Traceability cost model

Miguel L. Pardal

Department of Computer Science and Engineering
Instituto Superior Técnico, Technical University of Lisbon
Av. Rovisco Pais 1,
1049-001 Lisboa, Portugal
Miguel.Pardal@ist.utl.pt

Abstract. This report presents a cost model developed to compare traceability information system architectures.

1 Introduction

Traceability information systems collect data statements such as: "Object O was seen at time T, place L." and "Object O was aggregated in pallet P".

These statements are stored along the product supply chain in different databases owned by different organizations.

The collected data is used to answer traceability queries [1], such as:

- Track/Recall query:
  What is current location of object O?
- Trace/Pedigree query:
  What is the history of object O?
- Aggregation/Bill-of-Materials (BoM) query:
  What are the components of object O?

There are several proposals of traceability systems. The considered systems are highlighted in figure 1. The classification criteria are *data integration* and *centralization*, as proposed by Do et al. [2].

The *data integration* criterion considers where data is physically stored. Data can be copied to specific locations (materialized integration) or referenced (virtual integration).

The *centralization* criterion considers the reliance on special nodes for data capture and query processing. In a centralized system there are nodes with special functions. In a decentralized system all nodes are functionally equivalent.

Combining the criteria, we have four distinct approaches to traceability: *metadata integration* (virtual, centralized), *data integration* (materialized, centralized), *unstructured peer-to-peer* (virtual, decentralized), and *structured peer-to-peer* (materialized, decentralized).

This report presents an analytical cost model developed to compare the four approaches, given a specific supply chain problem.

The model is based on the model developed to compare Theseos and TraceSphere by Murthy and Robson [3].

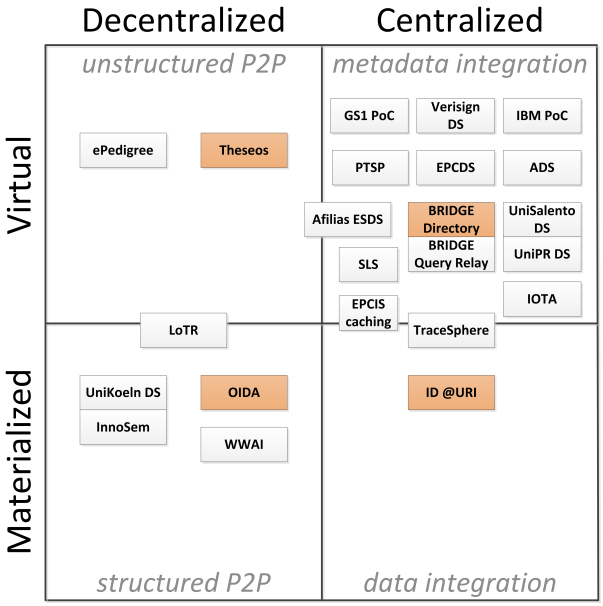

Fig. 1. Traceability system proposal classification.

2 Cost model

2.1 Chain modeling

We use directed acyclic graphs (DAGs) [4] to represent the supply chain.

A graph is defined by a set of vertices (V) and a set of edges (E), each connecting two vertices.

A DAG is a graph with directed edges, and without cycles. A DAG can be topologically sorted [4]. A DAG's in-degree is the number of incoming edges, and its out-degree is the number of outgoing edges. A vertex with in-degree 0 is a begin-vertex. A vertex with out-degree 0 is a end-vertex.

We consider item-defined DAGs and chain-defined DAGs.

Item DAG An item flowing in a supply chain defines a DAG. The vertices represent companies, and the edges represent the item flow between companies.

All vertices of an item DAG are connected by edges. Each vertex has, at most, in-degree 1 and, at most, out-degree 1. There is a single begin-vertex and a single end-vertex.

Figure 2 presents the item DAG for object A. Figure 3 presents the item DAG for object B.

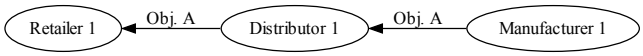

Fig. 2. Item-defined Directed-Acyclic Graph.

