2010

Architectural Patterns for RFID Applications in Manufacturing

Holger Ziekow
Humboldt-Universität zu Berlin, ziekow@wiwi.hu-berlin.de

Lenka Ivantysynova
Humboldt Universität zu Berlin, lenka@wiwi.hu-berlin.de

Oliver Guenther
Humboldt University, guenther@wiwi.hu-berlin.de

ARCHITECTURAL PATTENRS FOR RFID APPLICATIONS IN MANUFACTURING

Journal:	*18th European Conference on Information Systems*
Manuscript ID:	ECIS2010-0325
Submission Type:	Research Paper
Keyword:	IS/IT architecture, Information systems developers, Software architecture, IS design

ARCHITECTURAL PATTERNS FOR RFID APPLICATIONS IN MANUFACTURING

Ziekow, Holger, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, ziekow@wiwi.hu-berlin.de

Ivantysynova, Lenka, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, lenka@wiwi.hu-berlin.de

Günther, Oliver, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, guenther@wiwi.hu-berlin.de

Abstract

When applying RFID in production, it needs to be integrated into the used IT systems. However, till today system designers cannot rely on a standard solution for integrating RFID technology in manufacturing software systems. Each manufacturer has to deal with the same challenges: No consolidated findings on how to integrate RFID into the IT infrastructure exist. The consequence is that each IT department has to develop a solution from scratch. In order to give well-founded design guidelines for embedding RFID into the company's IT infrastructure, we conducted seven in-depth case studies of the state-of-the-art in manufacturing IT infrastructures. Our findings allowed us to specify architectural patterns for common RFID applications in manufacturing. With our work we support system designers in choosing the appropriate architecture for different RFID applications and design goals. We present our solutions in form of architectural patterns that enable manufacturing IT systems to benefit from RFID.

Keywords: RFID, Manufacturing, Architectural Patterns.

1 INTRODUCTION

RFID technology paves the way for numerous optimizations in production processes, e.g., Chappel et al. 2003. Some applications are a mere replacement of bar-code technology but others can only be realized exploiting the special properties of RFID. In our earlier work we have identified main use cases for RFID in manufacturing (Ivantysynova et al. 2008b). We found that a major benefit of RFID is that it allows restructuring manufacturers' IT architectures. This is because RFID enables to store production data and identifiers right at the product. This shift in data storage opens up new opportunities in deploying logic for control and data processing. The technology thereby provides opportunities for optimizing the IT architecture. However, applying RFID in production comes along with the challenge of integrating this technology into the used IT systems. Here, each manufacturer has to face the challenge of making the right architectural decisions for the integration. Each IT department has to develop a solution from scratch, without the foundation of a design framework. System architects lack dedicated design guidelines for using RFID in manufacturing applications. This generally increases the complexity of RFID introductions and hampers RFID investments in the manufacturing domain (Strüker et al., 2008, p.8).

Generally architectural patterns help system designers to pick an appropriate solution for their demands. The literature provides a range of patterns for standard problems (e.g., Shaw and Garlan 1996). However, these patterns are designed for generic applications and consequently coarse grained. In our work we present architectural patterns that are dedicated for integrating RFID in the manufacturing domain. We particularly focus on IT infrastructures in manufacturing. Based on previously conducted case studies (Ivantysynova et al. 2008a) we present architectural patterns that we developed particularly for integrating RFID into the manufacturing domain. These patterns provide solutions for reoccurring problems and help designing architectures for case specific requirements.

The remainder of the paper is structured as follows. First we present related work. Section 3 discusses findings from seven case studies on manufacturing IT systems with regards to common activities and hardware options. In Section 4 we present our architectural patterns for integrating RFID in manufacturing. We conclude in Section 5.

2 RELATED WORK

Patterns in software design provide solutions for reoccurring problems and help designing architectures for case specific requirements. Several patterns for general problems exist in literature (Gamma et al. 1994). Architectural patterns address the general structure of software systems and the task of its components. Prominent examples include the layered pattern for abstraction (e.g., in the OSI model) and the model-view-controller that is used to separate data from logic and presentation. However, such existing patterns are explicitly designed in a generic way and independent of application domains. More concrete and purpose build design patterns are required to provide dedicated support for integrating RFID in manufacturing.

So far, integration of RFID in business applications has mainly been studied for scenarios in logistics (e.g., Sabbaghi et al. 2008 and Lee at al., 2004). EPCglobal has defined standards for capturing RFID events and providing them to business applications (EPCglobal, 2007). However, these standards basically define interfaces for tracking and tracing applications. Details about the data processing and physical deployment of logic are not addressed. Furthermore, the integration into manufacturing infrastructures and manufacturing tasks are not part of EPCglobal standards. This is in contrast to our work that particularly addresses these architectural issues in the manufacturing domain.

