Data-Centric Workflow Approach to Lifecycle Data Management

Marko Junkkari[1] and Antti Sirkka[2]
[1]School of Information Sciences, University of Tampere, Kanslerinrinne 1, Tampere, Finland
[2]Tieto Finland, Hatanpäänvaltatie 30, Tampere, Finland

Keywords: Data-Centric Workflow, Complex Objects, Physical Assembly, Data Model, Lifecycle Data Management, Traceability Graph.

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. This, in turns, requires explicit modeling and presentation how objects are divided and/or composed and how information is cumulated via these tasks. The traceability graph focuses on these issues. The traceability graph is formally defined by set theory that is an established and exact specification method.

1 INTRODUCTION

The data-centric approach for designing workflows is based on defining how the data is transferred between processes and how it is logically stored (Akram, Kewley and Allan, 2006; Caswell and Nigam, 2003). The approach examines how processes transform data and which entities send and receive the data. The main goal of data-centric workflows is to present the data sets in the workflows. In other words, data-centric workflows must involve an integrated data model for storing and manipulating data.

Complex objects consist of objects that in turn may consist of smaller objects (Motschnig-Pitrik and Kaasböll, 1999). This makes their manipulation demanding because both immediate and indirect components must be taken into account in analyzing information associated with complex objects. Physical assemblies are complex objects with the exclusivity constrain i.e. no component is shared (Junkkari, 2005). Workflows are usually focused on management of manufacturing of physical assemblies and they describe how physical assemblies are constructed. In manufacturing of physical assemblies, objects may be composed of components or an object may be divided into smaller objects. In manufacturing, information related to physical assemblies is cumulated via different types of processes. Division and composition require specific rules for information accumulation.

In the present study, we develop a data-centric workflow approach, called the traceability graph, focused on management of information allocation and accumulation in object transformation (division or composition). It supports dynamic data management techniques for data refinement such as aggregation, attribute value propagation, and derived attributes. Integration of object transformation with other dynamic data management issues gives an advanced approach to analyze data associated with processes, products the processes yield, and their components.

In our approach a (database) object may correspond to a real life entity, a set of real life entities or mass of material that can be physically identified thorough the part of a process chain in which it is participating. This means that physical entities of a patch are manipulated by a single object in a database if physical entities cannot be individually identified.

In data-centric workflows, the manipulation of complex objects requires specific features because objects may be changed into other objects in both the logical (in databases) and physical (real world)

Junkkari M. and Sirkka A..
Data-Centric Workflow Approach to Lifecycle Data Management.
DOI: 10.5220/0005001001160124
In *Proceedings of 3rd International Conference on Data Management Technologies and Applications* (DATA-2014), pages 116-124
ISBN: 978-989-758-035-2

levels. This means that object transformation and related data derivation rules 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. The above issues means that there must be rules for determining how resources, emissions and other information of processes are allocated to products in object transformation and how these are accumulated in a workflow.

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. The supply chain is a directed subgraph of a workflow diagram, 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, for calculating the 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 transferred by derived attributes.

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 (Puettmann and Wilson, 2005).

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.

The rest of the paper is organized as follows. In Section 2 we present a short survey on workflow models and traceability. The used mathematical notational conventions are given in Section 3 and the traceability graph is defined in Section 4. The analytic capabilities of the traceability graph are demonstrated in Section 5. In Section 6, we discuss implementation issues of the present model, and finally, the conclusions are given in Section 7.

2 RELATED WORKS

In workflows, materials, documents and other information are transferred from a process to another (van der Aalst and van Hee, 2002; Bonner, 1999). 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 (Booch, Rumbaugh and Jacobson, 1999), contain diagrams for dynamic aspects of programs (interaction diagrams) and modeling of workflows (activity diagram). The purpose of workflow model of UML is to map real world activities to the underlying software solution.

