# Chapter 3
# EPCIS-Based Supply Chain Event Management

**Christoph Goebel, Sergei Evdokimov,
Christoph Tribowski, and Oliver Günther**

**Summary** The coordination of assembly networks still represents a major challenge in today's business environment. We present a Radio Frequency Identification (RFID)-based inter-organizational system architecture that 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 the standards specified by the international industry consortium EPCglobal. In contrast to the pull-based architecture proposed by EPCglobal that 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 and future events with short validity. The EPCglobal event data specification is extended to include the required context information. A common protocol layer that 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. Using analytical methods, we evaluate the pull- and push-based ar-

C. Goebel (✉)
International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, CA 94703, USA
e-mail: goebel@icsi.berkeley.edu

S. Evdokimov · C. Tribowski · O. Günther
Humboldt-Universität zu Berlin, Spandauer Straße 1, 10178 Berlin, Germany

S. Evdokimov
e-mail: evdokimov@wiwi.hu-berlin.de

C. Tribowski
e-mail: christoph.tribowski@wiwi.hu-berlin.de

O. Günther
e-mail: guenther@wiwi.hu-berlin.de

chitectures with respect to efficiency and reliability: the push-based architecture is shown to be particularly suitable for the realization of SCEM applications.

## 3.1 Introduction

Supply Chain Event Management (SCEM) systems are decision support systems that allow for monitoring, prioritizing, and reacting to events pertaining to the flow of goods in a supply chain (Otto 2003). A supply chain event can be any change of state with respect to the flow of goods, currency, or information. SCEM applications allow for the specification of rules, which can be applied to streams of events in order to identify those which are critical, i.e., events which call for immediate action in order to prevent financial loss. The business value of an SCEM application depends on the degree of supply chain visibility and the degree of freedom regarding possible ways to react to critical events; however, since the available options for all actions are as dynamic as the various states of supply chain operations, the logical design of SCEM systems can be very demanding.

Supply chain visibility refers to the amount of information available to the supply chain manager and can be characterized according to the following dimensions: detail, timeliness, and accuracy. Radio Frequency Identification (RFID) technology can be used for efficient tracking of materials as they move though the supply chain (Niederman et al. 2007). In contrast to the bar code, it allows for concurrent reading of item identification data without direct line of sight up to a certain read range (Michael and McCathie 2005). RFID is expected to increase supply chain visibility along all three dimensions mentioned. Since SCEM requires a high degree of supply chain visibility, the introduction of RFID into a supply chain could increase the business value of SCEM applications.

Standardization of a number of components that 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), the international industry consortium EPCglobal has specified Electronic Product Code Information Services (EPCIS) that should be responsible storing and exchanging these events across the supply chain (EPCglobal 2007). While EPC formats and RFID reader protocols have come a long way, EPCIS is still in an early stage of development.

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) provided a schematic description of an interorganizational information system based on RFID that provides visibility of the processes taking place at a third-party logistics provider via a web front-end. Trappey et al. (2009) described an intelligent agent system that, among other things, supports real-time surveillance of production progress. Although these authors have provided interesting starting points for the realization of interorganizational event-based applications, they do not go into technical details concerning the data formats and protocols required to realize these applications. Further research on interorganizational 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 interorganizational 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 interorganizational supply chain management systems.
- The interoperability of the different components that need to be integrated in order to realize such systems.
- The intra- and interorganizational management of the EPC context data provided by different enterprise applications.
- The requirements analysis of EPCglobal's architecture for the supply chain-wide exchange of EPC-related data for SCEM 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 interorganizational system interoperability. We present an approach to realize a two-layered interorganizational event-based architecture. In contrast to the components proposed by EPCglobal, our architecture follows a push approach for the dissemination of event data.

Our main contributions are the following:

- We describe a relevant business application of event-based systems in a multi-organizational context.
- Business requirements are mapped to technical system features while consistently referring to current EPCglobal specifications.
- We specify a tentative protocol layer that serves to integrate heterogeneous enterprise systems that exchange EPC context data.
- We develop quantitative evaluation criteria and compare the centralized EPCIS-based architecture proposed by EPCglobal with the decentralized EPCIS-based architecture proposed in the next section.

This work is structured in the following way. In Sect. 3.2, we introduce relevant components of the EPCglobal network. Section 3.3 outlines the business application we focus on. In Sect. 3.4, the main ideas behind and some details of our proposed architecture are presented. A comparison of the developed and the centralized architectures, the quantitative evaluation, and its results are presented in Sect. 3.5. In Sect. 3.6, we discuss our findings. Section 3.7 concludes this chapter and outlines further research opportunities.

## 3.2 EPCglobal Network

The accuracy and detail of RFID data are expected to open up new and more efficient ways to manage the supply chain and reduce many of the inefficiencies plaguing today's businesses such as incorrect deliveries, shrinkage, and counterfeiting. Since production and distribution of physical goods are seldom in the hands of one organization and efficiency gains in supply chains often emanate from centralized supply chain control, it makes sense in most cases to share selected RFID data among supply chain participants.

For the cross-company exchange of RFID reader events, the Auto-ID Center and the industry consortium EPCglobal have specified a stack of specifications which is currently one of the predominant standardization efforts of the RFID community (Floerkemeier et al. 2007). In this section, we will sketch the specifications of the EPCglobal network that are most relevant for this work.

### *3.2.1 EPCglobal Architecture Framework*

The EPCglobal architecture framework (EPCglobal 2009) describes a number of components required to realize a platform-independent system architecture for collecting, filtering, storing, and retrieving EPC-related data. EPCglobal does not define the system architecture that end users have to implement but defines interfaces that the components of end users' systems (hardware and software) may implement. These interfaces as well as these hardware and software roles are separated in Fig. 3.1.

EPCglobal has specified air interfaces for several tag types and reader protocols that serve to effectively read out EPC data in multitag, multireader environments.