From a technical perspective, RFID integration was mainly targeted in the context of middleware solutions. Existing work on RFID middleware addresses general tasks for RFID data processing, e.g.,

Bornhövd et al. (2004). These works provide some details about data processing technologies and the physical deployment. However, the technical integration in manufacturing infrastructures and the use of RFID in production tasks are not addressed.

A standard for IT systems in manufacturing is the ISA-S95 standard family (Brandl, 2000). These standards use a four layer model for describing software systems and their interaction in manufacturing. Simplified, layer four comprises software for Enterprise Resource Planning (ERP), layer 3 Manufacturing Execution Systems (MES), and layer 2 and 1 contain functionality of Distributed Control Systems (DCS) and software for Supervisory Control And Data Acquisition (SCADA). ISA-95 describes needed functionalities for these layers and how these functionalities should interoperate with each other. For example, ISA-S95 provides guidance under which circumstances which functionalities should run in which layer. However the standard only defines functionalities and which information need to be exchanged between these functionalities for the upmost level 4, level 3, and to level 3. Because RFID functionalities lie in level 1, their integration is not addressed in the standard.

3 CASE STUDIES

In this section we describe the requirements for the infrastructure and hardware options for the realization that we found in our case studies on IT infrastructures. Overall we conducted seven case studies that focused on IT infrastructures at manufacturers. Together with our previously conduced case studies on RFID in manufacturing (Ivantysynova 2008a), these studies provide the basis for our architectural patterns. Due to confidentiality agreements, we do not provide the company names. We investigated the IT infrastructures at a manufacturer of (1) engine coolers, (2) engines, (3) fire proof materials, (4) pharmaceuticals, (5) power plants, (6) milk products, and (7) tires. We thereby cover examples from the process industry as well as from discrete manufacturing. We provide details about each of these case studies in (Ivantysynova et al. 2009). Overall, the case studies show that IT architectures at manufacturers show significant variations. The hardware environment, the physical deployment of business logic, the data flow, and the location of information storages are among the varying architectural aspects. However, it is possible to find reoccurring structures that serve as basis for our architectural patterns.

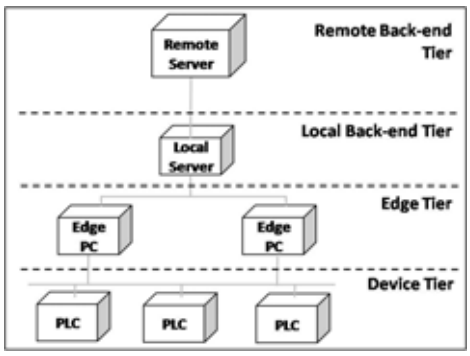

Figure 1. Common hardware tiers in manufacturing

The case studies show that the hardware infrastructures are generally organized in a hierarchical structure with several tiers. We found up to four different tiers as depicted in Figure 1. The lowest tier is the *device tier* on the shop floor. It comprises programmable logic controllers (PLC) of the machines on the shop floor. Above this, we often found an *edge tier* that comprises computing devices on the plant floor. For example, these can be PCs or terminals of control servers for production stations. In the following we refer to these computing devices as edge PCs. The tier above is the *local back-end tier*. It resembles central servers in a plant. For example, a server for hosting central MES components such as a historian or modules for task scheduling. The upmost tier is the *remote back-end*

tier. This tier refers to the often remotely hosted ERP systems. Note that not all tiers necessarily exist at each manufacturer. The number of tiers in a manufacturing infrastructure is particularly influenced by requirements for fast system responses, reliability, and scalability. Manufacturers with high demands for these requirements have usually all four hardware tiers in place. In contrast, at least one tier is missing otherwise. Which tiers exists and which functionality they host is one of the main architectural decisions in manufacturing and is also reflected in our architectural patterns.

In our case studies, we found that data acquisition from the shop floor serves two primary purposes: One is to observe the process and the other is to automate the execution of process steps. Both fit into the basic pattern of automation defined by Sheridan (2002). We adapt this pattern to the manufacturing domain and use it to define basic activities, see Figure 2. This model provides the underlying structure for our pattern definitions. We subsequently refer to this model as the *automation pattern for manufacturing*. Our architectural patterns set up on this model to define the realization of activities and data flows with respect to different design goals.