There are several 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 (van der Aalst, 1998; van der Aalst and van Hee, 2002). YAWL (van der Aalst and Ter Hofsted, 2005), Temporal logic (Attie et al., 1993), and Transaction Logic (Bonner, 1999) are other good representatives for formalizing workflows. These methods emphasize timing within processes and supply chains. From our perspective, timing is a secondary feature. Instead we emphasize handling the data aggregation and movement between processes.

Workflow models have also been investigated from the perspective of how they support different data-centric aspects (Curcin and Ghanem, 2008). The main aspects include concentrating on process functionality based on the data, i.e. each node has behavior instructions with regard to the data. Deutsch and others (2009) 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. The main difference between our study and their approach is that we address the explicit object sifting, transformation and derivation of the information through processes.

The present study focuses on tracing based on object transformation in a supply chain. In general, the importance of traceability has been noticed in software development (Gotel and Fincesteins, 1994,

Sommerwille, 2007; Podeswa, 2009; Bouillon et al., 2013; Breaux and Gordon, 2013; Rempel et al., 2013) as well as in manufacturing (Cheng and Simmons, 1994; Jansen-Vullers et al., 2003; 2004; Campos and Hardwick, 2006; Folinas et al., 2006). In software engineering it is essential to find the origin of a requirement when a system is changed (Gotel and Fincesteins, 1994; Sommerwille, 2007). In this context, traceability is seen as a property of a requirements specification that reflects the case of finding related requirements (Sommerwille, 2007). Requirements traceability refers to the ability to describe and follow the life of a requirement (Gotel and Fincesteins, 1994). In the context of UML based system specification, tracing between the behavioral and structural models has also noticed essential (Podeswa, 2009). This relates to our approach where aimed to integrated manipulation of data structures and workflows / data-flows.

In the context of requirements traceability and manufacturing, traceability is based on analyzing workflows or data-flows (with possible class and object diagrams) and there are different methods to represent traceability data. We, in turn, develop workflows for supporting information tracing through object transformations.

3 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.
- If t is a tuple and x its uniquely labeled member then t.x refers to x in t. For example if $t = \langle a, b, c \rangle$ 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 $\langle \_ \, x \, \_ \rangle$ 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 $\langle Node-id, P1, \ldots, Pn \rangle$, where Node-id is the identity of the node and P1, …, Pn are the properties associated with the node. For brevity, Node-id can be used to refer to the node. Thus, the notation Node-id.Pi refers to the property Pi in the node having the underlying identity.
- A directed edge is represented as a tuple $\langle Node-idS, Node-idE, P1, \ldots, Pn \rangle$, where Node-idS and Node-idE are the identities of the start and end nodes, respectively. P1 ,…,Pn are the properties associated with the edge.

4 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.

4.1 Node and Related Properties

Products which are identified physically and logically in the application domain are called *objects*. For logical identifying each object of interest possesses an identity. The set of possible identities of an application domain is denoted by ID.

In processes, different products are manufactured or manipulated. Products can be divided in different portions (patches) based on their types or

manipulation needs. The product portion is defined as follows:

Definition 1: *Product portion* is a tuple ⟨P-Name, C, ID-set, R⟩ 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.

An attribute describes some information bound to a process. The attributes are divided into the three categories 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 value of an info attribute is a set of strings or documents.

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 specific rules given later in Definition 6. So far we can assume that ordinal and cumulated values are the same. The attribute with its properties is defined as follows:

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.

Now, we are able to define a process node involving product portions and attributes.

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.

The nodes are connected to each other via edges. Next we define the primitives associated with edges.

4.2 Edge and Related Properties

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 tree types of the identity shift. 1. *Equivalence* means that objects of the start and end nodes are the same. 2. *Composition* means that several objects are composed to single objects. 3. *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. *division*, where M:ID $\rightarrow$ P(ID) and $\forall Y \in rng(M)\ \exists x \in dom(M)$: $|Y| > 1$
3. *composition*, where M:P(ID) $\rightarrow$ ID and $\forall y \in rng(M)\ \exists X \in dom(M)$: $|X| > 1$.

