = N \in node(\text{id}) \wedge \text{id} \notin predecessors(\text{id})$$

$$last\_node(\text{id}) = N \in node(\text{id}) \wedge \text{id} \notin successors(\text{id})$$

Below we demonstrate the use of these functions.

7.2 Horizontal and Vertical Views

Traditionally views are predefined queries containing a derivation rule. In the context of the traceability graph the view means a sub graph determined by a rule for some purpose. In this paper we consider horizontal and vertical views. A horizontal view means an extensionally connected subgraph fired by object/objects. The connection at the extensional level means that objects of nodes in a view are connected through object mapping. Vertical view means that nodes of the same type are merged within a meta-node.

Among horizontal views we consider two basic cases: supply chain of an object and range distribution of an object. The supply chain of an object contains all preceding nodes that are extensionally connected (via object mapping) with the object. In terms of the function *predecessors* the supply chain of the object id is defined by the function *SC*(id) as follows:

SC(id) = ⟨N-Set', E-Set'⟩ such that

$$\begin{cases} \text{N-Set'} = \{N \in \text{N-set} \mid t \in N.\text{P-set} \wedge \text{id} \in t.\text{ID-set} \cup predecessors(\text{id})\} \\ \text{E-Set'} = \{E \in \text{E-set} \mid N1, N2 \in \text{N-set'} \wedge E.N_S = N1 \wedge E.N_E = N2\} \end{cases}$$

For example considering the glulam beam with id5000. The horizontal view contains the nodes and edges the glulam beam has participated in. For the sake of brevity we refer to nodes by their identities and edges to the participating node identities.

$$SC(\text{id5000}) = \langle \{N1, N3, N5, N6\}, \{\langle N1,N3\rangle, \langle N3,N5\rangle, \langle N5,N6\rangle\}\rangle$$

The distribution of an object means those nodes where the object or its part has been a participating in the TG. The distribution can be defined as follows:

dist(id) = ⟨N-Set', E-Set'⟩ such that

$$\begin{cases} \text{N-Set'} = \{N \in \text{N-set} \mid t \in N.\text{P-set} \wedge \text{id} \in t.\text{ID-set} \cup successors(\text{id})\} \\ \text{E-Set'} = \{E \in \text{E-set} \mid N1, N2 \in \text{N-set'} \wedge E.N_S = N1 \wedge E.N_E = N2\} \end{cases}$$

For example considering the log with id1. The horizontal view shows all the nodes and edges the log has participated in.

$$dist(\text{id1}) = \langle \{N1, N3, N4, N5, N6\}, \{\langle N1,N3\rangle, \langle N3,N4\rangle, \langle N3,N5\rangle, \langle N5,N6\rangle\}\rangle$$

Horizontal views can also be applied to tracing a set of objects having a specific property.

A vertical view is conceptually different of the horizontal ones. Namely it is basically defined on the intensional level - for a set of nodes of the same type merged onto one pseudo node. However, the selection criteria for nodes in the view can be extensional as well. For example nodes of objects having a property may be a selection criterion for a vertical view. The following definition for the vertical view may at the first view look complex but basically the its parts follow the similar way to deriving. Next we give the definition of vertical view as a whole and then introduce in detail.

Let N-Set' ($\subseteq$ N-Set) be a set of nodes of the same type, i.e. for each nodes N_1, $N_2 \in$ N-Set': N_1.N-type = N_2.N-type. The horizontal view node is represented as a tuple $\langle Nid_h, N\text{-type}, P\text{-set}_h, A\text{-set}_h\rangle$ where

- $Nid_h = \bigcup_{N \in N\text{-Set'}} \{N.Nid\}$
- N-type = N.N-type: $N \in$ N-Set'
- $P\text{-set}_h$ is a set of tuples each of the form $tp = \langle P\text{-name}_h, C_h, ID\text{-set}_h, R_h\rangle$ such that

$$\begin{cases} P\text{-name}_h \in \{t.P\text{-Name} \mid N \in N\text{-Set'}: t \in N.P\text{-set}\} \\ C_h = \sum_{N \in N\text{-Set'}} t.C: t \in N.P\text{-set} \wedge t.P\text{-name} = P\text{-name}_h \\ ID.set_h = \bigcup_{N \in N\text{-Set'}} t.ID\text{-set}: t \in N.P\text{-set} \wedge t.P\text{-name} = P\text{-name}_h \\ R_h = \dfrac{\sum_{N \in N\text{-Set'}} t.R : t \in N.P\text{-set} \wedge t.P\text{-name} = P\text{-name}_h}{|N\text{-Set'}|} \end{cases}$$