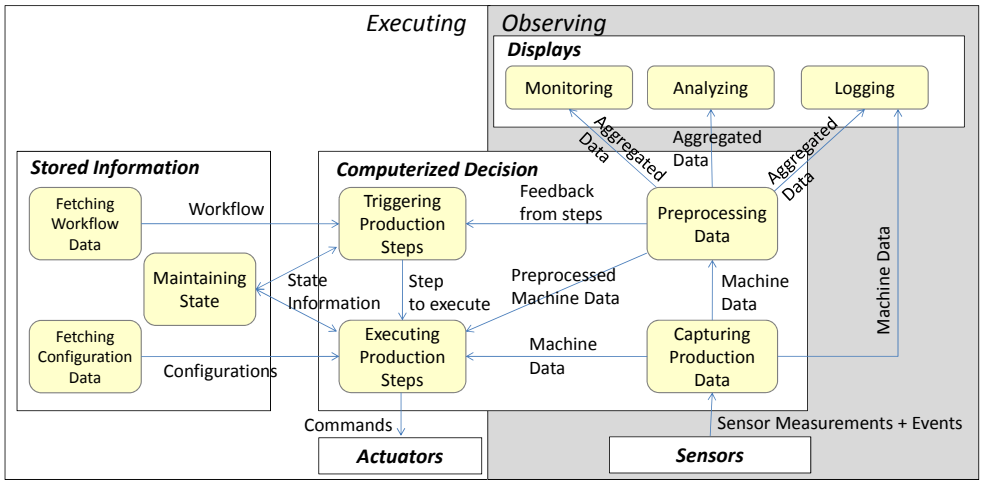

Figure 2. Elementary activities for automation in manufacturing.

4 PATTERN DEFINITIONS

In this section we present the architectural patterns for integrating RFID in the manufacturing domain. Our patterns target (1) the distribution and access of context data along the production process, (2) the distribution of filter logic in the system architecture, and (3) the distribution and structuring of control logic in the IT infrastructure.

4.1 Architectural Patters for Distributing Context Data

Patterns for distributing context data provide solutions for managing the flow of context data along the production process. With the term context data we refer to workflow data for controlling the process, state information about the product in the process, and configurations for each production step. The following patterns address the management of stored formation in the automation pattern for manufacturing, see Figure 2. They provide solutions to the distribution of workflow data, state information, and configurations along production steps. Additionally involved activities are the triggering and execution of production steps.

All three presented pattern comprise a central coordination component that initiates the production and manages it to a certain degree. They locate this component on a server in the back end. Additionally, these patterns make use of edge PCs at the production stations. These edge PCs host components for

triggering and executing production tasks. The patterns vary in how these components interact and exchange context data as well as in the organization of intermediate storage.

4.1.1 RFID-Pipeline Pattern

The RFID-pipeline pattern defines the deployment of logic for triggering production steps, executing production steps, and the corresponding data flow. Strengths of this pattern are that it supports reliability and scalability of the system. It further eases fast data access at the production stations and simplifies the control of data flow.

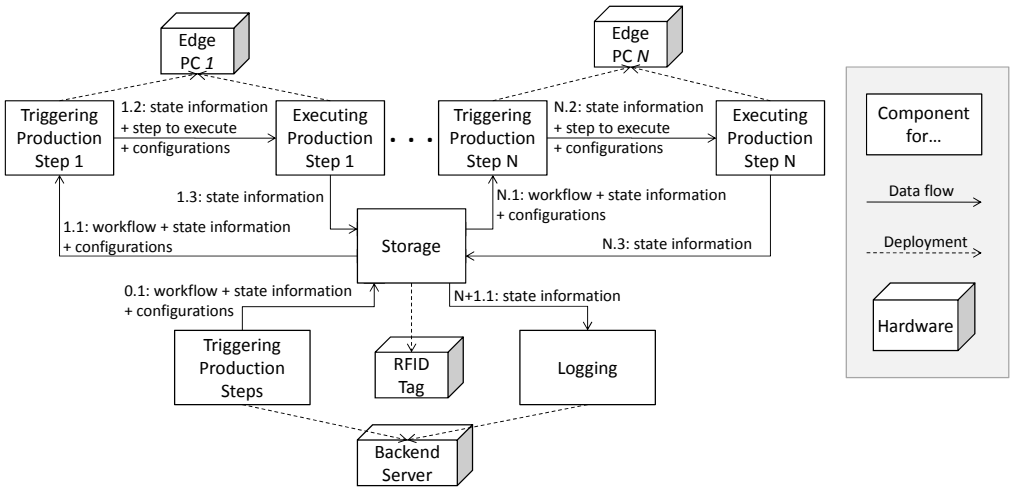

Figure 3. The RFID-pipeline pattern.

Figure 3 shows the data flow and physical deployment defined by the RFID-pipeline pattern. Activities in an RFID pipeline fall into three sequential parts. (1) Initializing the pipeline, (2) sequentially triggering and executing production steps, and (3) finalizing the pipeline. In the first step, initial state information, workflow data, and configurations are written on an RFID tag. In most cases the initialization would be triggered in the back-end system because it holds the required information. It follows the sequence of triggering and executing production steps. The processing-logic for each step is deployed on an edge PC on the plant floor. Each step reads state information from the previous step, workflow data, and configurations from the RFID tag. After processing, the updated state information is written back to the RFID tag. After the last processing step, log data about the conducted production is extracted from the tag. The pattern does not define details of this extraction, i.e. to where the log data is written. However, in most cases this logging process involves reports to the back end and storage in a historian.