The Application Level Events (ALE) specification defines how to request EPC data from readers so that it can be used as input for higher-level applications (EPCglobal 2008c).

The EPCIS Capturing Application has the context information about the data capturing process and supervises the lower elements in the EPCglobal architecture as described above. Its main task is to create an EPCIS event and store it in the EPCIS repository using the EPCIS Capture Interface.

To enable easy access to EPC-related data, EPCglobal described several dedicated services. The Object Name Service (ONS) standard specifies a hierarchical

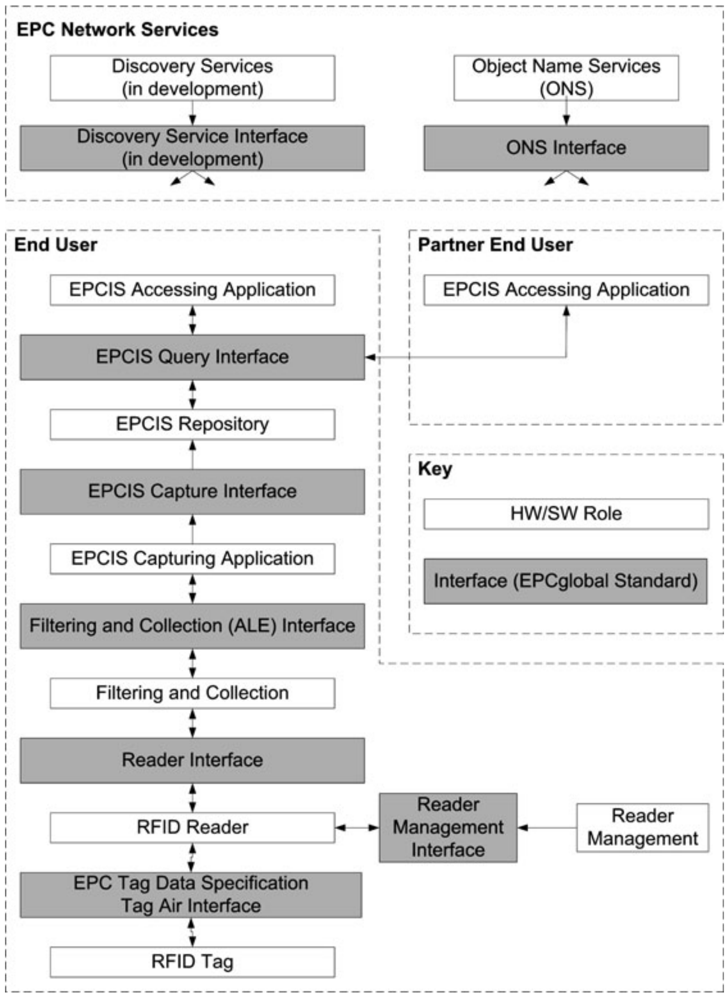
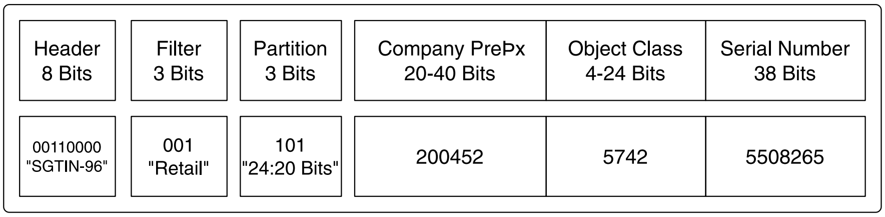

EPC Network Services
Discovery Services (in development)
Discovery Service Interface (in development)
Object Name Services (ONS)
ONS Interface
End User
EPCIS Accessing Application
EPCIS Query Interface
EPCIS Repository
EPCIS Capture Interface
EPCIS Capturing Application
Filtering and Collection (ALE) Interface
Filtering and Collection
Reader Interface
RFID Reader
EPC Tag Data Specification Tag Air Interface
RFID Tag
Reader Management Interface
Reader Management
Partner End User
EPCIS Accessing Application
Key
HW/SW Role
Interface (EPCglobal Standard)

**Fig. 3.1** EPCglobal architecture framework (based on EPCglobal 2009)

lookup service for locating service endpoints related to the EPC in question (EPCglobal 2008a). The EPC Discovery Services standard is currently under development, but the general description can be found in Kürschner et al. (2008). The main difference between the ONS and a Discovery Service is that EPC-related data provided by the services returned by the ONS is controlled by the entity that assigned the EPC to the item, while EPC-related data located using a Discovery Service can be controlled by any entity.

The EPC is described in more detail in Sect. 3.2.2, and EPCIS in Sect. 3.2.3.



| Header 8 Bits | Filter 3 Bits | Partition 3 Bits | Company PreÞx 20-40 Bits | Object Class 4-24 Bits | Serial Number 38 Bits |
|---|---|---|---|---|---|
| 00110000 "SGTIN-96" | 001 "Retail" | 101 "24:20 Bits" | 200452 | 5742 | 5508265 |

**Fig. 3.2** EPC identifier example (SGTIN-96)

## *3.2.2 Electronic Product Code*

The Electronic Product Code (EPC) specification describes a set of encoding schemas for universal identifiers that provide unique identities for physical objects (EPCglobal 2008b). The objects can be trade items (products), also logistical units, fixed assets, physical locations, etc.

The EPC schemas are designed to be backwards compatible with currently used GS1 codes. For example, the GS1 Serial Shipping Container Code (SSCC) for logistical units identifies unique objects. The corresponding EPC SSCC-96 encoding also identifies unique objects, and, therefore, there can be a one-to-one correspondence between SSCC and SSCC-96 identifiers. On the other hand, the Global Trade Item Number (GTIN)—the successor of the European Article Number (EAN)—identifies a category of objects, while the corresponding EPC Serialized GTIN (SGTIN-96) encoding allows identifying individual items by augmenting the GTIN with a serial number part.