- $A\text{-set}_h$ is a set of tuples each of the form $ta = \langle A\text{-Name}_h, T_h, V_h, W_h\rangle$ such that

$$\begin{cases} A\text{-name}_h \in \{t.A\text{-Name} \mid N \in N\text{-set'}: t \in N.A\text{-set}\} \\ T_h = t.T \mid N \in N\text{-Set'} \wedge t \in N.A\text{-set} \wedge t.A\text{-name} = A\text{-name}_h \\ V_h = \begin{cases} \bigcup_{N \in N\text{-Set'}} t.V: t \in N.A\text{-set} \wedge t.A\text{-name} = A\text{-name}_h, \text{ if } T_h = \text{info} \\ \sum_{N \in N\text{-Set'}} t.V: t \in N.A\text{-set} \wedge t.A\text{-name} = A\text{-name}_h, \text{otherwise} \end{cases} \\ W_h = \begin{cases} \bigcup_{N \in N\text{-Set'}} t.W: t \in N.A\text{-set} \wedge t.A\text{-name} = A\text{-name}_h, \text{ if } T_h = \text{info} \\ \sum_{N \in N\text{-Set'}} t.W: t \in N.A\text{-set} \wedge t.A\text{-name} = A\text{-name}_h, \text{otherwise} \end{cases} \end{cases}$$

In the formula the set of node identities is Nid_h, N-type is the common type of the nodes, $P\text{-set}_h$ is the set of unionized product portions and $A\text{-set}_h$ is the set of merged attributes. In a product portion, $P\text{-name}_h$ is the name of a product, C_h is the total amount of the products and ID-set is the set of all object identities in the unionized product portion. The ratio R_h is calculated by dividing the corresponding amount of products by the number of the nodes participating in the view. A merge attribute consists of its name $A\text{-Name}_h$, the type T_h of the attribute, merged ordinal value V_h, and cumulated value W_h. The values are calculated by summing or unionizing the original values depending on the type of the attribute.

A horizontal view can be used to compact a workflow diagram. For example we could merge the harvesting nodes of our running example and get the total amount of harvesting in our sample supply chain.

⟨{1,2}, harvesting,

{⟨PineSawLog, 650 m^3, {id1, …, id1800}, 0.525⟩,
⟨PinePulpWood, 500 m^3, ∅, 0.375⟩,
⟨HarvestingWaste, 40 ton, ∅, 0.10⟩},
{⟨CarbonDioxide, output, 570 kg, 570 kg⟩,
⟨Diesel, input, 190 liters, 190 liters⟩,
⟨CompanyCode, info, {111, 211}, {111, 211}⟩,
⟨Location, info, {lat 62.87 - lon 22.86, lat 65.21 - lon 21.36}, {lat 62.87 - lon 22.86, lat 65.21 - lon 21.36⟩}⟩

In an enlarged example we could analyze the effect of different drying programs by merging the drying nodes based on the info-attribute that indicates the drying program.

In Figure 3 the enlarged example is represented. In the example the drying nodes #D12, #D13 and #D14 are merged to node #D12-14 and nodes #D15, #D16 and #D17 are merged to node #D15-17. The merging of traceability graph can be used to benchmark group of processes as described in the next section.