Advantages of the RFID-pipeline pattern are its support for fast data access, system reliability, and scalability. The pattern ensures fast data access because the information is physically located at the point where it is needed, i.e., on the RFID tag of the currently handled product. We found in our case studies that manufacturers struggle to ensure timely access to remote system components and databases. Here, the RFID-pipeline pattern provides an alternative to ensure fast access; meaning access with low delay. Data throughput for RFID technology is low compared to wired data transfer. Thus, the pattern is suitable if the required fast data access is for small amounts of data per transaction.

System reliability is supported because the pattern provides autarky of system components for different production steps. That is, workflow data, state information, and configurations are available even in case of back-end failures. Once the initial control data is written on a tag (see step 0.1 in Figure 3), the production can continue without the back end. System scalability is supported by the distributed design of the pattern. Due to the decentralization of information storage and processing, the pattern scales well with the number of parallel executed production tasks.

The desire to apply this pattern was a major driver for the RFID application in half of our RFID specific case studies at manufacturers. We further found this pattern in one of our general case studies on manufacturing infrastructures (case (2).

4.1.2 Pulled-Context Pattern

The pulled-context pattern defines the deployment of logic for triggering production steps, executing production steps, and the corresponding data flow. Strengths of this pattern are that it supports the scalability of the system. It also eases fast data access at the production stations and simplifies the control of the data flow.

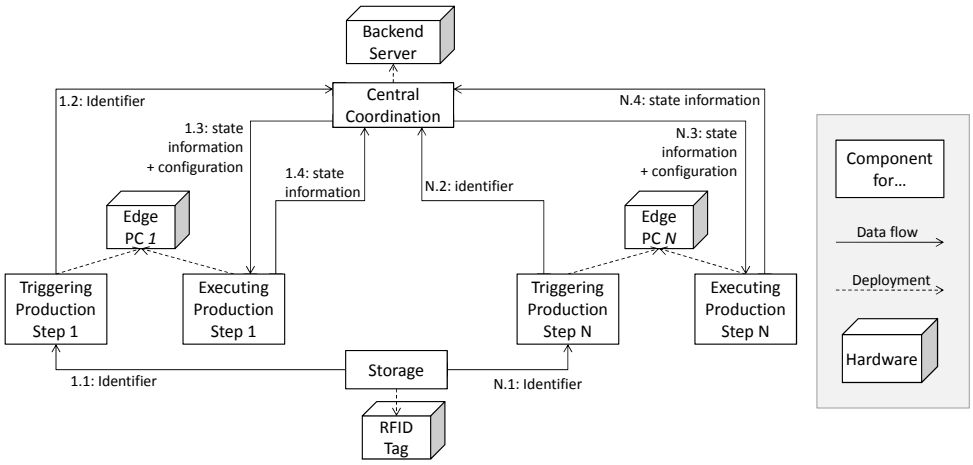

Figure 4. The pulled-context pattern.

Figure 4 shows the data flow and physical deployment that the pulled-context pattern defines. The data flow starts with reading the identifier from the RFID tag at a production station. The edge PC at the production station uses this identifier to pull the required context information for the production from a central coordination component in the back end. It is important to note that distance reads of tags RFID can help in conducting the pulling step before the data is actually required. The central coordination component sends context data for the production to the corresponding edge PC as response to the pull request. After conducting the production step, the edge PC transfers updated status information back to the central coordination component. This procedure is repeated afterward for each production step.

An advantage of the pulled context pattern is its support for system scalability. It further provides a simple solution for distributing context data for the production to the correct production stations. Depending on the production process, the use of RFID in this pattern can also prevent that delays in the IT system slow down the production. The pattern does not require extended user memory on the RFID tags. Instead, tags must only store a unique identifier. This makes the pattern suitable for open loop scenarios where tags remain on the products low cost RIFD label are required.

System scalability is supported due to the decentralization of data processing for each production step. However, a central coordination component is responsible for data distribution to each processing step. This design can result in delays during the data request. However, in some application settings, RFID can enable fetching the identifier for pulling data before the production starts. Particularly in manual processes the possibility of distance reads can allow for pulling proactively. This also comes with the benefit that data fetching is decoupled from the actual production process. We found potential for applying this pattern specifically in two case studies on RFID where the process allowed proactive fetching of state information.

4.1.3 Pushed-Context Pattern

The pushed-context pattern proactively caches context data for the production at production stations. It uses identifiers on RFID tags for selecting data in these caches when the corresponding product passes the production step.