Case 1 maintains the object identities, whereas in cases 2 and 3 object identities are typically changed. In case 2, a single object is mapped to a set of object identities, whereas in 3 a set of object identities is mapped to a single object identity. Further, only some objects may be selected for refining or objects for refining may be collected from several previous processes.

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 ⟨P-Name, C, *M*, Rp⟩, such that there exists ⟨P-Name,C',ID-set$_S$,_⟩ ∈ N_S.P-set and ⟨_,_, ID-set$_E$,_⟩ ∈ 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 (∈ D) is a tuple ⟨A.A-Name, A.T, DV⟩ where A ∈ 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

$$\begin{cases} \text{A.W, if A.T = info} \\ \text{A.W} \cdot \text{P.R} \cdot \text{SP.Rp where P} \in N_S\text{.P-set: P.P-Name} = \text{SP.P-Name, if A.T} \in \{\text{input, output}\} \end{cases}$$

In Definition 5, the sifted product portion is ⟨P-Name, C, *M*, Rp⟩ 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).

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

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

The traceability graph involves the abovementioned primitives and it is defined 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*.

4.3 Sample System

In order to illustrate the traceability graph let us consider an example from a forest industry. 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 (glulam beams) from boards.

The first phase of production has two nodes describing the amount of daily harvesting. Harvesting has attributes Diesel (input), CO2 (output), Location (info) and Company (info). Harvesting nodes has tree product portions: Logs, Pulp wood and Harvesting waste. Only logs 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). Information associated with logs is first allocated to shifted product portions and then the products of sawing. Then information is reallocated to forthcoming product portions in the supply chain.

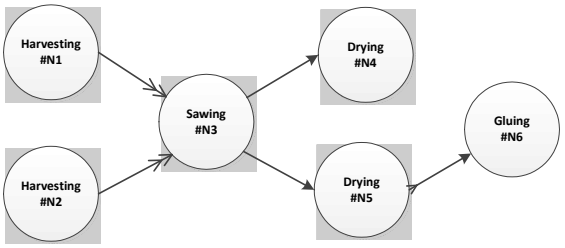

Figure 1: Sample Traceability Graph.

Between Sawing and Drying nodes the objects are not changed. This is illustrated with a plain arrow. 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.

In other words, the emissions, resources and other properties of the end products are accumulated through their supply chain. For example, calculating the CO2 footprint of a glulam beam all processes are taken into account and allocated among sub-products.

4.4 Schema

Definitions 1-7 determine explicitly the instance level of the traceability graph but they also implicitly involve schema level descriptions. The schema is a framework that determines the structure of instances with possible constraints in terms of which restrictions for nodes and edges can be stated.

At the schema level, a *node type* determines a set of similar nodes whereas an *edge type* determines a set of similar edges between similar nodes. A node type contains slots for a node id, node name, product portions and attributes, and possible constrains for them. Typical constraints are:

- Types of product portions
- Measures of product portions
- Types of attributes
- Measures of attributes

As an example, we define the harvesting node type.

{ N-type = 'Harvesting'
PP: ⟨Name: PineSawLog, Measure: cubic meter⟩
PP: ⟨Name: PinePulpWood, Measure: cubic meter⟩
PP: ⟨Name: HarvestingWaste, Measure: ton⟩
Attribute: ⟨Name:Diesel, Type: input, Measure:liter⟩
Attribute: ⟨Name:CO2, Type: output, Measure: kilograms⟩
Attribute: ⟨Name: Location, Type: info⟩
Attribute: ⟨Name: CompanyCode, Type: info⟩

An edge type contains slots for two node identities such that the type of the node can be specified. Further, there are slots for sifted product portions, identity mapping and derived attributes. The type (composition, division or equivalence) of mapping can be specified as well as the names, types and measures of the derived attributes.