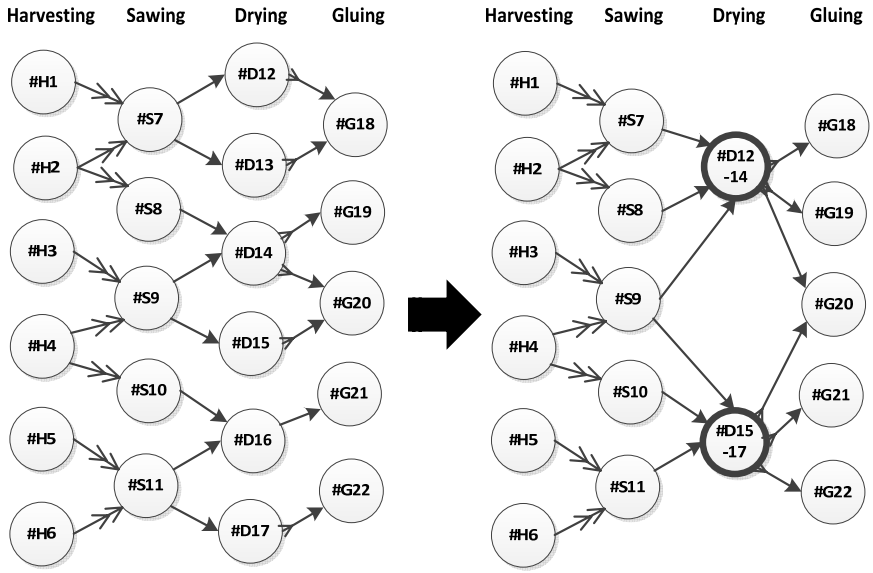

Figure 3. Merging Traceability Graph.

7.3 Examples

We demonstrate querying possibilities by sample queries that correspond to the uses cases described in Section 2.

Sample query 1: Calculating the item level carbon footprint

The carbon footprint of the object can be calculated by using the cumulated value of the CO2 attribute of the final node that the object has participated in. For example the carbon footprint of the glulam beam with id5000 is:

$$\frac{|\{\text{id5000}\}|}{|\{\text{t.ID-set}\}|} \cdot \text{t.R} \cdot \text{t'.W}$$

where t ∈ N.P-set: id5000 ∈ t.ID-set ∧ t' ∈ N.A-set: t.A-name = CO2 such that N = *last_node*(id)

Carbon footprint = (1 / 500) · 0.9 · 834.454 kg = **1,5020172 kg**

The formula can easily be extended to concern several objects when the dispensation would be multi-valued set.

Sample query 2: Origin of raw material

As described in Section 3 the information about the origin of raw material is becoming more and more important. Using the traceability graph we can trace the origin of the product. The origin of the gulam beam with id5000 is achieved by the location attribute in the harvesting nodes as follows:

$$t.V \mid t \in N.A\text{-set} \wedge t.A\text{-name} = \text{location: } N \in N\text{-set'} \wedge N \notin \mathit{rng}(E\text{-set'}) \text{ where } \langle N\text{-set'}, E\text{-set'}\rangle = SC(id5000)$$

In other words, CS determines the supply chain of the gulam beam and the V value is returned from the location attribute. The condition $N \notin \mathit{rng}$(E-set') ensures that the node is an initial process. The result is {lat 62.87 - lon 22.86}.

It is worth noting that above definition is based on a non-cumulated attribute. In cumulated attributes a corresponding query is simpler. CompanyCode is such an attribute. In our example customer is a corresponding cumulated attribute. For example the custody of the glulam beam with id5000 can easily achieved as follows:

$$t.W \mid t \in N.A\text{-set} \wedge t.A\text{-name} = \text{CompanyCode: } N = \mathit{last\_node}(id5000)$$

The result is {111, 211, 311}

Sample query 3: Recalling products

The traceability information can be used to recall products accurately and rapidly without needing to do the total recall throughout the supply chain. With information of the traceability graph we can find out all the products that some object is used as a component or raw material. The function *final_p* gives the final products in which an object (id) is used.

$$\mathit{final\_p}(id) = \{id' \in \mathit{successors}(id) \mid \mathit{last\_node}(id') \notin \mathit{rng}(E\text{-Set})\}$$

In our example, *final_prod*(id1) = {id3000, d5000}, i.e. beam id3000 and glulam beam id5000 are the recalled final products.

Sample query 4: Benchmarking

Benchmarking the processes between companies and manufacturing facilities enables to identify the processes with biggest environmental impact so that we improve the environmental performance of the supply chain. By using the vertical view we can calculate a key performance indicator for the nodes using the *bench()* functions.