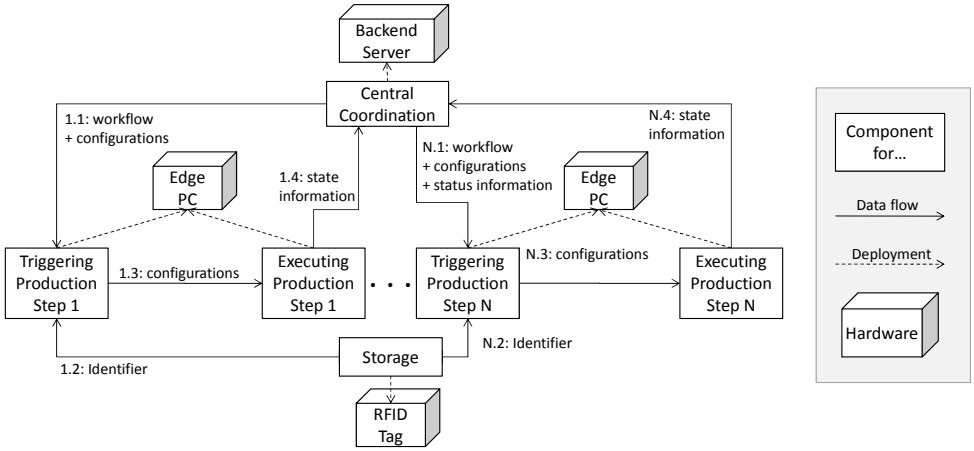

Figure 5. The pushed-context pattern.

Figure 5 shows the data flow and physical deployment defined by the pushed-context pattern. The data flow starts with pushing context information to edge PCs at the production stations. When the production starts, the edge PC retrieves the needed data from its local cache using the identifier of the RFID tag. Subsequently the central control component receives updates about the product status and updates the downstream production station accordingly.

The main advantages of the pattern are the support for system reliability, scalability and fast reactions of the IT system. Fast reactions are supported due to the proactive caching. Thus, it is ensured that required context data are quickly available at the point where they are needed. The downside of proactive caching compared to pulling is the increased complexity of the data-distribution process. It is required that the central-coordination component can accurately predetermine where and when processing of a certain product will happen. Challenges can occur for example if multiple production stations perform the same tasks in parallel. A possible solution to this is a combination of the pushed-context pattern with the pulled-context pattern. That is, the central-coordination component can push context data to a specific control station that manages a set of parallel running production stations. From there the data can be pulled as the corresponding product arrives at a particular station.

Scalability is supported due to the decentralized control of production steps. However, the central-control component poses a potential bottleneck. Nevertheless, delayed responses of this component are tolerable due to the proactive caching. Support for reliability is dependent on how big the proactive filled caches and material buffers are. In case of a back-end failure, the production can continue as long as caches and material buffers are filled. Using this pattern was an explicit design decision in one of the manufacturing infrastructures that we investigated. Here the manufacturer used proactive caches to ensure continuous production in case of back-end or network failures for up to four hours. Generally, we found this pattern to be applicable in all discrete productions that we analyzed.

4.2 Architectural Patters for Distributing Filter Logic

Raw RFID reads require several filtering and preprocessing steps before business applications can use them. That is, the IT system must transform the stream of tag observations in business relevant information. Following the terminology of Luckham (2001) we call this transformation of raw events into primitive events and subsequently into increasingly more complex events. For RFID events the

sequence of operations typically starts with projection followed by data-cleaning operations. Jeffery et al. (2006) define four cleaning steps in a general framework. Subsequent to data cleaning, additional projections and complex event processing (CEP) steps extract the relevant data for higher level applications, e.g., production control or monitoring.

The design questions that arise in context of RFID-data filtering and preprocessing concern the distribution of logic in the IT infrastructure. In this subsection we present two architectural patterns that provide solutions to this design question. These patterns mark extremes in a spectrum of possible solutions. Application designers may adjust the pattern due to application specific constraints.

We group filters and preprocessing operations into operations for selection, low pass filtering, error correction, aggregation, and CEP. With selection we refer to operations to select data sources and projecting out data. This is equivalent to the point operation in the data cleaning framework of Jeffery et al. (2006). With error correction, we refer to data-cleaning operations that go beyond filtering on a single input stream. For example, this can be filtering after correlation with additional sensor data as described in (Jeffery et al., 2006). With aggregation we refer to operations that summarize a set of input events. An example is forwarding the number of detected items rather than each individual detection. With CEP we refer to the detection of complex-event patterns and evaluation of complex predicates on RFID reads. Generally, this adds semantics to the input data and enables filtering on a higher semantic level. A simple example is inferring from a stream of RFID events, that a certain set of items was packed in a certain box. A filter on top of that may be to report only boxes that were packed in a wrong way. Further selection operations reduce the data set - that might have been enriched with additional sensor and context data for intermediate processing steps - to the attributes required for higher level applications.

4.2.1 Thin-Filter Edge Pattern

The thin-filter edge pattern defines the deployment of logic for filtering and preprocessing of RFID data on different hardware tiers. The pattern is an architectural alternative to the later described thick-filter edge pattern. The main strength of the pattern is its simplicity. It fits best to infrastructures with a thin edge tier and to applications where RFID is used for process monitoring and documentation rather than for real time control of individual production steps.