For example, an edge type between harvesting and sawing nodes can be specified as follows:

{ Node type of N_S = 'Harvesting'
Node type of N_E = 'Sawing'
EdgeType = Division
PP: ⟨Name: PineSawLog, Measure: cubic meter⟩
D_attribute: ⟨Name: CO2, Type: output, Measure: kilograms⟩
D_attribute: ⟨Name: Diesel, Type: input, Measure: liter⟩
Attribute: ⟨Name: CompanyCode, Type: info⟩

The schema level description may also contain other constraints or derivation rules. For example, the value of the CO2 attribute (output) may be derived from the values of the Diesel attribute (input).

5 ANALYZING TRACEABILITY GRAPH

For analyzing structural aspects of the traceability graph functions different function can be defined. For the sake of limited space we only two sample function needed in the sample queries.
Successors of an object mean 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.

$i\_successors(\mathrm{id}) = \{\mathrm{id}' \in \mathrm{ID} \mid E \in \text{E-Set} \wedge \mathrm{id} \in rng(\text{E.SP.M}) \wedge \mathrm{id}' \in dom(\text{E.SP.M}): \langle \mathrm{id}, \mathrm{id}' \rangle \in \text{E.SP.M}\}$

$successors(\mathrm{id}) =$

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

For example, if id1 is the identity of a log then *successor*(id1) all products that is sawed from the log or has a component come down the log. 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:

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

Among these nodes the last one is of special interest because it refers to the final state of the object. The function *last_node*(id) returns it as follows.

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

Next we demonstrate querying possibilities of the traceability graph.

Sample Query 1: Calculating the Item Level Carbon Footprint

The carbon footprint of an 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, let us assume that object with id5000 is a glulam beam then the carbon footprint associated with it can be achieved as follows:

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

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

In other words t.R refers to the ratio R of the node t (a tuple) and t'.W refers to the cumulated value W of the attribute t' (a tuple).

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

Sample Query 2: 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. 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) =

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

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

According to Figure 1, #N1 and #N2 are identities of harvesting nodes of different companies The harvesting efficiency can be calculate as follows:

bench({N1,N2},PineSawLog,Diesel,CompanyCode)

The result a set of pairs that represents how much diesel companies have used per cubic meter of saw logs.

6 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 an object identity in the database. This also gives natural interpretation for an object in the traceability graph.

We have implemented a pilot project for using RFID (Radio Frequency IDentification) marking in forest industry for product level tracing (Junkkari and Sirkka, 2011**;** Björk et al. 2011). The product level marking needs marking of single products and their components. This is, of course, expensive and requires advanced infrastructure. Rougher marking reduces the costs but makes tracing vaguer. However, the price of RDIF techniques have decreased constantly which will enable item level tracing probably near in future.

We have implemented the traceability graph on the top of a relational database and investigated possibilities to OLAP (Online Analytic Processing) analysis (Sirkka and Junkkari, 2012). According to our experience the present formalisms give a good background to analyze traceability data multidimensionally. The product, process, time and elementary flow are possible dimensions for reporting recourses and emissions for benchmarking and the decision support.

7 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. The input and output costs of processes can be allocated into products, which enables tracing and analyzing these costs precisely. The model can be applied to single products as well as larger patches. Unlike existing methods the traceability graph enables precisely calculated input costs (e.g. recourses) and output costs (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 data allocation and aggregation in the composition and division of objects in processes.

REFERENCES

van der Aalst, W.M.P., 1998. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8, 1, 21-66.

van der Aalst, W.M.P., Ter Hofstede, A.H.M., 2005. YAWL: Yet another workflow language. *Information Systems*, 30, 4, 245-275.

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

Akram, A., Kewley, J., Allan, R., 2006. A Data Centric approach for Workflows. In *Proceedings of EDOC Workshops*, 10.

