Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2008 Proceedings

European Conference on Information Systems (ECIS)

2008

EPCIS-based Decision Support for Assembly Networks

Christoph Goebel
Humboldt-University, Berlin, christoph.goebel@wiwi.hu-berlin.de

Christoph Tribowski
Humboldt-Universitat zu Berlin, christoph.tribowski@wiwi.hu-berlin.de

EPCIS-BASED DECISION SUPPORT FOR ASSEMBLY NETWORKS

Goebel, Christoph, Humboldt-Universität zu Berlin, Spandauer Straße 1, 10178 Berlin, Germany, christoph.goebel@wiwi.hu-berlin.de

Tribowski, Christoph, Humboldt-Universität zu Berlin, Spandauer Straße 1, 10178 Berlin, Germany, christoph.tribowski@wiwi.hu-berlin.de

Abstract

The coordination of assembly networks still represents a major challenge in today's business environment. We present a RFID-based inter-organizational system architecture which provides the technological basis for appropriate decision support. While mapping requirements in terms of information storage and exchange to technical system features, we consistently refer to current EPCglobal specifications. In contrast to the pull-based architecture proposed by EPCglobal, which is designed to retrieve and process historical data with a long lifetime, our system architecture follows a push approach. It allows for the propagation of relevant decision support information on past as well as future suppy chain events with short validity. The EPCglobal event data specification is extended to include the required context information. A common protocol layer which interconnects supply chain stages is described in detail. The use of the protocol layer in connection with standardized formats for event and context data supports the interoperability of information systems used in different organizations and facilitates the integration of event-based applications into enterprise architectures.

Keywords: Supply Chain Collaboration, RFID, Information System Integration, EPCglobal.

1 INTRODUCTION

Automatic identification and tracking of material movement by advanced data collection devices such as Radio Frequency Identification (RFID) is expected to enable more efficient supply chain management (Gaukler & Seifert 2007). Standardization of a number of components, which make up the architecture of event management solutions, is on the way. Apart from protocols and data schemas designed to serve the purpose of receiving, accumulating, filtering, and reporting events pertaining to particular electronic product codes (EPCs), a preliminary standard for storing and exchanging these events across the supply chain called EPC Information Services (EPCIS) has been specified by the global industry consortium EPCglobal (EPCglobal 2007a). While EPC formats and RFID reader protocols have come a long way, the EPCIS as well as the Object Name Service (ONS) and the EPCIS Discovery Services (EPCIS DS) are still in an early stage of development. EPCglobal offers the possibility to certify EPCIS implementations. Such certificates serve to guarantee that software applications being offered on the market adhere to the standard. IBM has been offering an EPCIS-compliant software component since December 2006 (Bacheldor 2007). Other software vendors are expected to follow.

Although the software industry is quick to offer products able to process EPC data, the development of value-generating business applications still lags behind. As long as real-world applications are rare, it is hard to justify the definition of a comprehensive standard. Little academic research on supply chain wide decision support systems based on auto ID technologies has been published so far. Chow et al. (2007) provide a schematic description of an inter-organizational information system based on RFID which provides visibility of the processes taking place at a third party logistics provider via a web front-end. Trappey et al. (2007) describe an intelligent agent system that among other things supports real-time surveillance of production progress. Although these authors provide interesting starting points for the realization of inter-organizational event-based applications, they do not go into technical details concerning the data formats and protocols required to realize these applications. Further research on inter-organizational decision support systems based on auto ID technology is thus needed. In particular, it has to be determined which architectures suit which business applications. Since standardization plays a significant role in the design of inter-organizational information systems, research on the appropriateness of the current standards proposed by EPCglobal is warranted.

Motivated by the knowledge gap identified above, we focus on three promising areas for further research:

- The specification of concrete business applications of event-based inter-organizational supply chain management systems
- The interoperability of the different components that need to be integrated in order to realize such systems
- The intra- as well as inter-organizational management of the EPC context data provided by different enterprise applications

We believe that without a specific requirements analysis, the degree of system interoperability cannot be assessed. The type of context data that needs to be managed and exchanged naturally depends on the application. Our approach thus consists of first putting the discussion about EPC-based material tracking into a concrete business context. To this end we describe the challenges involved in coordinating decentralized make-to-order assembly networks. Our choice of the business context and application example tries to be as simple and general as possible. Thereafter, we derive technical requirements that need to be addressed by the architectural design, in particular with respect to inter-organizational system interoperability. We present an approach to realize a two-layered inter-organizational event-based architecture. In contrast to the components proposed by EPCglobal, our architecture follows a push approach for the dissemination of event data. In section 4.1 we describe a

protocol layer providing all necessary communication primitives to interconnect the enterprise systems of several organizations.

Our main contributions are the following:

- We describe a relevant business application of event-based systems in a multi-organisational context.
- Business requirements are mapped to technical system features while consistently referring to current EPCglobal specifications.
- We specify a tentative protocol layer serving to integrate heterogeneous enterprise systems that exchange EPC context data in order to coordinate an assembly network.
- We identify strengths and weaknesses as well as guidance for the further development of the current EPCIS standard.