Each EPCglobal subscriber manages its own range of EPCs contained within the organization's Company Prefix. The subscriber organization is responsible for maintaining the numbers of Object Classes and Serial Numbers which, together with the organization's Company Prefix, form the EPC (EPCglobal 2008b).[1] An example of such an EPC identifier encoded in SGTIN-96 format is displayed in Fig. 3.2.

## *3.2.3 EPC Information Services*

The EPCIS, as conceived by EPCglobal, consists of three components: a repository for event data and two interfaces that serve 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 for an extendible data model for supply chain events as depicted in Fig. 3.3.

The capture interface receives formatted event data from the ALE and adds the required context data, resulting in one of the event types shown. The query interface

[1]The only exception is DoD-96 encoding that includes two parts and is used for shipping goods to the US Department of Defense.

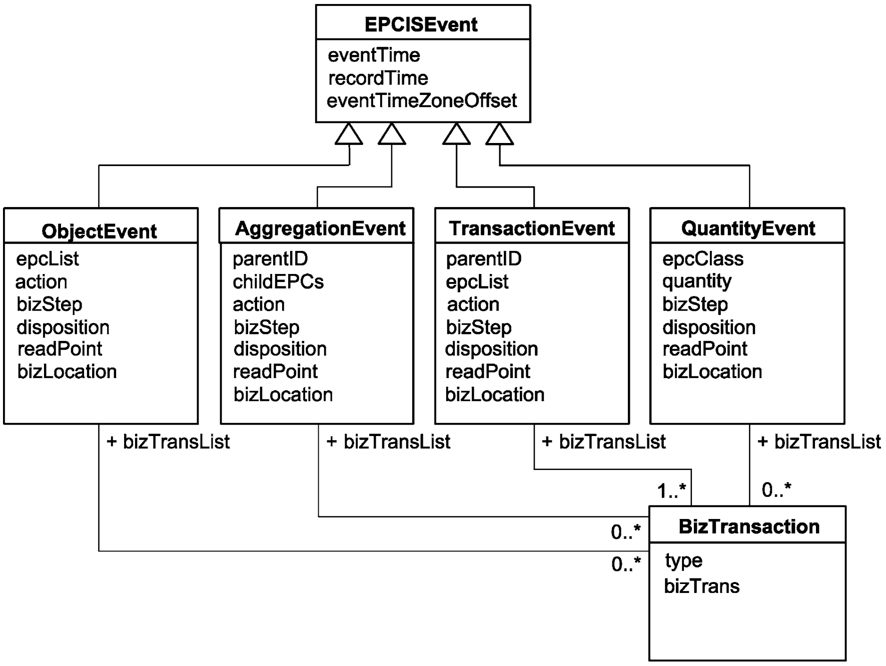


**Fig. 3.3** EPCglobal's EPCIS data model

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).

An EPCIS event can refer to anything happening in a supply chain that can be linked to a physical item and a discrete date. 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 that identify 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.
- The *when* of an event is established by two time stamps that specify the time the event happened and when it was captured.
- The *where* dimension is specified by the two variables, readPoint and bizLocation. The readPoint value is expected to be some technical ID, whereas the business location provides the corresponding context information.
- The *why* refers to a business step (bizStep) and disposition ID (disposition) that denote the state of the physical item by the time its EPC is read and its disposition after that moment.

The EPCIS data model defines four event types. An `ObjectEvent` captures information about an event that pertains to one or several physical objects identified by EPCs. Its mandatory attributes are the event and record times inherited from `EPCISEvent`; a list of EPCs; and an action attribute that can have three values: `ADD`, `OBSERVE`, and `DELETE`. If the value of the action attribute is set to `ADD`, it means that the EPCs were associated with the physical object for the first time. `OBSERVE` means that the object has been observed, whereas `DELETE` signifies that the EPCs listed in this event were decommissioned as part of the event. All other attributes of the object represent optional context information and have to be provided by other enterprise applications before the event gets stored in the database.

An `AggregationEvent` describes events that pertain to objects that have been physically aggregated (e.g., products in a box). The action attribute uses the same semantics: `ADD` signifies that the aggregation has been observed for the first time in this event, whereas `DELETE` means that it has been decommissioned, i.e., child tags are no longer associated to a parent tag but are still in existence.

The class `QuantityEvent` represents events that take place with respect to a specified quantity of some type of objects. This event could be captured, for instance, when the inventory level needs to be reported.

Depending on the value of its action attribute, a `TransactionEvent` describes the association or disassociation of physical objects to one or more business transactions. Its structure is similar to the `ObjectEvent` class except that it has to be associated with at least one business transaction.

For more information on the EPCIS events, refer to Hribernik et al. (2007).

## 3.3 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 and Meindl 2004; Simatupang and Sridharan 2005). Supply chain collaboration is a growing field of research; however, most collaborative efforts have so far been focusing on the demand side (VICS 1998; Waller et al. 1999): 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 et al. 1997). Although more advanced identification technology can help to increase downstream inventory accuracy (Atali et al. 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).

In order to optimize short-term operations, decision-makers along the supply chain need to be informed about problems at upstream stages and their options for dealing with a particular problem. The short-term actions available to steer supply vary according to individual supply chain characteristics: in long- and medium-haul

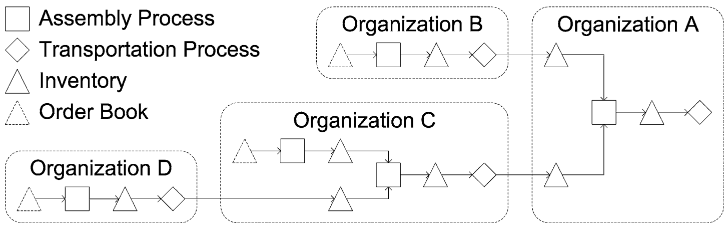


**Fig. 3.4** Example of formalized assembly network

transportation there often exists 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; or 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 only valid for a very short period of time; 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.