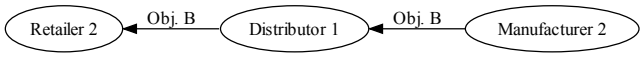

Fig. 3. Another item-defined Directed-Acyclic Graph.

Chain DAG Each item flowing during the traceability system's lifetime defines an item DAG of its own. At the end of the system's lifetime, we can define a chain DAG from the set of item DAGs.

The vertices of the chain DAG are defined by the union of item DAG vertices. The edges of the chain DAG are defined by the union of item DAG edges.

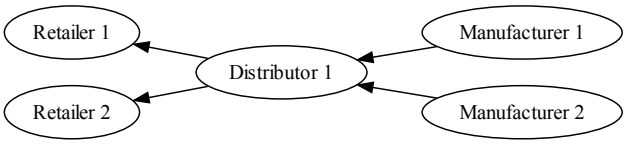

Fig. 4. Chain-defined Directed-Acyclic Graph.

An example chain DAG is represented in figure 4. It combines the item DAGs of objects A and B, presented earlier.

Traceability query formulation The traceability queries can be defined using the graph formulation.

For the Track/Recall query, given a chain DAG and an item, we want to find the vertex in the item DAG with the highest topological ordering. We want to find the vertex furthest ahead in the item DAG.

For the Trace/Pedigree query, given a chain DAG and an item, we want to recover the item DAG.

The Bill-of-Materials (BoM) query formulation requires aggregation.

Aggregation An aggregation is a whole-part association between two physical objects: the aggregate and the component.

The aggregation can be made for transportation purposes - the aggregate carries the component - or for manufacturing purposes - the component is assembled to the aggregate.

We can define the relationship $agg(a, c)$ meaning that object c (component) is aggregated to object a (aggregate).

We can also define the recursive relationship $aggr(a, c)$ meaning that either $agg(a, c)$ holds or that there is another object $a2$ such that both $agg(a, a2)$ and $aggr(a2, c)$ hold.

Aggregated-item DAG We can now define an aggregated-item DAG for object i where the vertices are companies and the edges are defined by the flow between companies of the item i or of an aggregate a such that $aggr(a, i)$.

An aggregated-item DAG has all vertices with out-degree of, at most, 1. For a transported-item, the in-degree of all vertices is, at most, 1. For an assembled-item, the in-degree of all vertices is greater or equal to 0, and is determined by the assembly.

Figure 5 presents the transported-item DAG for object A. Object A was aggregated to object C during the flow from Distributor 1 to Distributor 2.

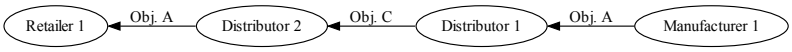

Fig. 5. Transported-item Directed-Acyclic Graph.

Figure 6 presents the assembled-item DAG for object C. Objects A and B were aggregated on object C on the Manufacturer 1 node.

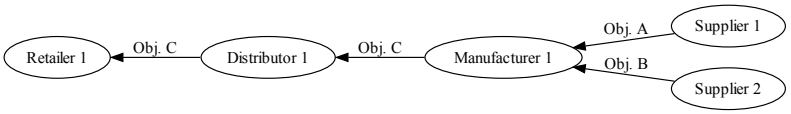

Fig. 6. Assembled-item Directed-Acyclic Graph.

Traceability query formulation with aggregation We can now extend the traceability query formulation to include aggregation.

For the Track/Recall query with aggregation, given a chain DAG and an item, we want to find the vertex in the transported-item DAG with the highest topological ordering.

For the Trace/Pedigree query with aggregation, given a chain DAG and an item, we want to retrieve the transported-item DAG defined by the item and by all of its containers.

For the Bill-of-Materials (BoM) query, given a chain DAG and an item, we want to retrieve the assembled-item DAG defined by the item and by all of its parts.

2.2 Parameters

The cost model's parameters are presented in table 1. The parameters aim to capture characteristics of the system, application, chain, and product.