The paper is structured in the following way. Section 2 outlines the business application we focus on. Section 3 provides a short introduction to the current EPCglobal standard as far as it concerns our work. In section 4 the main ideas behind and some details of our proposed architecture are summarized. Section 5 concludes and outlines further reseach opportunities.

2 BUSINESS APPLICATION

Fierce competition and the resulting pressure to reduce costs while maintaining high customer satisfaction has drawn attention to possible ways to improve supply chain wide coordination. Collaboration in this context means that several independent organizations work together to achieve the common goal of supply chain wide cost reduction (Chopra & Meindl 2004, Simatupang & Sridharan 2005). Supply chain collaboration is a growing field of research, however most collaborative efforts have so far been focussing on the demand side (Waller et. al 1999, VICS 1998): Sharing information on historical or expected demand and planning production jointly can greatly reduce common supply chain inefficiencies caused by phenomena such as the bullwhip effect (Lee 1997). Although more advanced identification technology can help to increase downstream inventory accuracy (Atali 2006), it has often been argued that the many benefits of standardized auto ID technologies can be obtained from the ability to track items as they are moving through the supply chain (Gaukler 2005). Interestingly, short-term coordination of supply processes using upstream information sources has received little attention in the operations community to date (Chen 2003).

2.1 Event-based Supply Chain Management

The business application described in this paper aims at realizing the benefits of short-term supply coordination by sharing real-time order progress information among the participants of a supply chain. Auto ID technologies such as RFID are expected to provide more real time visibility of upstream supply chain stages and therefore play a pivotal role in building systems to support operational supply chain management. A recent concept in logistics management termed Supply Chain Event Management (SCEM) conceives upstream information as a stream of discrete events that can be used to identify exceptions and trigger alarms (Otto 2003). However, little has been published to date about the system architecture required to meet the requirements of SCEM and how they can be integrated into current enterprise system infrastructures. Günther et al. (2006) provide useful starting points for research in this area.

In order to optimize short-term operations, decision makers along the supply chain need to be informed about problems at upstream stages as well as their options to deal with a particular problem. The short-term actions available to steer supply vary according to individual supply chain characteristics. In long- and medium-haul transportation there oftentimes exist the possibility to choose among different transportation modes, e.g. sea, sea/air, and air. The picking process taking place in warehouses can be accelerated if needed, for instance by skipping certain quality assurance

processes. Capacity can be added to production processes, e.g. by increasing machine throughput or by extending shifts. Information on the available short-term control options is usually valid for a very short period of time only. Thus, any event management system designed to support operational supply chain management has to include a component capable of transmitting and offering up-to-date control options.

In supply chains providing highly complex products such as cars, the end product usually consists of thousands of parts being delivered to the Original Equipment Manufacturer (OEM) by different suppliers. Those in turn have their own suppliers who are not visible for the OEM because it maintains no direct business relationship with them. As cost pressure further increases, complex assembly networks will become even more dispersed due to specialization, short-term supply contracting and outsourcing of production and transportation functions. Furthermore, many manufacturing companies have introduced just-in-time production to minimize undedicated inventory along the supply chain. The only way to cope with the resulting increase of supply uncertainty is to acquire effective means to coordinate the flow of material on a real-time basis. The information systems used to provide the required visibility and decision support need to be highly flexible and easily deployable.

2.2 Formalization of Assembly Networks

To be able to analyze the problem of short-time management of assembly networks in a structured manner, we introduce the semantics of a simple formalization of such networks in the following. According to Chopra and Meindl (2004) the four drivers of supply chain management are facilities, inventories, transportation and information. The way that these drivers are applied determines the performance and operational cost of a supply chain. Each of the four drivers will be reflected in our formalization. According to our model, an assembly network consists of one or more supply chain organizations. A supply chain organization in turn consists of an arbitrary number of internal nodes which can either be an assembly process node, an inventory node or a transportation node. Internal nodes are connected by edges indicating the flow of material. Assembly and transportation processes always need to be decoupled by an inventory node. Furthermore, one inventory node always refers to one particular item type. The upstream end of the formal assembly network is marked by order book nodes. Each order book holds the production orders for a subsequent assembly node.

Figure 1 shows an exemplary assembly network consisting of four supply chain organizations forming a three-tiered assembly network. The network conforms to the rules states above. We will use this example throughout the paper to illustrate the working of our event-based architecture.