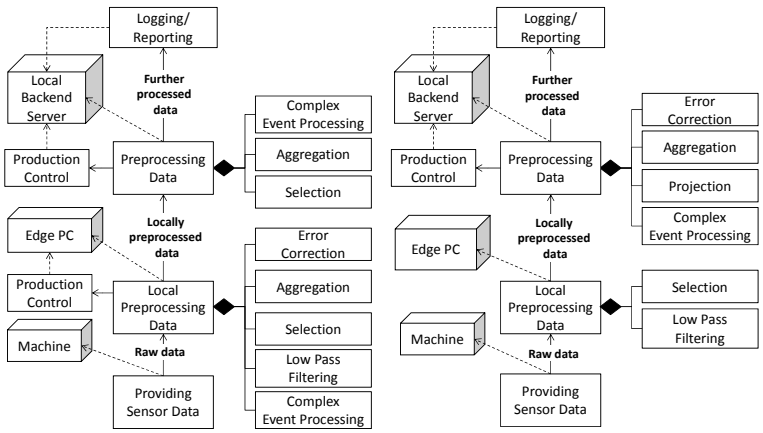

Figure 6. The thick- filter edge pattern (left) and thin-filter edge pattern (right).

The pattern directs raw RFID events through a pipeline of filtering and preprocessing operations, see Figure 6 right. The first operations in the pipeline run on edge PCs that control the RFID readers. (Note that some readers have powerful computation capabilities and can replace the edge PC in this pattern.) Subsequent filter operations run in the back end before data is passed on to higher level applications.

Filter operations on the edge tier are restricted to selecting available data streams and low-pass filtering for simple error correction. Typically device controllers for RFID readers support this operation (see Bornhövd et al., 2004). Filter operations in the back-end tier comprise more complex operations, error correction, aggregation, CEP, and projection. (Note that all these operations may run as rules in a dedicated rule engine in an RFID middleware (Bornhövd et al., 2004) or CEP engine (Coral8, 2006).

The main advantage of the thin-filter edge pattern is that it keeps processing at edge devices simple. It matches with the typical distribution of filter operations in RFID middleware solutions. These solutions focus on monitoring applications e.g., for logistic tracking or warehouse management. For manufacturing this pattern is most suitable if RFID is used for monitoring and documentations rather than for real-time control of production steps. The low requirements for the edge layer make this pattern fit into environments with little IT on the plant floor. Here, an RFID implementation with a dedicated middleware can provide RFID data for process monitoring via a slim interface and without major interactions with other manufacturing operations. This property makes the pattern suitable in three of our RFID specific case studies.

4.2.2 Thick-Filter Edge Pattern

The thick-filter edge pattern defines the deployment of logic for filtering and preprocessing of RFID data on different hardware tiers. It is an architectural alternative to the previously described thin-filter edge pattern. Strengths of this pattern are its support for scalability, autarky and ability for fast responses of the IT system.

The pattern directs raw RFID events through a pipeline of filters and preprocessing operations, see Figure 6 left. A large set of these operations is deployed on edge PCs. These are selection, low-pass filters, aggregation, projection, error correction, and operations for CEP. From here, enriched and filtered data can directly flow into modules for production control in the edge tier. The data is further passed on to the back end for additional processing steps.

The RFID data that arrives in the back end is already cleaned and enriched. However, some additional processing that could not have been done in the edge tier may be required. For example, the back end may integrate data from several sources and apply operations for CEP and aggregation afterward. Subsequently, event data can be passed on to higher level applications.

The main advantage of the thick-filter edge pattern is that it supports scalability. The pattern is particularly useful in applications were RFID creates massive data volumes or is involved in control functionality. Extensive decentralized filtering and preprocessing in the edge layer reduces the burden for the back-end system. This was exploited e.g., in case (4) and (6) of the investigated companies. The design also supports scalability of the overall system and helps to keep response times low. Furthermore, the pattern supports fast reactions in decentralized control and steering processes for the production. The edge layer filters and preprocesses raw RFID events to a large extend. From there the data can directly feed modules for production control in the edge layer. (For example, an RFID event may trigger a process step.) Note that the back end is out of the loop. This supports fast reactions in the edge layer and autarky for the production control as desired in three of our RFID specific case studies.

4.3 Architectural Patterns for Distributing Control Logic

In this subsection we provide architectural patterns for distributing production control logic in IT infrastructures of manufacturers. This concerns the control and execution of production steps; like controlling machines and managing the order of production tasks. The distribution of control logic is a fundamental design decision that manufacturers have to make whether or not they apply RFID. However, the design decision impacts how RFID can be applied and vise versa. For example, not all

patterns presented in this section are compatible with every distribution of control logic. (For a discussion see section 4.4) In our case studies we found two fundamental options for distributing control logic. We present two architectural patterns that reflect these options.

4.3.1 One Tier Control Pattern

The one-tier control pattern defines the flow of information and deployment of logic for production control from the perspective of a whole production plant, see Figure 7.