To be able to analyse 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 3.4 shows an example assembly network consisting of four supply chain organizations forming a three-tiered assembly network. The network conforms to the rules stated above. We will use this example throughout the section to illustrate the working of our event-based architecture.

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 Sect. 3.4 is to provide all nodes in the network with the technical means to share the required information in a decentralized way. A coordination mechanism that determines the optimal control option in case the scheduled events do not match the expected events within a certain range of tolerance is deliberately not part of this work.

## 3.4 Decentralized EPCIS-Based SCEM

In this section we propose an extension to the EPCIS specification and describe a process that utilizes this extension for providing participants of a supply chain with additional planning capabilities and enabling them to timely detect delays and promptly react to them. The proposed extension affects data, protocol, and application layers of the EPCIS specification. Below we provide a detailed description of these modifications.

### *3.4.1 Data Layer*

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 and for the interchange of event data between organizations. However, the business application described in Sect. 3.3 requires context data in the shape of expected events; therefore, we extended the EPCIS event data framework by the class ExpectedEvent (see Fig. 3.5).

Interorganizational sharing of event data can also be done using an ONS or a Discovery Service; however, this results in a centralized query infrastructure in the

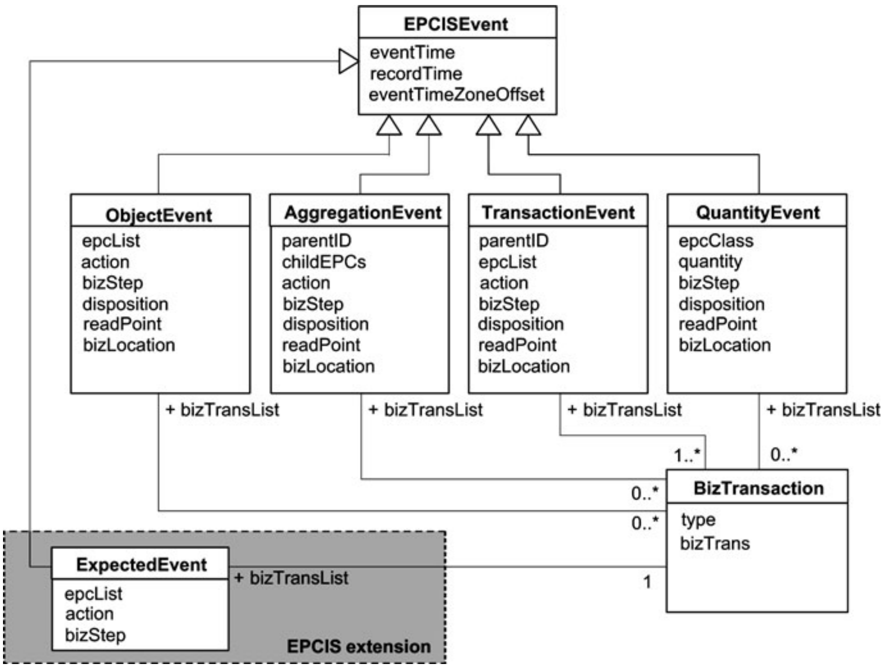


**Fig. 3.5** Extended EPCIS data model

hands of EPCglobal with the two mentioned services representing 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. In our proposal we strive to specify an architecture that takes advantage of existing bilateral business relationships in the supply chain.

### *3.4.2 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.

The communication taking place to coordinate the assembly of products is separated into six phases. Figure 3.6 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 in order 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 pseudocode below:

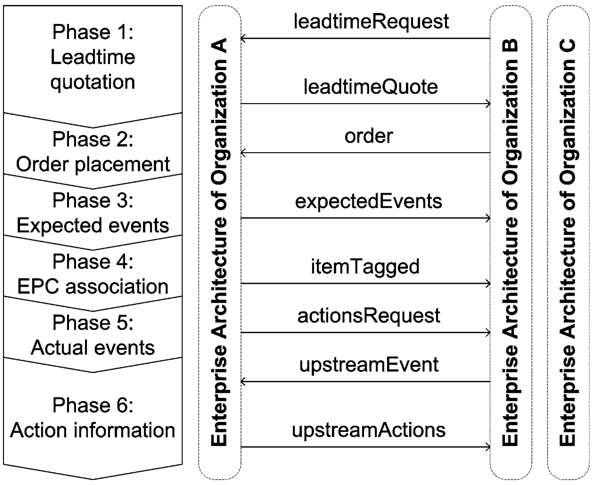


**Fig. 3.6** Protocol layer for EPCIS-based SCEM

```
• 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, endDate) 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 and 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 pseudocode 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, deliveryDate) 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 Fig. 3.5). According to EPCglobal, adding a new event type implies updating the EPCIS standard specification. 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. It then creates the events it expects to happen at its own entry and exit points. As indicated by Fig. 3.5, 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 canceled, 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 pseudocode 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 and 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 pseudocode 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 being sent downstream to spread the news on actual events taking place upstream. Each of them carries an EPCISEvent object including the attached BizTransaction object which refers 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 pseudocode 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
  – Forward 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. The path of nodes between the node that caused the delay and the node that identified the delay, we refer to 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 where 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 points. 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 and 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 pseudocode below:

- Upon reception of actionsRequest(EPCs:Set[EPC], orderID:BizTransaction, delayedStepID:BizStepID) by node i from successor:
  – If an ExpectedEvent containing EPCs exists:

      · If delayedStepID equals exitStepID:
        · Retrieve available speedup actions for EPCs and orderID
        · Send message upstreamActions(actions) 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

### *3.4.3 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 Sect. 3.4.2.

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 Sect. 3.4.2 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. 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 an SCEM application interfacing with users. Expected and actual events are stored in local EPCIS repositories that need to be accessed by the SCEM application to identify delays. The SCEM application also needs to have direct access to the protocol layer in order to retrieve action paths.

Figure 3.7 depicts the general layout of the proposed architecture. The three components Supply Chain Event Management, EPC Management, and EPC Information

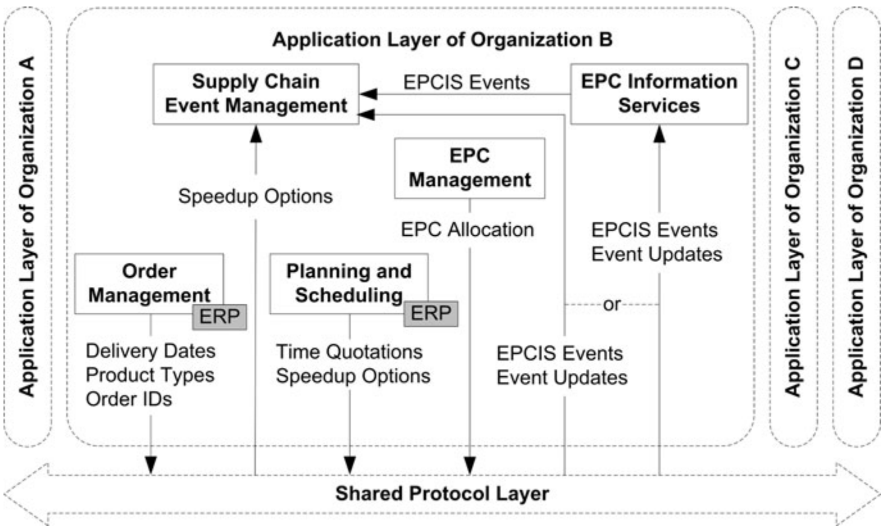


**Fig. 3.7** Two-layered EPCIS-based architecture for SCEM

Services have to be added to the existing ERP solution in order to let an organization take advantage of the data being transmitted in the protocol layer.

## 3.5 Quantitative Comparison of Two Architecture Approaches

In the previous section we described a communication protocol based on the EPCIS data format that allows for distributing scheduling and monitoring data within an assembly network. Based on the data transmitted on this protocol layer, the supply chain stake holders can estimate lead times, detect delays, and obtain complete information about speedup options. In our proposal we assumed that the communication hubs exchange event data according to established bilateral relationships determined by the structure of the assembly network. In this section we aim at providing an objective comparison of this approach with the approach implied by the current EPCIS specification, in particular the design of the EPCIS Discovery Service used for identifying EPCIS repositories along the supply chain that contain data related to a given EPC. In the following, we briefly describe the methods applied to evaluate and compare both architectural designs, present the evaluation results, and discuss their implications.

### *3.5.1 EPCIS-Based Event Sharing Using Event Pull*

EPCglobal proposes a centralized query infrastructure which can be used to retrieve all events relating to a particular EPC from all accessible EPCISs worldwide. The

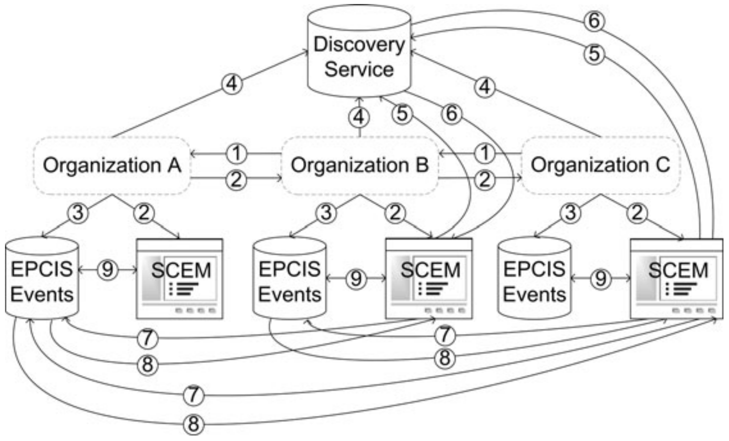


**Fig. 3.8** Event Pull in a three-tiered supply chain

retrieval process has two steps: first, the EPCIS Discovery Service (DS) is queried for a set of references to all EPCISs which have stored events involving a particular EPC. Upon receiving this set, the query interfaces of all EPCIS in the set can be directly queried for particular types of events, i.e., the range of events searched for can be restricted to the information of interest. This architecture is well suited for situations when there is no ex ante knowledge about the applications which will use it. In principle, it allows for the retrieval of EPCIS events based on arbitrary search criteria which have previously been stored in any EPCIS; therefore, it can also be used to realize SCEM.

Figure 3.8 describes how the event-sharing mechanism works in the context of SCEM if the architecture approach of EPCglobal is followed. The concrete steps are as follows:

1. The last organization (C) in the supply chain places an order with its supplier (B), which in turn places an order with its own supplier (A).
2. The first organization in the chain (A) schedules activities, translates the schedule into expected events, stores these events in its SCEM application, and sends the set of expected events downstream. The next organization in the supply chain schedules its own activities relating to the order, adds the corresponding expected events to the set, saves all events in its SCEM application, and sends the extended set downstream and so forth.
3. Actual events are continuously captured by the EPCIS.
4. Each time an actual event is captured, the organization publishes the event's availability to the EPCIS DS: in this case a key-value pair of EPC and EPCIS reference.
5. If the SCEM application wants to request the status of an EPC, it has to query the EPCIS DS to receive the address of the relevant EPCIS repositories; alternatively, a so-called standing query can be saved so that a foreign EPCIS does not need to be polled continuously.

6. The addresses of the EPCIS repositories which contain event data related to the EPCs of the order are sent to the SCEM application.
7. The SCEM applications separately and directly query each EPCIS repository which contains relevant events.