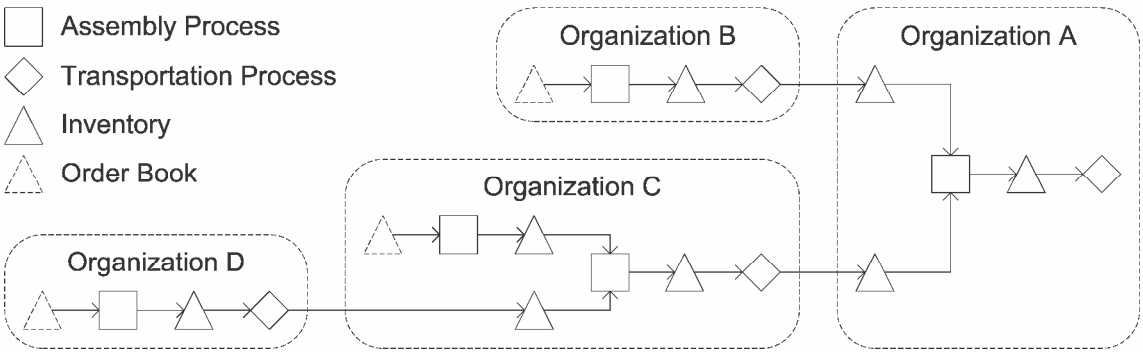

Figure 1. Example of formalized assembly network

The information required to optimize the coordination of an assembly network basically consists of schedules, i.e. events that are expected to take place at certain dates, the events actually taking place as material moves downstream, and the relevant control options. The purpose of the system architecture proposed in section 4 of this paper is to provide all nodes in the network with the technical means to share the required information in a decentralized way.

3 THE EPCGLOBAL SPECIFICATIONS

3.1 The EPCglobal Architecture Framework

The EPCglobal specifications describe a number of components required to realize a platform-independent system architecture for storage and retrieval of EPC-related data. EPCglobal has specified air interfaces for several tag types and reader protocols that serve to effectively read out EPC data in multi-tag, multi-reader environments. The Application Level Events (ALE) specification defines how to request event data from readers so that it can be used as input for higher level applications. These applications are supposed to work on the basis of the EPCIS which is described in more detail in section 3.2. To enable easy access to worldwide EPC-related data, EPCglobal has provided the specification for a hierarchical EPCIS lookup service. It is based on similar principles as the well-known Domain Name Service (DNS) used for the resolution of Internet host addresses. The lookup service used to access the EPCIS of the company which commissioned the EPC is specified in the Object Name Service (ONS) standard, whereas the EPCIS Discovery Services standard, which will be used to access the EPCIS of all companies that have information about the object, is still under development.

3.2 The EPCglobal EPCIS Specification

The EPCIS as conceived by EPCglobal consists of three components: A repository for event data and two interfaces serving to capture and query event data stored in this repository. Although EPCglobal does not provide an implementation of any of these components they have developed the specification of an extendable data model for supply chain events which is depicted in figure 2.

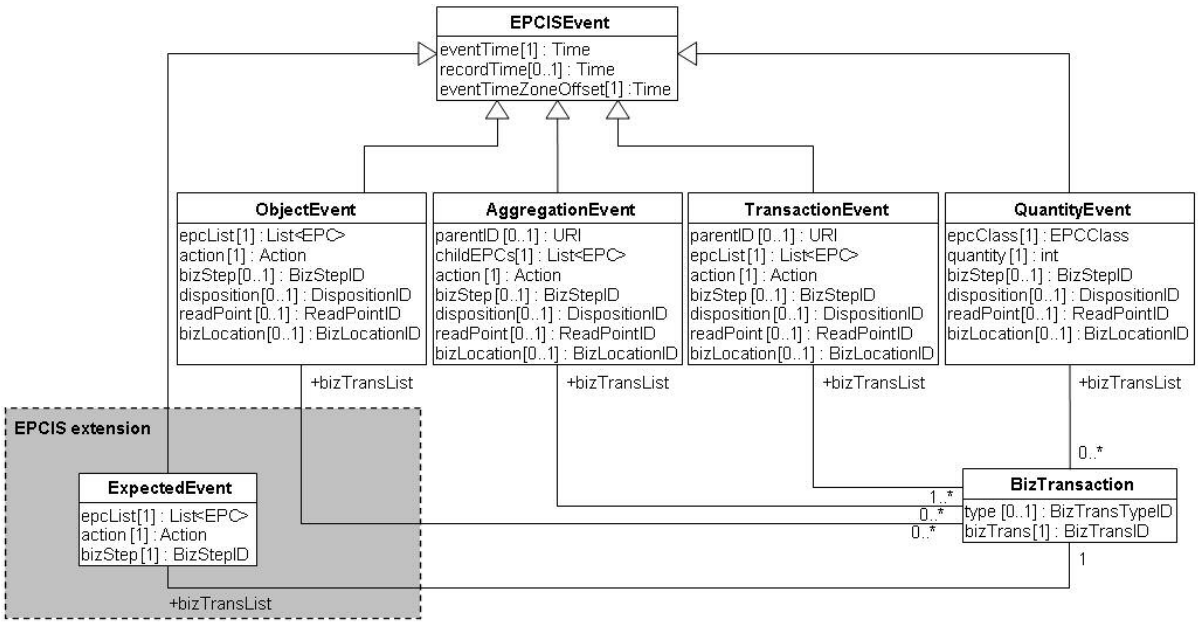

Figure 2. Extended EPCglobal Data Model

The capture interface receives formatted event data from the ALE and adds the required context data resulting in one of the event types shown in figure 2. The query interface allows applications to specify and manage queries for event data using a query control interface. Querying can be done on-

demand ('pull' approach) or by using the control interface to define and register standing queries that are executed periodically ('push' approach).[1]