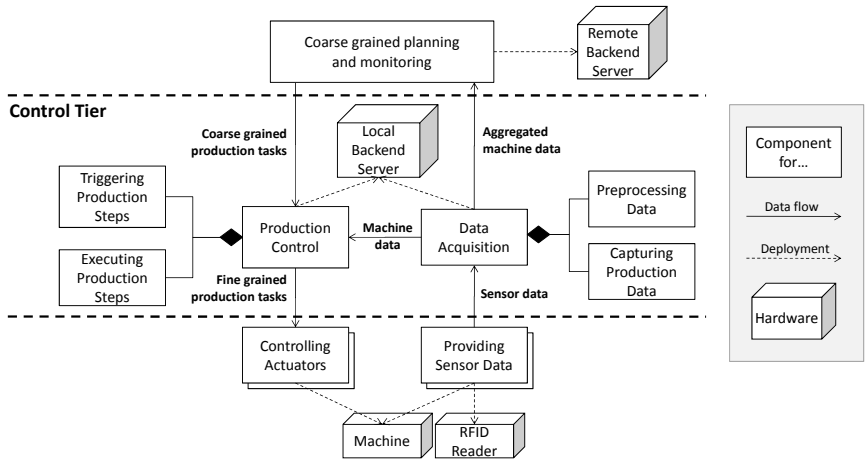

Figure 7. The One Tier Control pattern

Coarse grained control of the production tasks is managed in the remote back-end system. In the local back end the pattern defines a tier that hosts all modules for controlling the production processes. That is, from here instructions are directly send to controllers for actuators on machines. Feedback from the plant floor is sent back in form of sensor or RFID data. This information is then evaluated in the local back end and used in production control.

The main advantage of the pattern is its simplicity. It therefore fits well to productions where light-weight infrastructures are required (e.g., case (5)). Its central design is also favorable if the production requires a centralized view on the processes (e.g., case (4) and (6) of our studies). In the context of RFID, the pattern fits best where RFID is mainly used for monitoring and documenting the production processes rather than for real-time control of operations (this was the case in three of our RFID specific case studies).

4.3.2 Multi-Tier Control Pattern

The multi-tier control pattern defines the flow of information and deployment of logic for production control in several hardware tiers of a production plant, see Figure 8. The main advantages of the pattern are its support for scalability, autarky, and fast reactions of the system. However, these benefits come at the cost of a complex system design.

The pattern distributes control logic in two hardware tiers. These two control tiers are located between the software for coarse grained planning and monitoring (e.g., ERP) and the machines on the plant floor. The first (upper) control tier hosts software that coordinates the production on task level. The lower control tier executes the actual production and passes corresponding reports back to the first control tier that reports to the remote back end in turn. In its hierarchical structure the multi-tier control pattern is related to the presentation-abstraction-control pattern for hierarchically organized software agents. Similarly, more than one level of hierarchy may be implemented. In manufacturing, the number of levels depends on the organization of production facilities.

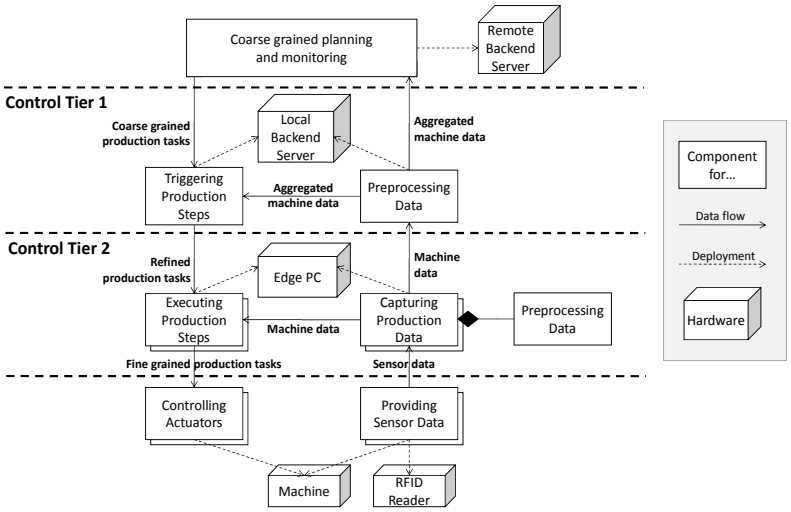

Figure 8. The multi-tier control pattern with two tiers.

The main advantages of the multi-tier control pattern are its support for scalability, autarky, and fast reactions of the system. A prerequisite for its application is availability of computation hardware on the plant floor. This can make the pattern infeasible where light-weight IT infrastructures are required. In the context of RFID, the pattern fits best if RFID is directly involved in control functionality (e.g. in case (2)). This is due to the patterns support for processing RFID data on the edge and thereby ensuring fast reactions to RFID inputs (e.g. as in three of our RFID specific case studies).