8. The EPCIS repositories send the event data requested by the downstream SCEM applications.
9. The SCEM applications constantly compare scheduled with actual events.

### *3.5.2 EPCIS-Based SCEM Using Event Push (Our Proposal)*

Contrary to the approach described in the previous section, events can be exchanged according to the bilateral relationships in a supply chain, e.g., between a manufacturer and its suppliers. Instead of replying to concrete requests from downstream organizations, upstream organizations can simply push all events relevant for SCEM to them. These events can be forwarded downstream without a previous request because upstream organizations know which events are relevant from previous interaction, e.g., the sharing of schedules. In this alternative architecture, the supply chain also serves as a type of communication network at the same time; therefore, data only needs to be exchanged by parties which are already involved in a business relationship.

Figure 3.9 describes the Event Push. Steps 1 and 2 are the same as Event Pull above. The following steps are:

3. Actual events are continuously captured and immediately sent to the adjacent downstream supply chain organization, which does the same and so forth.
4. The SCEM applications constantly compare scheduled with actual events.

### *3.5.3 Evaluation*

In order to allow for a rigorous comparison of the two architectures, we outline how they work in detail in the following sections. In spite of a number of qualitative criteria which may also have an influence on which of the proposed architectures will be preferred in practice, we will focus on quantitative

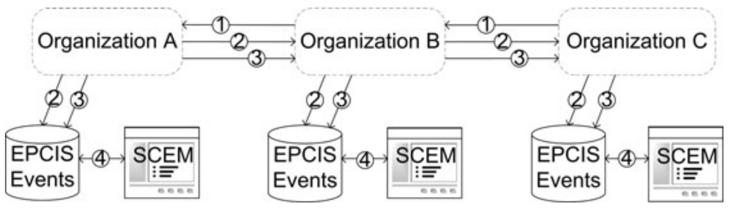


**Fig. 3.9** Event Push in a three-tiered supply chain

measures for evaluating and comparing the two system architectures presented in Sect. 3.5. The performance criteria we use refer to three of the most frequently mentioned performance characteristics of information systems (Bocij et al. 2005; Garcia et al. 2006): efficient use of network capacity, efficient use of storage, and system reliability. These performance criteria were operationalized by quantitative performance metrics based on the number of data objects stored along the supply chain and the number of messages exchanged between supply chain stages.

The performance metrics depend on several parameters which characterize the structure of and the flow of material in the supply chain; thus, the evaluation of the EPCIS-based SCEM architecture depends on performance metrics which inherently reflect the particularities of the supply chain context. Parameters and performance metrics will be formally defined in the following sections.

#### 3.5.3.1 Parameters

A supply chain is composed of at least two organizations, tiers, sites, or stages which work together in order to provide one product to the end customer. The number of tiers in each supply chain is denoted by $l \in N^+ \setminus \{1\}$. The number of supply chains which are monitored by the SCEM application is denoted by $d \in N^+$. Note that for the sake of simplicity, we do not consider intermeshed supply chains; intermeshed supply chains come into existence if at least one company takes part in two different supply chains and the organizations in these supply chains are not the same. The last parameter which is considered in our analysis is the number of tagged components or products which move through each supply chain during a fixed period of time. This parameter is denoted by $p \in N$.

#### 3.5.3.2 Efficient Use of Network Capacity

The efficient use of available network capacity is measured in terms of the absolute number of messages exchanged during a fixed period of time. A message in this case is defined as a temporarily enclosed and distinct exchange of data between the information systems of different organizations. Since the two system architectures to be compared do not differ regarding the way in which order data and schedules (or expected events) are forwarded along the supply chain, these steps are not included in the number of messages exchanged.

In the Event Push approach, the actual events which are forwarded along the supply chain are the only remaining messages. The amount of messages exchanged grows multiplicatively with the depth of supply chains; events are managed separately for different supply chains.

Consider, for example, a supply chain involving two organizations A and B; one message is sent from organization A to organization B when an event related to one product has been captured by A. If the supply chain is extended by one organizational tier (Organization C), not only the captured events of B, but also those captured by A are sent from B to C (B serves as a communication hub in this case). The number of exchanged messages in Event Push can be calculated in the following way:

$$M_{\text{push}} = d \cdot \sum_{k=1}^{l-1} k \cdot p = \frac{1}{2} \cdot d \cdot p \cdot \left(l^2 - l\right). \tag{3.1}$$

In the Event Pull approach, captured events are not forwarded to subsequent organizations in the supply chain but rather pulled from upstream organizations on demand. Again, the amount of exchanged messages grows multiplicatively with the number of supply chains the organizations are involved in. For each EPC that is read by an organization, the corresponding key-value pair has to be published via the EPCIS DS (step 4). In order to compare an expected event with the corresponding actual event, an SCEM system has to query the EPCIS DS for the reference to an EPCIS repository (steps 5 and 6). For each received reference, SCEM systems have to query the EPCIS repository for the corresponding EPCIS event (steps 7 and 8); therefore, the number of exchanged messages in the Event Pull approach is

$$M_{\text{pull}} = d \cdot \left[ l \cdot p + 4 \cdot \sum_{k=1}^{l-1} k \cdot p \right] = d \cdot p \cdot \left(2 \cdot l^2 - l\right). \tag{3.2}$$

Event Push dominates Event Pull in terms of the number of exchanged messages. The factor with which the push approach performs better can be calculated using the following formula:

$$\delta_M = 4 + \frac{2}{l-1}. \tag{3.3}$$

As (3.3) indicates, the number of supply chains $d$ and products $p$ do not play a role when comparing the performance of the two proposed architectures with respect to their use of network capacity: the performance advantage of Event Push only depends on the length $l$ of the supply chains. The number of exchanged messages produced by Event Pull is six times higher than the one produced by Event Push for supply chains with two participants ($l = 2$), 4.5 times higher for $l = 5$, 4.2 times for $l = 10$, and approaches 4 times higher for high values of $l$.