An `EPCISEvent` can refer to anything happening in a supply chain that can be linked to a physical item and a discrete date. The different event types specified in the EPCglobal data model constitute the basic information handled by the EPCIS. Each event makes a statement about the *what*, *where*, *when*, and *why* of a supply chain event. The *what* dimension is specified by a list of EPCs identifying one or several physical objects and a list of so-called business transactions that these items are involved in. A business transaction can for instance be a production order. While the EPCs can be read from the transponder itself, the information of the corresponding business transaction needs to be retrieved from some information system. The *when* of an event is established by two time stamps specifying the time when the event was captured and the time it took place. The *where* dimension is specified by the two variables `readPoint` and `bizLocation` representing the place where the event was recorded and the place where the item is expected to be located after the occurance of the event. The value of `readPoint` is expected to be a technical ID (for instance derived from the reader infrastructure), whereas the business location provides the corresponding context information. The *why* refers to a business step (`bizStep`) and disposition ID (`disposition`) denoting the state of the physical item by the time its EPC is read and its disposition after that moment. For more detailed information on the different types of events present in figure 2 we refer the reader to Hribernik et al. (2007).

3.3 Application Requirements versus EPCglobal Architecture Features

Using a common data format like the one specified by EPCglobal to store EPC-related data is definitely valuable in providing interoperability between applications used in one organization as well as for the interchange of event data between organizations. However, the business application described in section 2 requires context data in the shape of expected events. Therefore we extended the EPCIS event data framework by the class `ExpectedEvent`.

According to EPCglobal, inter-organizational sharing of event data should be done using the EPCglobal core services, in particular the Object Name Service and the EPCIS Discovery Services. However, this results in a centralized query infrastructure in the hands of EPCglobal with the two mentioned services representing possible single points of failure. Furthermore, the context data required to make sense of the event data would have to be shared via an additional, unstandardized communication channel. We strive to specify an architecture that works in a decentralized manner and takes advatage of existing bilateral business relationships in the supply chain. We believe that the alternative system architecture proposed in the following suits the requirements of our business application better than the one envisioned by EPCglobal.

4 AN EPCIS-BASED SYSTEM ARCHITECTURE FOR SCEM

4.1 Protocol Layer

The entities communicating on the protocol layer are the nodes of the assembly network. Within our architecture these nodes represent communication hubs and controllers at the same time. Each node in the assembly network maintains a list of predecessors and a successor node for each type of product. Upstream messages are sent to some subset of predecessor nodes while downstream messages are sent to the successor node. Different product types have different bills of material, i.e. nodes would maintain at most one predecessor and successor list for each product type.

[1] In a very general sense both approaches follow the pull principle since in both cases data delivery is preceded by a more or less specific request.

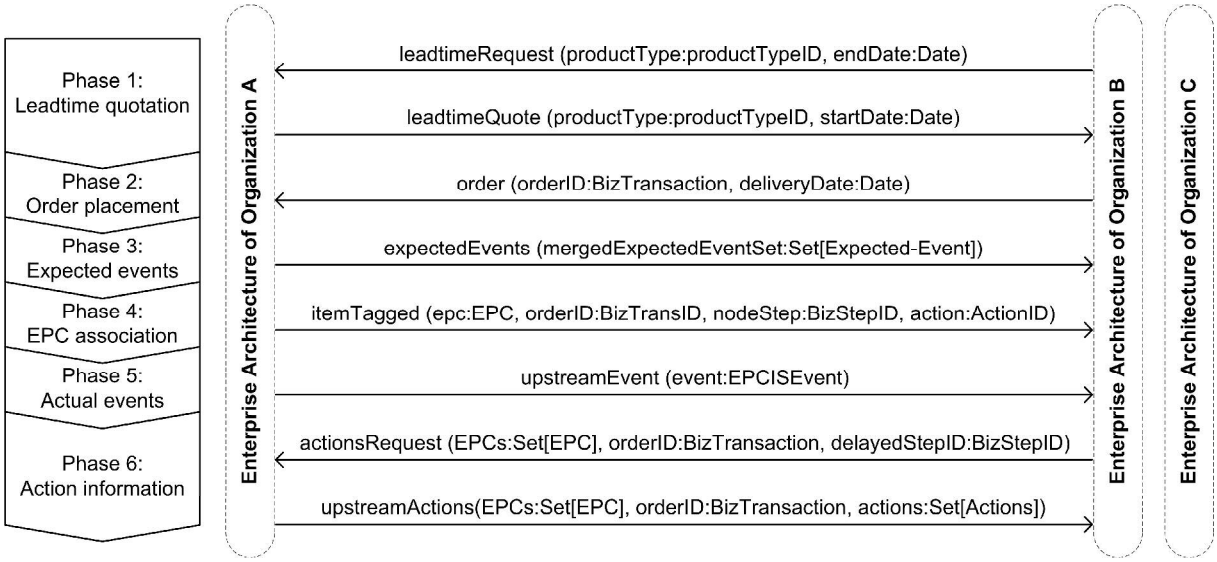

Figure 3. Protocol Layer for EPCIS-based SCEM

The communication taking place to coordinate the assembly of products is separated into six phases. Figure 3 presents an overview of the entire protocol. During phase one, lead times are quoted recursively. Each node implementing the protocol's communication primitives can query its upstream assembly network to find out if a certain delivery date can be met. Answering the query implies searching the assembly tree of a particular product type for the maximum lead time path. The answer consists of the date by which the order has to be issued at the root node in order to meet the requested delivery date. There are two message formats defined for this communication phase. An upstream message called `leadtimeRequest` containing the attributes `productType` and `endDate`, and a downstream message called `leadtimeQuote` containing the attribute `startDate`. Phase 1 is given as pseudo code below:

- `Upon reception of leadtimeRequest(productType:productTypeID, endDate:Date) by node i:`
  - `If node i is of type orderbook:`
    - `Set startDate to the earliest startDate incremented by node i's expected duration`
    - `Send leadtimeQuote(productType:productTypeID, startDate:Date) to involved successor node`
  - `Otherwise:`
    - `Send leadtimeRequest(productType:productTypeID, endDate:Date) to all corresponding predecessor nodes`
- `Upon reception of leadtimeQuote(productType:productTypeID, startDate:Date) by node i from all involved predecessor nodes:`
  - `Set startDate to the maximum startDate quoted by the predecessors incremented by i's expected duration`
  - `Send leadtimeQuote(productType:productTypeID, startDate:Date) to the involved successor node`

In our example, if the root node of organization A initiates the request, it will eventually end up with the expected lead time of the entire assembly process. From the value of `startDate` it can infer whether the order can be filled before the requested delivery date or not. If the quoted `startDate` has already passed, another query using a later delivery date can be initiated.

Order propagation constitutes the second communication phase. In our example, upon reception of a customer order, organization A initializes an upward information diffusion process of order data: A's root node sends an `order` message to its predecessors indicating that an order has been issued. The message contains a unique order ID as well as the scheduled date of delivery. Each order ID is represented by a `BizTransaction` object. The predecessor nodes propagate the order ID and the scheduled delivery date decremented by their respective expected process durations. The propagation process terminates when an order book node is reached. Thereafter, the order is stored in the order

book until the transmitted date coincides with the actual time. If this happens, the assembly process represented by the successor node is triggered. Phase 2 is given as pseudo code below:

- `Upon reception of order(orderID:BizTransaction, deliveryDate:Date) by node i from successor:`
  - `Set deliveryDate to the deliveryDate sent by the successor decremented by i's duration`
  - `Send order(orderID:BizTransaction, deliveryDate:Date) to all involved predecessor nodes`

The third communication phase consists of messages containing `ExpectedEvent` objects which are sent downstream. The `expectedEvents` messages used in this phase serve to let downstream nodes know when certain items are scheduled to enter and leave each node. The `ExpectedEvent` class which is used to store expected events represents an extension of the EPCglobal EPCIS standard. We embedded the event type `ExpectedEvent` as child of `EPCISEvent` (see figure 2). According to EPCglobal, adding a new event type implies updating the standard specification (EPCglobal 2007b). In our case the semantics of the `EPCISEvent` class would have to be adapted to include the possibility of events that have not yet taken place. Upon reception of an `expectedEvents` message concerning a particular order from all involved predecessors, a node remembers which events are scheduled to take place in the future by storing them in its local event repository or in the volatile storage of an SCEM application. Then it creates the events it expects to happen at its own entry and exit points. As indicated by figure 2, an expected event requires the attributes `epcList`, `action` and `BizStep`. By the time an `ExpectedEvent` object is created, there are no EPCs stored as values of its `epcList` attribute. If the object is created in response to an order, the action attribute is set to `ADD`. In case expected events need to be withdrawn, for instance because the corresponding order was cancelled, the action attribute is set to `DELETE`. The `BizStep` attribute is needed as a key to later match the expected with the actual events and is either set to the `BizStepID` of the entry or the exit point of the node. Newly created `ExpectedEvent` objects are combined with the received objects into a new set and sent downstream. Phase 3 is given as pseudo code below:

- `Upon reception of expectedEvents(expectedEventSet:Set[Expected-Event]) pertaining to a particular BizTransaction by node i from all involved predecessors:`
  - `Capture all ExpectedEvent objects contained in all expectedEventSets`
  - `Merge all expectedEventSets to obtain mergedExpectedEventSet`
  - `Create own ExpectedEvent objects and add them to mergedExpectedEventSet`
  - `Send expectedEvents(mergedExpectedEventSet:Set[Expected-Event]) to the involved successor node`