Type	Name	Symbol	Unit	Default value
System	Bandwidth	β beta	bps	1 000 000 000
System	Processing speed	γ gamma	bps	1 000 000 000
System	Seek time	θ theta	s	0,001
Application	Message size	μ mu	bit	100 000
Application	Item record size	δ delta	bit	100 000
Chain	Nodes	n	vertex	3
Chain	Avg. item records per node	r	item record	1
Chain	Avg. length	z	vertex	3
Product	Avg. BoM depth	b	level	3
Product	Avg. children per BoM level	c	node	2

Table 1. Common parameters.

The number of nodes in the chain (n) is the number of vertices in the chain DAG.

The average number of item records per node (r) is the average of the number of events recorded on each node about a particular item. For Discovery Service purposes, a value of 1 is sufficient.

The average length of the chain DAG (z) is the average of the length of all item DAGs defined over the lifetime of the system.

The average depth (b) and children per node (c) parameters are used to characterize an average product BoM tree. The number of components for a level is given by c^b with b starting at 0 for the root node. The accummulated number of components is given by $\sum_{j=0}^{b} c^j$.

2.3 Assumptions

1. System-level assumptions
   (a) A system node is defined as a data store with processing capabilities owned by a specific company.
   (b) System parameters are the same for every node.

(c) Messages and received item records can be processed in main memory.
(d) All data stores are append-only.
(e) The time cost of accessing the data store to retrieve a record is independent of store size and independent of record size.
(f) Storing a record can be done asynchronously, so the time cost can be ignored.

2. Application-level assumptions
   (a) Application parameters are the same for every node.
   (b) All messages have the same size.
   (c) All item records have the same size.
   (d) The mapping from Object ID to Product Class is well-known.
   (e) The mapping from Product Class to Manufacturer is well-known.
   (f) The cost of locating the node where to issue a query (e.g. Manufacturer) is negligible.
3. Chain-level assumptions
   (a) Companies store data records about the items moving through the chain.
   (b) A supply chain can be represented by a Direct-Acyclic Graph (DAG).
   (c) A vertex (node) in the graph represents a company.
   (d) A directed edge in the graph represents a flow of items.
   (e) Nodes in the DAG can be topologically sorted [4] and each item moves in the chain according to this ordering i.e. from a begin-node towards an end-node.
   (f) An end-node is an end-destination of items (e.g. end-retailer in a distribution chain, end-manufacturer in a supply chain).
4. Product-level assumptions
   (a) A BoM for a product is represented as a tree. The root node is the product. The children nodes are components of the parent node.
   (b) A product is never disassembled.

2.4 Cost formulae

The time cost is measured in seconds. It is denoted by C.

The storage cost is measured in bits. It is denoted by S.

Time cost The cost of processing a message (C_{MP}) is the message size divided by the processing speed.

$$C_{MP} = \frac{\mu}{\gamma}$$

The cost of transferring a message over the network (C_{MT}) is the message size divided by the bandwidth.

$$C_{MT} = \frac{\mu}{\beta}$$

The cost of a lookup (C_L) is a constant.

$$C_L = \theta$$

The cost of a one-way message (C_M) is the sum of the cost of sending, transferring, and receiving the message.

$$C_M = 2 \cdot C_{MP} + C_{MT}$$

The cost of a message exchange (C_{MX}) is the sum of the cost of the request and of the response. A data lookup cost can added when response data has to be fetched.

$$C_{MX} = 2 \cdot C_M$$

The previous message cost definitions can be extended to include a payload of k item records.

$$C_{MP}(k) = \frac{\mu + k \cdot \delta}{\gamma}$$

$$C_{MT}(k) = \frac{\mu + k \cdot \delta}{\beta}$$

$$C_M(k) = 2 \cdot C_{MP}(k) + C_{MT}(k)$$

$$C_{MX}(k) = C_M(0) + C_M(k)$$

For a message exchange, k is the sum of item records in the request and response payloads.

Storage cost The storage cost of k item records ($S(k)$) is the multiplication of the item record size.

$$S(k) = k \cdot \delta$$

3 Metadata integration approach

In this approach the system is centralized to integrate metadata, namely, the location of data sources.

This approach is exemplified by the *BRIDGE Directory* [5] system. It is also considered to be the most closely related to the EPCglobal Discovery Service standard [6] under consideration.