4.4 Combining Presented Patterns

The presented patterns focus on different views of architectural designs. However, there is some overlap in the addressed aspects. Thus, some patterns can be combined while others are conflicting. Table 1 provides an overview of how patterns may be combined. (The table entries denote that patterns can be combined very well ("++"), can be combined ("+"), can hardly be combined ("-"), or are contradictory ("- -").

	RFID Pipeline	Pulled Context	Pushed Context	Thin Filter Edge	Thick Filter Edge	One-Tier Control	Multi-Tier Control
RFID Pipeline		- -	- -	+	++	-	++
Pulled Context			- -	+	++	-	++
Pushed Context				+	++	-	++
Thin Filter Edge					-	++	-
Thick Filter Edge						+	++
One-Tier Control							- -
Multi-Tier Control							

Table 1. Combining the presented architectural patterns.

5 CONCLUSION

The integration of RFID in business applications was mainly studied in the field of logistic applications. Here EPCglobal developed standards for capturing and exchanging RFID events.

However, these standards basically define interfaces for tracking and tracing applications. Existing work on RFID middleware does not provide details on the integration into manufacturing infrastructures. That is, manufacturing specific use cases and particularities in manufacturing infrastructures have not been addressed so far. With our work, we fill this gap by providing guidance for technically integrating RFID in manufacturing.

With the presented architectural patterns we provide concrete design option for integrating RFID in manufacturing IT systems. The presented patterns provide solutions to reoccurring questions in the system design for integrating RFID at manufactures. Note that the patterns also provide solutions for application independent from RFID (e.g., in bar code driven applications). Specifically, our the patterns answer the following three design questions for integrating RFID in manufacturing: (1) What kind of filter logic should run where in the system? (2) Where should which control functionality be located in the system? And (3) How should process relevant data traverse through the process? Our patterns provide candidate designs along with guidance when to opt for which solution.

References

Bornhövd, C., Lin T., Haller S., and Schaper J. (2004). Integrating automatic data acquisition with business processes – experiences with SAP's Auto-ID infrastructure. In Proceedings of the VLDB – Very Large Data Bases, pp.1182–1188.

Brandl (ed.) (2000). ANSI/ISA-95.00.01 enterprise-control system integration part 1: Models and terminology. Technical report, ISA Research Triangle Park, North Carolina, USA.

Chappell, G., L. Ginsburg, P. Schmidt, J. Smith, and J. Tobolski (2003).Auto-ID on the line: The value of Auto-ID technology in manufacturing. Technical report, Auto-ID Center.

Coral8 (2006). Complex event processing: Ten design patterns. Technical report, Coral8 Inc.

EPCglobal. (September 2007). The EPCglobal Architecture Framework - Version 1.2,

2007. http://www.epcglobalinc.org/standards/architecture/.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J.M. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Ivantysynova, L., Ziekow, H., Günther, O., Kletti, W., and Kubach U. (2008a). Six Case Studies. In O. Günther, W. Kletti, and U. Kubach (Eds.), RFID in Manufacturing (1st ed.), Springer Berlin Heidelberg, 2008, pp. 61-112.

Ivantysynova, L., Ziekow, H., Günther, O., Kletti, W., and Kubach U. (2008b). Lessons Learned. In O. Günther, W. Kletti, and U. Kubach (Eds.), RFID in Manufacturing (1st ed.), Springer Berlin Heidelberg, 2008, pp. 61-112.

Ivantysynova L., Klafft M., Ziekow H., Günther O., and Sekin K. (2009). RFID in Manufacturing: the Investment Decision. In Proceedings of the Pacific Asia Conference on Information Systems, India.

Jeffery, S., G. Alonso, M. Franklin, W. Hong, and J. Widom (2006). Declarative support for sensor data cleaning. In Proceedings of the Pervasive Conference, pp. 83–100.

Lee Y. M., Chend F., Leung Y.T. (2004). Exploring the Impact of RFID on Supply Chain Dynamics. In Proceedings of the 2004 Winter Simulation Conference (WSC '04), IEEE Computer Society, Washington DC (USA), 1145-1152.

Luckham, D. (2001). The Power of Events: an Introduction to Complex Event Processing in Distributed Enterprise Systems (1st ed.). Addison-Wesley Longman Publishing Co., Inc.

Sabbaghi, A. and Vaidyanathan, G. (August 2008). Effectiveness and efficiency of RFID technology in supply chain management: strategic values and challenges. Journal of Theoretical and Applied Electronic Commerce Research,3(2), 71-81.

Shaw M., and Garlan D. (1996). Software Architecture: Perspectives on an Emerging Discipline. Addison-Wesley

Sheridan T.B. (2002). Humans and Automation: System Design and Research Issues. John Wiley & Sons, Inc. New York, NY, USA.

Strüker, J., Gille D., and Faupel T. (2008). RFID report 2008. Technical report, Alber-Ludwig University, Friedrichstr. 50, Freiburg im Breisgau, Germany.