Phase 4 serves to complete the expected events created in phase 3 by the EPCs. This information is needed to identify pairs of expected and actual events which have to be compared in order to detect delays. We assume that EPCs are allocated at about the same time that physical objects are associated with an EPC. We believe that this is a reasonable assumption considering practical constraints such as RFID printers, which store fixed EPCs on passive tags. When a physical object gets associated with an EPC at some node, this node sends an `itemTagged` message to its successor. Each message of this type contains an EPC as well as the keys required to map the allocated or removed EPC to event entries at downstream nodes. Furthermore, it contains the type of action to be triggered by the message, i.e. either association or disassociation of EPC and expected event. When all stored `ExpectedEvent` objects have been enabled by adding one or several EPCs, each node possesses the information it needs to identify delays as upstream events of any type. Phase 4 is given as pseudo code below:

- `Upon reception of itemTagged(epc:EPC, orderID:BizTransID, nodeStep:BizStepID, action:ActionID) by node i from a predecessor:`
  - `If action is ADD:`
    - `Add EPC to all previously captured ExpectedEvents with the corresponding BizTransID and BizStepID`
  - `If action is DELETE:`
    - `Remove EPC from all previously captured ExpectedEvents with the corresponding BizTransID and BizStepID`

- Send itemTagged(epc:EPC, orderID:BizTransID, nodeStep:BizStepID, action:ActionID) to involved successor node

In phase 5 messages of type upstreamEvent are sent downstream to spread the news on actual events taking place upstream. Each of them carries an EPCISEvent object including the attached BizTransaction object referring to the order. It would be straightforward to only use the generated ObjectEvent objects in the protocol since they are created at all process steps. Phase 5 is given in pseudo code below:

- Upon capturing of event:EPCISEvent at node i:
  - Send upstreamEvent(event:EPCISEvent) to the involved successor node
- Upon reception of upstreamEvent(event:EPCISEvent) by node i from a predecessor:
  - Capture event
  - Send upstreamEvent(event:EPCISEvent) to the involved successor node

The final phase of the communication protocol allows each node to collect up-to-date action alternatives to make up for a particular delay. By comparing the dates of expected and actual events that have been captured during the previous communication phases a node can identify upstream delays. However, in order to exert control, the node requires information about which actions can currently be taken to influence the processing of a particular order. We refer to the path of nodes between the node which has caused the delay and the node that identifies it as the *action path* of a delay. Our protocol provides the opportunity to query the upstream network for these action paths. Any node can initiate such a query by sending a message of type actionsRequest to all its predecessors. This message contains three attributes: The EPCs that the delayed event refers to, the orderID of the delayed order and the BizStepID of the processing step at which the delay occurred. When an upstream node receives a message of type actionsRequest, it first checks whether it has stored an ExpectedEvent containing the EPCs in the message. If this is the case, it compares the BizStepID with the one of its exit point. If the two BizStepIDs are not equal, it forwards the message to all of its predecessors that are involved in the assembly process. Otherwise, the node which has caused the delay has been reached. This node then creates a message of the type upstreamActions containing information on all possible actions that can be taken to speed up order processing at its site. Then it forwards the message to its successor. If the successor is not the original requester, it adds its own ways to deal with delays concerning this order and sends the message to its own successor. This way the original requester ends up with a list of all up-to-date opportunities along the action path to speed up a particular order. Phase 6 is given in pseudo code below:

- Upon reception of actionsRequest(EPCs:Set[EPC], orderID:BizTransaction, delayedStepID:BizStepID) by node i from successor:
  - If an ExpectedEvent containing any of the EPCs in EPCs exists:
    - If delayedStepID equals exitStepID:
      - Retrieve available speedup actions for EPCs and orderID
      - Send message upstreamActions(actions:Set[Action]) to involved successor
    - Otherwise:
      - Send message actionsRequest(EPCs:Set[EPC], orderID:BizTransaction, delayedStepID:BizStepID) to all involved predecessors
- Upon reception of upstreamActions(EPCs:Set[EPC], orderID:BizTransaction, actions:Set[Action]) by node i from a predecessor:
  - If node i is the original requester:
    - Evaluate and trigger actions
  - Otherwise:
    - Retrieve available speedup actions corresponding with EPCs and orderID
    - Append these speedup actions to actions
    - Send message upstreamActions(EPCs:Set[EPC], orderID:BizTransaction, actions:Set[Actions]) to the involved successor

4.2 Application Layer

Having presented the protocol layer of our architecture in the previous section, we now turn to its application layer. The application layer consists of all enterprise systems that use the primitives of the protocol described in section 4.1.

Capacity in the shape of production slots, warehouse space or transportation capacity is usually managed by a corresponding information system which forms part of an Enterprise Resource Planning (ERP) solution. The quotation of process durations in phases 1 and 2 of the communication protocol described in section 4.1 thus depends on the input from those systems. The data that has to be provisioned to the protocol includes the quotable process start dates, the identifiers of entry and exit points of nodes in the assembly network, and the available speedup actions. Customer facing systems such as order management provide other inputs required for the working of the protocol. These inputs include the requested delivery date for an order, the type of product to be assembled and the allocated order IDs. Order management forms part of most standard ERP solutions.

EPCs are allocated by the EPC management of an organization. Each EPCglobal subscriber manages its own set of EPCs which contain the organization's unique General Manager Number. The subscriber organization is responsible for maintaining the numbers of Object Classes and Serial Numbers, which together with the organization's General Manager Number form the EPC (EPCglobal 2006). When a new EPC is created and attached to a physical object, this information needs to be published on the protocol layer.