#### 3.5.3.3 Efficient Use of Storage Capacity

The efficient use of storage capacity by the two architectural approaches is measured in terms of the number of stored data objects which refer to the flow of goods. We initially compare the number of events saved in the EPCIS repositories at the different supply chain participants.

In Event Push, each supply chain participant stores its expected and actual events at its own and all subsequent sites. The number of supply chains affects this number multiplicatively. The number of saved EPCIS events can be calculated using the following formula:

$$O_{\text{push}} = 2 \cdot d \cdot \sum_{k=1}^{l} k \cdot p = d \cdot p \cdot \left(l^2 + l\right). \tag{3.4}$$

In Event Pull as proposed by EPCglobal, schedules would not be stored in the form of events within the EPCIS repositories but would be directly exchanged by the SCEM applications; therefore, the number of stored EPCIS events can be calculated according to formula (3.5):

$$O_{\text{pull}} = d \cdot p \cdot l. \tag{3.5}$$

However, Event Pull requires the storage of other data objects. Both the key-value pairs used as references in the EPCIS DS and the expected events stored separately by the SCEM applications have to be taken into account. Thus, a fair basis for comparison regarding the number of stored data objects in the pull approach is given by formula (3.6):

$$\bar{O}_{\text{pull}} = 2 \cdot d \cdot p \cdot l + d \cdot p \cdot \sum_{k=1}^{l} k = d \cdot p \cdot \left(\frac{1}{2} l^2 + \frac{5}{2} l\right). \tag{3.6}$$

No approach formalized in functions (3.4), (3.5), and (3.6) is dominated with respect to the number of stored data objects. The relative advantage of Event Push over Event Pull (or vice versa) expressed by formula (3.7) is independent of the number of supply chains and the number of products moving through each of them.

$$\delta_O = 2 - \frac{8}{l+5}. \tag{3.7}$$

The number of stored objects is 1.2 times higher if Event Pull is used for two supply chain participants, equal for three participants, 1.2 times smaller for five participants, 1.5 times smaller for ten participants, and approaches 2 times smaller for high values of $l$.

### 3.5.3.4 Reliability

The number of data objects stored at each supply chain participant should not be much above the average number in order to minimize bottlenecks and maximize reliability. We operationalize this performance criterion by measuring how dispersed the required data objects are stored in the supply chain. A standard measure of statistical dispersion is the Gini coefficient $G$. The value of $G$ ranges from 0 to 1; the nearer it is to 1, the greater the dispersion. Since reliability is expected to be greater if data objects are distributed more equally among the databases along the supply chain, a lower Gini coefficient of the number of stored objects indicates higher reliability.

**Table 3.1** Comparison results

(a)

| $l$ | $d$ | $G_{\text{push}}$ | $G_{\text{pull}}$ | $\frac{G_{\text{push}}-G_{\text{pull}}}{G_{\text{pull}}}$ |
|---|---|---|---|---|
| 2 | 1 | 0.333 | 0.381 | 12.5% |
| 2 | 2 | 0.458 | 0.471 | 2.8% |
| 2 | 3 | 0.500 | 0.531 | 5.8% |
| 3 | 1 | 0.444 | 0.438 | −1.6% |
| 3 | 2 | 0.528 | 0.530 | 0.4% |
| 3 | 3 | 0.556 | 0.575 | 3.4% |
| 4 | 1 | 0.500 | 0.478 | −4.7% |
| 4 | 2 | 0.563 | 0.562 | −0.1% |
| 4 | 3 | 0.583 | 0.598 | 2.5% |
| [···] | | | | |
| 10 | 1 | 0.600 | 0.577 | −4.0% |
| 10 | 2 | 0.625 | 0.624 | −0.2% |
| 10 | 3 | 0.633 | 0.641 | 1.2% |

(b)

| $l$ | $d$ | $G_{\text{push}}$ | $G_{\text{pull}}$ | $\frac{G_{\text{push}}-G_{\text{pull}}}{G_{\text{pull}}}$ |
|---|---|---|---|---|
| 2 | 1 | 0.333 | 0.381 | 12.5% |
| 2 | 100 | 0.581 | 0.673 | 13.7% |
| 2 | 10000 | 0.583 | 0.679 | 14.0% |
| 3 | 1 | 0.444 | 0.438 | −1.6% |
| 3 | 100 | 0.609 | 0.677 | 10.0% |
| 3 | 10000 | 0.611 | 0.681 | 10.2% |
| 4 | 1 | 0.500 | 0.478 | −4.7% |
| 4 | 100 | 0.624 | 0.678 | 8.0% |
| 4 | 10000 | 0.625 | 0.681 | 8.2% |
| [···] | | | | |
| 10 | 1 | 0.600 | 0.577 | −4.0% |
| 10 | 100 | 0.650 | 0.676 | 3.9% |
| 10 | 10000 | 0.650 | 0.677 | 3.9% |

We do not compare the Gini coefficients of data dispersion for the two system architectures formally since the derivation of a mathematical expression is highly complex if feasible at all; instead, we base our analysis on a numerical comparison. Table 3.1(a) shows the relevant results of the numerical calculations and provides the relative performance differences between both architecture approaches. The performance metrics are invariant with respect to the number of products $p$ but depend on the depth $d$ of the supply chains.

When comparing the architecture approaches based on our reliability metric, several impacts of the parameters $l$ and $d$ can be observed. The longer the supply chain becomes, the smaller the advantage of Event Push compared to Event Pull. The more supply chains there are, the greater the advantage of Event Push becomes. Table 3.1(a) shows that if the number of supply chains is very low, the push approach can have a higher Gini coefficient; however, as Table 3.1(b) shows, this disadvantage of Event Push only persists up to parameter configuration with $d = 2$, i.e., it should be negligible in realistic settings.

## *3.5.4 Results*

Supply chain wide visibility of the flow of goods is a precondition for supply chain event management. We have compared two possible system architectures that enable the sharing of standardized supply chain event data with respect to a number of quantifiable criteria. According to our evaluation, none of the approaches can be preferred without further consideration.

**Table 3.2** Relative advantage of event push over event pull

| Length of supply chain $l$ | Network capacity | Storage capacity | Reliability |
|---|---|---|---|
| 2 | 83.3% | 14.3% | 14.0% |
| 3 | 80.0% | 0.0% | 10.2% |
| 4 | 78.6% | −11.1% | 8.2% |

The parameters we used to evaluate and compare the two architectures are realistic variable values for length, depth, and number of products. Iyengar (2005) calculated the average length of supply chains using the US Benchmark Input–Output tables published by the Bureau of Economic Analysis. Based on data from more than 1 million supply chains, he found that in 1997 the average US supply chain had a length between 3.4 and 4.1 depending on the industry. Length was defined as the number of echelons of the supply chain. On the basis of these figures, it seems realistic to consider supply chains consisting of two to four participants.