- nodes define the set of nodes used in benchmarking
- prod_name defines the product portion used in benchmarking
- prop_name defines the attribute used to calculate the key performance indicator.
- group defines the attribute used to analyze the traceability graph.

bench(nodes, prod_name, prop_name, group) =

$$\{\langle x,y\rangle \mid x \in t_1.V: t_1 \in N.A\text{-set} \wedge t_1.A\text{-name} = \text{group} \wedge t_1.T = \text{info} \wedge y = \frac{\sum_{i \in S} i}{\sum_{j \in T} j} \text{ where}$$

$$\begin{cases} S = \{t_2.V \mid t_2 \in N.A\text{-set} \wedge t_2.A\text{-name} = \text{prop-name}\} \\ T = \{t_3.V \mid t_3 \in N.P\text{-set} \wedge t_3.P\text{-name} = \text{prod-name}\} \end{cases}$$

where $N \in$ nodes}

In our running example we can calculate the harvesting efficiency as follows:

bench({1,2}, PineSawLog, Diesel, CompanyCode)

The result {⟨111, 0.286⟩, ⟨211, 0.3⟩} presents how much diesel companies have used per cubic meter of saw logs. The comparison value for the company 111 is 0.286 and for the company 211 is 0.3.

8. DISCUSSION

The presented data-centric workflow model enables tracing, monitoring, analyzing and querying the properties of processes and their mutual relationships. The formal specification allows services to handle the products lifecycle data formally. To be able to share the life cycle data in real a world supply chain, we must:

1. ensure correspondence between logical objects with real life products of processes
2. have an infrastructure that enables multiple companies in a supply chain to share and use the information regarding products.

In tracing products, in addition to logical identities, they must be identified by physical identifiers. For physical products, various marking methods are in use: Imprinting, the finger print method, Laser marking, Label marking, Ink jet marking and transponder marking. In practice a physical identifier corresponds to object identity in database. This also gives natural interpretation for an object in the traceability graph. Below we consider an RFID (Radio Frequency IDentification) marking case related to our running example.

The modularity of a supply chain means that each actor is responsible for generating data from a part of the supply chain of the product and to share it with other stakeholders. To be able to share product related information in the complex supply chains the organizations have to agree on a common standard. One of the most promising is EPCglobal Architecture Framework[2] standards which are generally accepted methods for sharing product data in supply chains. They enable supply chain stakeholders to capture, store and share product related data. The EPCGlobal architecture includes EPC Information Services specification [17] that defines storing and sharing the traceability data that is created when a product marked with an RFID-tag passes an RFID-reader in a process in a supply chain. This event data normally contains unique identification code, location and time. By extending the EPC Information Service specification also environmental data can be included in event data. For example: 'At location X in time Y the object Z was observed with the environmental data [elementary flow #1, elementary flow #2]'

To be able to generate a total carbon footprint for a product the organizations must share environmental information of the products that were handled by them in their part of the supply chain. For example, in our running example, some organizations are responsible for harvesting the timber; sawmill companies handle the sawing and glued laminated timber manufactures are using the boards sawn in sawmills. All these stakeholders own a part of the final product's life cycle information. To be able to share environmental information each stakeholder must implement an

[2] http://www.epcglobalinc.org/standards/

EPC Information Service that implements the extension for environmental data. To be able to handle the object transformation (division or composition) in the supply chain, the stakeholder responsible for the transformation part of the supply chain is also responsible for aggregating the environmental information from up to that point. In other words, when a manufacturer is further processing products, the manufacturer is responsible for calling the EPC Information Services of a supplier and to add this (derived attribute in the traceability graph) information to the environmental information of the further processed product.

9. CONCLUSIONS

We have presented a data-centric workflow model, called the traceability graph. It integrates data-centric aspects of products and processes to traditional graph-based workflows. The approach supports attribute value propagation and aggregation in the supply chains. Input and output costs of processes can be allocated into products, which enable tracing and analyzing these coasts precisely. The model can be applied to single products as well as larger patches. Unlike existing methods the traceability graph enables precisely calculated input coasts (e.g. recourses) and output coasts (e.g. emissions and waste) of products and processes. So far these have been based on average values from a large set of processes.

Through the presented object transformation it is possible to model and manipulate the composition and division of objects in processes. We defined horizontal and vertical views. A horizontal view can represent a supply chain or the distribution of resources or components. In terms of the vertical view a traceability graph can be compacted by collecting similar processes together. We demonstrated analyzing possibilities of the traceability graph by several sample functions and use cases.

10. REFERENCES

[1] van der Aalst, W. M. P. The application of Petri nets to workflow management. *The J. of Circuits, Syst. and Computers*, 8(1), 1998, 21-66.