The application layer component of the EPCIS-based decision support architecture required at each node consists of two components: A local EPCIS implementation and a SCEM application interfacing with users. Expected and actual events are stored in local EPC repositories which need to be accessed by the SCEM application to identify delays. Alternatively, SCEM applications can maintain their own event storage which eliminates the need for stand-alone EPC repositories as envisioned by EPCglobal. Furthermore, the SCEM application also needs to have direct access to the protocol layer in order to retrieve action paths.

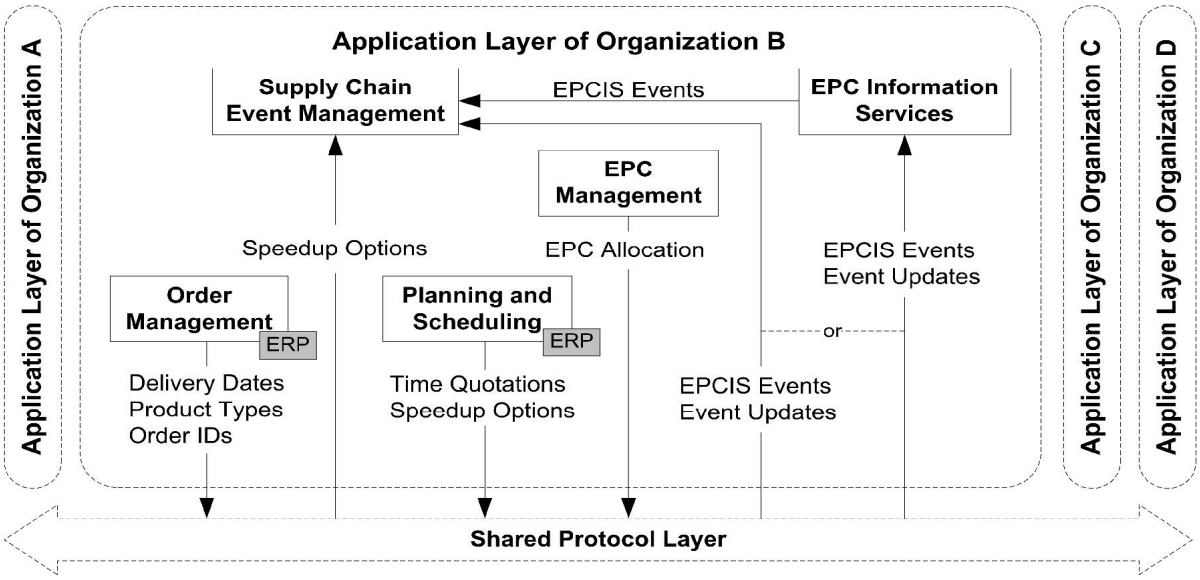

Figure 4. Two-layered EPCIS-based Architecture for SCEM

The purpose of the SCEM application is to provide the human decision maker, who is responsible for the coordination of the decentralized assembly network, with the means to monitor and control activities. The SCEM application could for instance allow for the individual specification of service levels that need to be met. These service levels would then be translated into tolerable delays so that action paths are only retrieved if delays reach a certain threshold. Furthermore, the functionality of the

SCEM application could include optimization routines that support the decision maker to pick an optimal action plan in each situation. Since there exists a potentially large number of possible actions that can be taken to speed up order processing on the action path and little time to decide which combination results in the least implementation cost, further IT support is definitely warranted.

Figure 4 depicts the general layout of the proposed architecture. The three components Supply Chain Event Management, EPC Management and EPC Information Services have to be added to the existing ERP solution in order to let an organization take advantage of the data being transmitted on the protocol layer.

5 CONCLUSIONS

We have presented a business application and a corresponding information system architecture providing the basis for the short-term coordination of a multi-organizational assembly network. The proposed system architecture was chosen for a number of reasons each of which can be attributed to the requirements of short-term decision support in dynamic multi-organizational business environments, in particular system interoperability and the inter-organizational management of EPC context data.

We have chosen to address the informational needs of our business application in order to derive concrete requirements. The concept we describe comes near to what is known as SCEM. SCEM has found general approval in practice since it addresses a number of pressing problems in today's competitive environment. To the best of our knowledge, the paper at hand represents the first attempt to suggest possible ways to realize SCEM applications based on the EPCglobal specifications while taking their specific requirements regarding interoperability and systems integration in multi-organizational environments into account.

From an operational point of view, an obvious shortcoming of the proposed architecture is that it does not address dynamic scheduling. Although it allows for order cancellation, the schedule of other orders encoded in the form of `ExpectedEvent` objects throughout the network cannot be changed in response to such an event. Certainly the protocol layer could be extended in order to deal with dynamic scheduling but it remains to be seen if such an extension is feasible in practical circumstances. Another limitation of the architecture results from its decentralized structure. Messages are forwarded along the supply chain, i.e. if an organization in the middle of the supply chain does not implement the protocol, our approach will not work. This problem could be solved by a third party willing to act as a trusted communication intermediary.