Estimating a realistic number of supply chains, which would benefit from SCEM applications, and the number of products flowing through these supply chains is considerably more difficult but can be expected to be very high. Kürschner et al. (2008) state that the EPCIS Discovery Services will have to be able to handle queries from millions of clients. Against this background, our estimation of 10,000 supply chains, which are monitored using an SCEM application, should be realistic.

Table 3.2 summarizes the relative advantage of Event Push compared to Event Pull with respect to the quantitative metrics defined in Sect. 3.5.3.1 and based on realistic parameter values. In spite of the typical trade-off between usage of data storage and network bandwidth, Event Push appears to be the preferable architectural choice for short supply chains: up to a supply chain length of three echelons, the push approach dominates the pull approach according to our criteria.

## 3.6 Discussion

We have presented a business application and a corresponding information system architecture that provide the basis for the short-term coordination of a multiorganizational 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 multiorganizational business environments, in particular system interoperability and the interorganizational 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, this work 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 multiorganizational 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 bilateral character. 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: first, 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; second, one common way to describe event data and its context based on the EPCglobal event data specification is used both for intra- and interorganizational communication.

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 relieves the burden of downstream organizations from the need to maintain a comprehensive up-to-date internal process view of other organizations. Furthermore, ex ante knowledge of the organizational 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.

The second research question has been whether the current proposal for the distributed system architecture of the Internet of Things is suitable for SCEM applications. Based on three quantitative criteria, we come to the conclusion that an alternative approach based on the idea of pushing EPCIS events downstream could be the preferable choice. We have also mentioned some qualitative advantages of the latter architecture, such as taking advantage of existing business relationships in the supply chain and not requiring a central authority for data management and authentication which speak for Event Push.

## 3.7 Conclusions and Future Work

Although our quantitative measures are coarse and based on simplistic assumptions, they provide an objective means for an initial comparison of Event Pull and Push to realize inter-organizational SCEM.

Further research on the topic is definitely warranted: in order to make an informed decision, the relative importance of different performance criteria and metrics will have to be determined (e.g., based on the available network and data storage capacity and on the variable costs of storing and transmitting data). Furthermore, additional criteria and metrics should be defined to obtain a more detailed picture of possible cost-benefit trade-offs. For instance, operational properties such as latency and throughput could be measured and compared using supply chain simulations as soon as implementations of the proposed architectures are available. Finally, economic translations of the somewhat technical performance measures used in this work have to be defined in order to enable a sound investment decision by adopters of EPCIS-based SCEM.

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 behavior, e.g., by using dedicated public key infrastructures.

## References

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

Bocij, P., Chaffey, D., Greasley, A., Hickie, S.: Business Information Systems: Technology, Development & Management for the E-Business, 3rd edn. Prentice Hall, Upper Saddle River (2005)

Chen, F.: Information Sharing and Supply Chain Coordination. In: T. de Kok, S. Graves (eds.) Supply Chain Management: Design, Coordination and Operation, pp. 341–421. Elsevier, Amsterdam (2003)

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

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

EPCglobal: EPC Information Services (EPCIS), Final Version 1.0.1 (2007)

EPCglobal: Object Naming Service Standard Version 1.0.1. http://www.epcglobalinc.org (2008a)

EPCglobal: Tag Data Standard Version 1.4 (2008b)

EPCglobal: The Application Level Events (ALE) Specification. Version 1.1 Part I: Core Specification (2008c)

EPCglobal: The EPCglobal Architecture Framework, Version 1.3 (2009)

Floerkemeier, C., Roduner, C., Lampe, M.: RFID Application Development with the Accada Middleware Platform. IEEE Systems Journal **1**(2), 82–94 (2007)

Garcia, J.D., Carretero, J., Garcia, F., and Calderon, J.F., Singh, D.E.: A Quantitative Justification to Partial Replication of Web Contents. In: Computational Science and Its Applications, pp. 1136–1145. Springer, Berlin (2006)

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

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

Iyengar, D.: Effect of Transaction Cost and Coordination Mechanisms on the Length of the Supply Chain. Ph.D. thesis, University of Maryland (2005)

Kürschner, C., Condea, C., Kasten, O., Thiesse, F.: Discovery Service Design in the EPCglobal Network—Towards Full Supply Chain Visibility. In: Proceedings of the Internet of Things 2008, Zürich, pp. 19–34 (2008)

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

Michael, K., McCathie, L.: The Pros and Cons of RFID in Supply Chain Management. In: International Conference on Mobile Business, Sydney, pp. 623–629. Sydney, Australia (2005)

Niederman, F., Mathieu, R.G., Morley, R., Kwon, I.W.: Examining RFID Applications in Supply Chain Management. Communications of the ACM **50**(7), 92–101 (2007)

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

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

Trappey, A.J.C., Lu, T.H., Fu, L.D.: Development of an Intelligent Agent System for Collaborative Mold Production with RFID Technology. Robotics and Computer-Integrated Manufacturing **25**(1), 42–56 (2009)

VICS. V.I.C.S.A.: Collaborative Planning, Forecasting and Replenishment. http://www.cpfr.org (1998)

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