MDI stands for MetaData Integration; DS for Discovery Service; EPCIS for EPC Information Services [7].

3.1 Additional assumptions

1. There is a single, well-known DS Search engine.
2. There are multiple DS instances.
3. There is one EPCIS instance for each company.
4. The DS Search is consulted once per company per product class to locate the DS instance.
5. The cost of DS Search is considered negligible because the result can be cached for the duration of the system's lifetime.
6. A DS publication is done only once for each item for each node.

3.2 Capture cost

When a company receives a product, r EPCIS records are created locally. A single publication is sent to the DS.

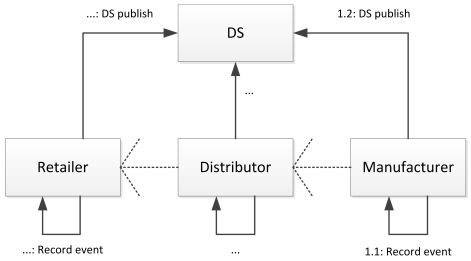

Fig. 7. Metadata integration capture.

The time cost of an item's data capture ($C_{mdi\,capture}$) is a message exchange with an item record for each company where the product passes.

$$C_{mdi\,capture} = z \cdot C_{MX}(1)$$

The storage cost of an item's data capture ($S_{mdi\,capture}$) is the sum of the cost of EPCIS records with the cost of the DS publication.

$$S_{mdi\,capture} = z \cdot \big(S(r) + S(1)\big)$$

3.3 Track query cost

To answer a track query the DS Root is contacted to locate the suitable DS instance.

A query is issued to the DS and the EPCIS location is returned in the response. The asker contacts the EPCIS with the most recent sighting of the object.

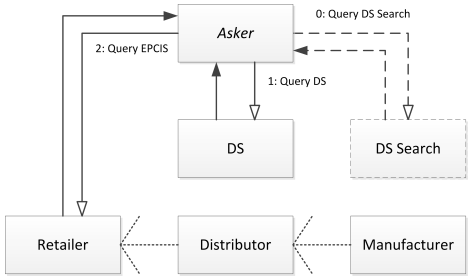

Fig. 8. Track query cost in the metadata integration approach.

The time cost of a track query ($C_{mdi\,track}$) is the sum of the cost of querying the DS with the cost of querying the EPCIS.

$$C_{mdi\,track} = 2 \cdot \left(C_{MX}(1) + C_L\right)$$

3.4 Trace query cost

A trace query is issued to the DS and a list of EPCIS locations are returned. The asker contacts each EPCIS for all records.

The time cost of a trace query ($C_{mdi\,trace}$) is the sum of the cost of querying the DS with the cost of querying each EPCIS.

$$C_{mdi\,trace} = \left(C_{MX}(z) + C_L\right) + z \cdot \left(C_{MX}(r) + C_L\right)$$

3.5 BoM query cost

A BoM query starts with a DS query to track the first observation of the product. Then, the asker contacts the EPCIS for all aggregation records. The BoM query continues recursively for each component: ask DS for first observation, ask EPCIS for aggregations.

The time cost of a BoM query ($C_{mdi\,BoM}$) is the sum of the cost of the DS track queries for each component in the BoM tree with the cost of the EPCIS aggregation queries.

$$C_{mdi\,BoM} = \sum_{i=0}^{b} \left(c^i \cdot \left(C_{MX}(1) + C_L\right)\right) + \sum_{i=0}^{b} \left(c^i \cdot \left(C_{MX}(c) + C_L\right)\right)$$

3.6 Parameter dependency analysis

The Metadata integration model's dependency to chain and product parameters is presented in table 2.

Formula/Parameter	n	r	z	b	c
$C_{mdi\,Capture}$			x		
$C_{mdi\,Track}$					
$C_{mdi\,Trace}$		x	x		
$C_{mdi\,BoM}$				x	x
$S_{mdi\,Capture}$		x	x		

Table 2. Metadata integration formulae dependency to chain and product parameters.

4 Data integration approach

In this approach the system is centralized to integrate all data.

This approach is exemplified by the *ID@URI* [8] system.