[2] van der Aalst, W. M. P. and Ter Hofstede, A. H. M. YAWL: Yet another workflow language. *Inf. Syst.*, 30(4), 2005, 245-275.

[3] van der Aalst, W. and van Hee, K. *Workflow Management, Models, Methods, and Systems*, The MIT Press, Cambridge, MA, 2002.

[4] Akram, A., Kewley, J. and Allan, R.: A Data Centric approach for Workflows. EDOC Workshops 2006, 10

[5] Attie, P., Singh, M., Sheth, A., and Rusinkiewicz, M. Specifying and enforcing intertask dependencies. In *Proceedings of 19th International Conference on Very Large Data Bases (VLDB'93)* (Dublin, Ireland, August 24-27, 1993),134-145.

[6] Bonner, A. J. Workflow, transactions and datalog. In *Proceedings of the Eighteenth ACM Symposium on Principles of Database System (PODS'99)* (Philadelphia, Pennsylvania, May 31-June 2, 1999), 294-305.

[7] Booch, G., Rumbaugh, J and Jacobson I. *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA, 1999.

[8] Caswell, N. S. and Nigam, A. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3), 2003, 428–445.

[9] Chen, P. The entity-relationship model—toward a unified view of data , *ACM Transaction on Database Systems*, 1(1), 1976, 9-36.

[10] Coad, P. and Yourdon, E. *Object-Oriented Analysis.* Prentice-Hall: 1990.

[11] Curcin, V. and Ghanem, M. Scientific workflow systems - can one size fit all? In *Proceedings Biomedical Engineering Conference (CIBEC 2008)*. (Cairo, 18-20 Dec. 2008), 1-9.

[12] Deutch, A., Hull, R., Patrizi, F. and Vianu, V. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory*, (St. Petersburg, March 23-26, 2009), 252-267.

[13] Ellis, C. A. and Nutt, G. J. Office Information Systems and Computer Science, *ACM Comp. Surv.*, 12(1), 1980, 27-60.

[14] EPCGlobal, EPC Information Services (EPCIS) Version 1.0.1 Specification, EPCglobal Ratified Standard, 2007.

[15] ISO 14020, Environmental labels and declarations -- General principles, ISO 14020:2000, International Organization for Standardization, Geneve, Switzerland, 2000.

[16] ISO 14040, *Environmental Management - Life Cycle Assessment - Principles and Framework*, ISO 14040:1997(e), International Organization for Standardization, Geneve, Switzerland, 1997.

[17] ISO 14064, Greenhouse gases -- Part 1: Specification with guidance at the organization level for quantification and reporting of greenhouse gas emissions and removals, ISO 14064-1:2006, International Organization for Standardization, Geneve, Switzerland, 2006.

[18] Junkkari, M. PSE: An Object-Oriented Representation for Modeling and Managing Part-of Relationships, *J. of Intelligent Inf. Syst.*, 25(2), 2005, 131-157.

[19] Junkkari, M. and Sirkka A. Using RFID for tracing cumulated resources and emissions in supply chain, *Int. J. Ad Hoc and Ubiquitous Computing*, 8(4), 2011, 220-229.

[20] Kifer, M., Lausen, G. and Wu, J. Logical foundations of object-oriented and frame-based languages, *J. of ACM*, 42(4), 1995, 741-843.

[21] Niemi, T., Junkkari, M., Järvelin, K. and Viita, S. Advanced query language for manipulating complex entities, *Inf. Processing and Management*, 40(6), 2004, 869-889.

[22] Puettmann, M. and Wilson, J. 2005. Gate-to-gate Life-Cycle Inventory of Glued Laminated Timbers Production. *Wood Fiber Sci.*, 37, 2005, 99-113.

[23] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall. 1991.

[24] Rumbaugh, J., Jacobson, I., & Booch, G. The Unified Modeling Language Reference Manual. Reading, MA: Addison-Wesley. 1999.

[25] Savnik, I., Tari, Z., And Mohoric, T. QAL: A query algebra of complex objects. *Data & Knowledge Engineering*, 30(1), 1999, 57–94.

[26] Sirkka, A. and Junkkari, M. Data management framework for monitoring and analyzing the environmental performance, In *Proceedings of INNOV 2010, ICGREEN 2010,* (Athens, July 29-31, 2010), 57-62.