Attie, P., Singh, M., Sheth, A., Rusinkiewicz, M., 1993. Specifying and enforcing intertask dependencies. In *Proceedings of 19th International Conference on Very Large Data Bases (VLDB'93)* Morgan Kaufmann Publishers, 134-145.

Björk, A., Erlandsson, M., Häkli, J., Jaakkola, K., Nilsson, Å., Nummila, K., Puntanen, V., Sirkka, A. 2011. Monitoring environmental performance of the forestry supply chain using RFID, *Computers in Industry,* 62, 8-9, 830–841.

Bonner, A.J., 1999. Workflow, transactions and datalog. In *Proceedings of the Eighteenth ACM Symposium on Principles of Database System (PODS'99),* ACM Press, 294-305.

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

Podeswa, H., 2009. *UML for the IT Business Analyst: A Practical Guide to Requirements Gathering Using the Unified Modeling Language*, Course Technology, Boston, MA, USA.

Bouillon, E., Mäder, P., Philippow. I., 2013. A survey on usage scenarios for requirements traceability in practice. In *Proceeding of 20th International Working Conference on Requirements Engineering: Foundation for Software Quality,* LNCS 7830, 158-173.

Breaux, T.D., Gordon, D.G., 2013. Regulatory requirements traceability and analysis using semi-formal specifications. In *Proceeding of 20th International Working Conference on Requirements Engineering: Foundation for Software Quality,* LNCS 7830, 141-157

Campos, J.G., Hardwick, M., 2006. A traceability information model for CNC manufacturing. *Computer-Aided Design*, 38, 540-551.

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

Cheng, M.J., Simmons, J.E.L., 1994. Traceability in manufacturing systems. *International Journal of Operations & Production Management*, 14, 10, 4-16.

Curcin, V., Ghanem, M., 2008. Scientific workflow systems - can one size fit all? In *Proceedings Biomedical Engineering Conference (CIBEC 2008)*, 1-9.

Deutch, A., Hull, R., Patrizi, F., Vianu, V., 2009. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory*, 252-267.

Folinas, D., Manikas, I., Manos, B., 2006. Traceability data management for food chains. *British Food Journal*, 108, 8, 622-633.

Gotel, O.C.Z., Finkelstein, A.C.W., 1994. An analysis of the requirements traceability problem. In: *IEEE Proceedings of the First International Conference on Requirements Engineering*, 94–101.

Jansen-Vullers, M.H., van Dorp, C.A., Beulens, A.J.M., 2003. Managing traceability information in manufacture. *International Journal of Information Management*, 23, 395-413.

Jansen-Vullers, M.H., Wortmann, J.C., Beulens, A.J.M., 2004. Application of labels to trace material flows in multi-echelon supply chains. *Production Planning & Control: The Management of Operations*, 15, 3, 303-312.

Junkkari, M., 2005. PSE: An object-oriented representation for modeling and managing part-of relationships. *Journal of Intelligent Information Systems,* 25, 2, 131-157.

Junkkari, M., Sirkka, A., 2011. Using RFID for tracing cumulated resources and emissions in supply chain, *International J. of Ad Hoc and Ubiquitous Computing,* 8, 4, 220-229.

Motschnig-Pitrik, R., Kaasböll, J., 1999. Part-whole relationship categories and their application in object-oriented analysis. *IEEE Transactions on Knowledge and Data Engineering*, 11, 5, 779-797.

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

Rempel, P., Mäder, P., Kuschke, T., Philippow, I., 2013. Requirements traceability across organizational boundaries - a survey and taxonomy. In *Proceeding of 20th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS 7830, 125-140.

Sirkka, A., Junkkari, M., 2012. Multidimensional Analysis of Supply Chain Environmental Performance, Chapter 10 in *Sustainable ICTs and Management Systems for Green Computing*, 231-250.

Sommerwille, I., 2007. *Software Engineering*, 8th edition, Addison-Wesley, Harlow, England, 2007.