DI stands for Data Integration.

4.1 Capture cost

The capture records are sent to the Manufacturer for storage. No data is kept at the other nodes.

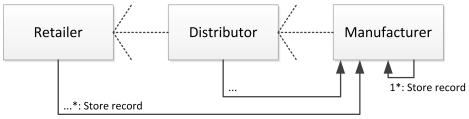

Fig. 9. Data integration capture.

The time cost of an item's data capture ($C_{di\,capture}$) is a message exchange with an item record for each company where the product passes.

$$C_{di\,capture} = (z-1) \cdot r \cdot C_{MX}(1)$$

The storage cost of an item's data capture ($S_{di\,capture}$) is the sum of the cost of all item records stored at the manufacturer.

$$S_{di\,capture} = z \cdot S(r)$$

4.2 Track query cost

The track query is sent directly to the Manufacturer.

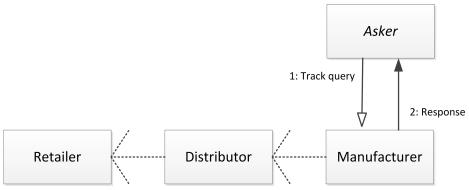

Fig. 10. Data integration track query.

The time cost of a track query ($C_{di\,track}$) is the cost of exchanging a message with the Manufacturer with a single result item record.

$$C_{di\,track} = C_{MX}(1) + C_L$$

4.3 Trace query cost

The trace query is also sent directly to the Manufacturer.

The time cost of a trace query ($C_{di\,trace}$) is the cost of exchanging a message with the Manufacturer with a list of result item records.

$$C_{di\,trace} = C_{MX}(z \cdot r) + C_L$$

4.4 BoM query cost

The BoM query is sent to the Manufacturer of each component.

The time cost of a BoM query ($C_{di\,BoM}$) is the cost of exchanging a message with the Manufacturer of each component to get the child component list.

$$C_{di\,BoM} = \sum_{i=1}^{b} c^i \cdot \left(C_{MX}(c) + C_L\right)$$

4.5 Parameter dependency analysis

The Data integration model's dependency to chain and product parameters is presented in table 3.

Formula/Parameter	n	r	z	b	c
$C_{di\,Capture}$		x	x		
$C_{di\,Track}$					
$C_{di\,Trace}$		x	x		
$C_{di\,BoM}$				x	x
$S_{di\,Capture}$		x	x		

Table 3. Data integration formulae dependency to chain and product parameters.

5 Unstructured P2P approach

In this approach the system is decentralized and data is distributed across the capture nodes.

This approach is exemplified by the *Theseos* [1] system.

UP2P stands for Unstructured Peer-to-Peer.

5.1 Additional assumptions

1. The cost of determining the address of the next node is a single lookup.

5.2 Capture cost

Data is captured along the supply chain and no communication is required.

There is no communication cost for an item's data capture ($C_{up2p\,capture}$).

$$C_{up2p\,capture} = 0$$

The storage cost of an item's data capture ($S_{up2p\,capture}$) is the sum of the cost of all item records stored along the chain.

$$S_{up2p\,capture} = z \cdot S(r)$$

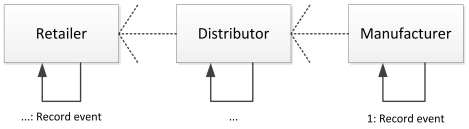

Fig. 11. Unstructured P2P capture.

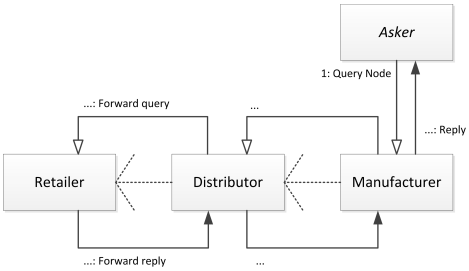

Fig. 12. Unstructured P2P track query.