The proposed architecture supports interoperability in two ways: Firstly, due to its two-layered design there is no need to standardize any components on the application layer which facilitates the development and integration of the EPC/SCEM components. Secondly, one common way to describe event data and its context based on the EPCglobal event data specification is used both for intra- and inter-organizational communication.

Regarding the use of the EPCglobal specifications for SCEM applications we come to the following conclusions: Firstly, although EPCglobal provides a very good starting point for the format of event data, the specifications would need to be extended semantically to include the expectation of events. Secondly, an implementation of the EPCIS query interface is optional for the application described in this paper. Neither do we use the EPCglobal core services which are designed to search and retrieve EPC-related event data for our application. However, we acknowledge that the distribution of actual events (phase 5 of our protocol) could also be realized by querying the EPC Discovery Services for events related to all EPCs known to be involved in the processing of the order.

In our application up-to-date context data required by downstream nodes and organizations gets distributed without former request as soon as it becomes available. This approach disburdens downstream organizations from the need to maintain a comprehensive up-to-date internal process view

of other organizations. Furthermore, ex-ante knowledge of the organisational structure of the assembly network is not required, which represents a crucial advantage in today's dynamic and complex supply chains. Synchronization of data and context is assured by design since data and context are sent via the same communication channel.

We see a number of promising areas for further research on the proposed architecture. First of all, the architectural design needs to undergo further validation. Secondly, it needs to be extended to cope with dynamic rescheduling. The business logic of the actual decision support system, i.e. the development of algorithms used to optimize courses of action based on action path data, are a promising research direction. Still another issue that needs to be dealt with is authentication and security. The communication taking place on the protocol layer needs to be secured against malicious behaviour, e.g. by using dedicated public key infrastructures.

References

Atali, A., Lee, H. and Özer, Ö. (2006). If the Inventory Manager Knew: Value of Visibility and RFID under Imperfect Inventory Information. In Proceedings of the Manufacturing and Service Operations Management Conference, Evanston, Illinois.

Bacheldor, B. (2007). EPCglobal Readies EPCIS Certification Program. http://www.rfidjournal.com.

Chen, F. (2003). Information Sharing and Supply Chain Coordination. In T. de Kok and S. Graves (eds.): Handbook of Operations Research and Management Science: Supply Chain Management, North-Holland, Amsterdam, The Netherlands, p. 341-421.

Chopra, S. and Meindl, P. (2004). Supply Chain Management. Strategy, Planning, and Operations. Prentice-Hall, Upper Saddle River.

Chow, H. K. H., Choy, K. L., Lee, W. B. and Chan F. B. S. (2007). Integration of Web-based and RFID Technology in Visualizing Logistics Operations - A Case Study. Supply Chain Management: An International Journal, 12 (3), 221-234.

EPCglobal (2006), Tag Data Standard Version 1.3.1 Specification. http://www.epcglobalinc.org.

EPCglobal (2007a). EPC Information Services (EPCIS) Version 1.0 Specification. http://www.epcglobalinc.org.

EPCglobal (2007b). The EPCglobal Architecture Framework. http://www.epcglobalinc.org.

Gaukler, G. M. (2005). RFID in Supply Chain Management. Ph.D. dissertation, Stanford University.

Gaukler, G. M. and Seifert, R. W. (2007). Applications of RFID in Supply Chains. In H. Jung, F. F. Chen, and B. Jeong, (eds.): Trends in Supply Chain Design and Management: Technologies and Methodologies, Springer-Verlag London Ltd.

Günther, O., Ivantysynova, L., Teltzrow, M. and Ziekow, H. (2006). Kooperation in RFID-gestützten Wertschöpfungsnetzen. Industrie Management, 22 (3), 41-44.

Hribernik, K. A., Schnatmeyer, M., Plettner, A. and Thoben, K.-D. (2007). Application of the Electronic Product Code EPC to the Product Lifecycle of Electronic Products. Proceedings of the EU RFID Forum 2007, Brussels, Belgium.

Lee, H. L., Padmanabhan, V. and Whang, S. (1997). Information Distortion in a Supply Chain: The 'Bullwhip Effect'. Management Science, 43 (4), 546-558.

Otto, A. (2003). Supply Chain Event Management: Three Perspectives. International Journal of Logistics Management, 14 (2), 1-13.

Simatupang, T. M. and Sridharan, R. (2005). An Integrative Framework for Supply Chain Collaboration. The International Journal of Logistics Management, 16 (2), 257-274.

Trappey, A. J. C., Lu, T.-H. and Fu, L.-D. (2007). Development of an Intelligent Agent System for Collaborative Mold Production with RFID Technology. Robotics and Computer-Integrated Manufacturing, Article in Press.

Voluntary Interindustry Commerce Standards Association (VICS) (1998). Collaborative Planning, Forecasting and Replenishment. http://www.cpfr.org.

Waller, M., Johnson, E. M. and Davis, T. (1999). Vendor Managed Inventory in the Retail Supply Chain. Journal of Business Logistics, 20 (1), 183-203.