5.3 Track query cost

The time cost of a track query is ($C_{up2p\,track}$) the sum of the cost of propagating the query to the node where the item is with the cost of propagating back the response. There is a lookup at each node required for the forwarding.

$$C_{up2p\,track} = z \cdot \big(C_M + C_L\big) + z \cdot C_M(1)$$

5.4 Trace query cost

The time cost of a trace query is ($C_{up2p\,trace}$) the sum of the cost of propagating the query to the node where the item is with the cost of propagating back the response with the accumulated trace records.

$$C_{up2p\,trace} = z \cdot \big(C_M + C_L\big) + \sum_{i=1}^{z} \big(C_M(i \cdot r)\big)$$

5.5 BoM query cost

The time cost of a BoM query ($C_{up2p\,BoM}$) is the sum of the cost of propagating the forward queries (one message for each component) with the cost of propagating back the response with the accumulated BoM tree.

$$C_{up2p\,BoM} = \sum_{i=1}^{b} \big(c^i \cdot (C_M + C_L)\big) + \sum_{i=1}^{b} \Big(c^i \cdot C_M\big(\sum_{j=i}^{b} c^{(b-j)}\big)\Big)$$

Formula/Parameter	n	r	z	b	c
$C_{up2p\,Capture}$					
$C_{up2p\,Track}$			x		
$C_{up2p\,Trace}$		x	x		
$C_{up2p\,BoM}$				x	x
$S_{up2p\,Capture}$		x	x		

Table 4. Unstructured P2P formulae dependency to chain and product parameters.

5.6 Parameter dependency analysis

The unstructured P2P model's dependency to chain and product parameters is presented in table 4.

6 Structured P2P approach

In this approach the system is decentralized and data is distributed according to a hashing algorithm.

This approach is exemplified by the *OIDA* [9] system.

SP2P stands for Structured Peer-to-Peer; DHT for Distributed Hash Table.

6.1 Additional assumptions

1. There is one DHT node for each chain node.
2. The hashing algorithm distributes the data evenly across the DHT nodes, using a unique item identifier.
3. The number of message hops to put or get a value is the logarithm of the number of DHT nodes [10].
4. The cost of determining the address of the next hop is a single lookup.
5. The nodes join the DHT once during the system's lifetime and never leave.
6. The cost of joining the DHT is negligible.
7. The item records are all kept on the same DHT node, indexed by the unique item identifier.

6.2 Capture cost

When a company receives a product, it contacts a known DHT node to store the data record. The hash algorithm determines the destination node and the destination is reached through a series of forwarding messages (hops).

The time cost of data capture ($C_{sp2p\,capture}$) is the sum of the cost of the message exchange with a DHT node with the cost of the message hops required to reach the node where data will be stored, and with the cost of the acknowledgement message.

$$C_{sp2p\,capture} = z \cdot r \cdot \Big(C_{MX}(1) + log(n) \cdot \big(C_M(1) + C_L \big) + C_M \Big)$$

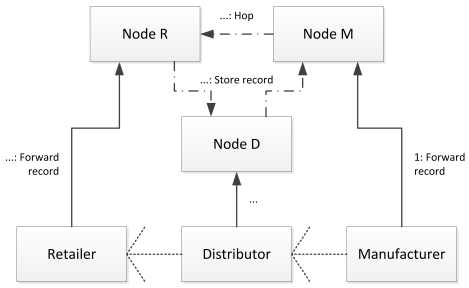

Fig. 13. Structured P2P capture.

The storage cost of an item's data capture ($S_{sp2p\,capture}$) is the sum of the cost of all item records stored in a single DHT node plus the hop pointers.

$$S_{sp2p\,capture} = z \cdot \Big(S(r) + S\big(log(n)\big) \Big)$$

6.3 Track query cost

When a track query is issued, a known DHT node is contacted to retrieve the data record. The hash algorithm determines the location node and it is reached through a series of forwarding messages (hops).

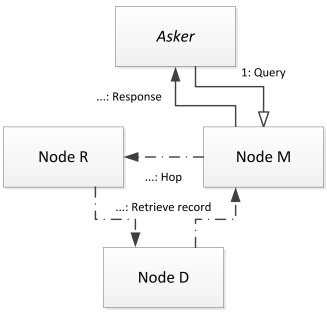

Fig. 14. Structured P2P track query.

The time cost of a track query ($C_{sp2p\,track}$) is the sum of the cost of the message exchange with a DHT node with the cost of the message hops required to reach the node where data is stored with the cost of the response message containing the single item record.

$$C_{sp2p\,track} = C_{MX}(1) + log(n) \cdot \big(C_M + C_L \big) + C_M(1)$$

6.4 Trace query cost

The time cost of a trace query ($C_{sp2p\,trace}$) is the sum of the cost of the message exchange with a DHT node with the cost of the message hops required to reach the node where data is stored, with the cost of the response message containing the item records.

$$C_{sp2p\,trace} = C_{MX}(z \cdot r) + log(n) \cdot \big(C_M + C_L\big) + C_M(z \cdot r)$$

6.5 BoM query cost

The time cost of a BoM query ($C_{sp2p\,BoM}$) is the cost of exchanging a message with the DHT for each component to get the child component list.

$$C_{sp2p\,BoM} = \sum_{i=1}^{b} \Big(c^i \cdot \big(C_{MX}(c) + log(n) \cdot (C_M + C_L) + C_M(c)\big)\Big)$$

6.6 Parameter dependency analysis

The structured P2P model's dependency to chain and product parameters is presented in table 4.

Formula/Parameter	n	r	z	b	c
$C_{sp2p\,Capture}$	x	x	x		
$C_{sp2p\,Track}$	x				
$C_{sp2p\,Trace}$	x	x	x		
$C_{sp2p\,BoM}$	x			x	x
$S_{sp2p\,Capture}$	x	x	x		

Table 5. Structured P2P formulae dependency to chain and product parameters.

Acknowledgments

Miguel L. Pardal is supported by a PhD fellowship from the Portuguese Foundation for Science and Technology FCT (SFRH/BD/45289/2008).

References

1. R. Agrawal, A. Cheung, K. Kailing, and S. Schonauer, "Towards traceability across sovereign, distributed RFID databases," in *International Database Engineering and Applications Symposium (IDEAS)*, 2006.
2. H.-H. Do, J. Anke, and G. Hackenbroich, "Architecture evaluation for distributed Auto-ID systems," in *Proc. 17th International Workshop on Database and Expert Systems Applications (DEXA)*, 2006, pp. 30–34.

3. K. Murthy and C. Robson, “A model-based comparative study of traceability systems,” in *Proceedings of the International Conference on Information Systems, Logistics and Supply Chain (ILS)*, 05 2008, madison, Wisconsin.
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.
5. J. J. Cantero, M. A. Guijarro, A. Plaza, G. Arrebola, and J. Baos, “A design for secure discovery services in the EPCglobal architecture,” in *Unique Radio Innovation for the 21st Century*, D. C. C. Ranasinghe, Q. Z. Z. Sheng, and S. Zeadally, Eds. Springer Berlin Heidelberg, 2010, pp. 183–201.
6. K. Traub, F. Armenio, H. Barthel, L. Burstein, P. Dietrich, J. Duker, J. Garrett, B. Hogan, O. Ryaboy, S. Sarma, J. Schmidt, K. Suen, and J. Williams, *The EPCglobal Architecture Framework 1.3*, GS1 Std., 03 2009. [Online]. Available: http://www.epcglobalinc.org/standards/architecture/
7. EPCglobal, *EPC Information Services (EPCIS) 1.0.1 Specification*, GS1 Std., 09 2007. [Online]. Available: http://www.epcglobalinc.org/standards/epcis
8. K. Framling, T. Ala-Risku, M. Karkkainen, and J. Holmstrom, “Design patterns for managing product life cycle information,” *Communications of the ACM*, vol. 50, no. 6, pp. 75–79, 2007. [Online]. Available: http://dialog.hut.fi/publications/CommACM2007_KF_forWeb.pdf
9. B. Fabian, “Implementing secure P2P-ONS,” in *IEEE International Conference on Communications (ICC)*, 2009, pp. 1–5.
10. H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Looking up data in p2p systems,” *Communications of the ACM*, vol. 46, pp. 43–48, 02 2003. [Online]. Available: http://doi.acm.org/10.1145/606272.606299