

**MBA International Business – Publikationen**

**Development of an Interface for the Synchronization  
of Delivery Dates in an ERP Migration Phase**

Dipl.-Ing. (FH) Christian Kath, M.Sc.

Prof. Dr. Michaela Gläß, MBA

veröffentlicht in:  
LIBDOC Publikationsserver  
der Westsächsischen Hochschule Zwickau

DOI: [10.34806/mba-int-016](https://doi.org/10.34806/mba-int-016)

veröffentlicht durch:  
Westsächsische Hochschule Zwickau  
Fachgruppe Unternehmensführung  
Scheffelstraße 39  
08066 Zwickau

### **Search Terms**

*ERP Migration, Delivery Date Synchronization, PPC Systems (Production Planning and Control), API-based Integration, Heterogeneous System Landscapes, Microservices Principles, Time-Delayed Validation*

### **Executive Summary**

*This paper focuses on designing a robust interface solution for synchronizing critical delivery dates during a complex ERP migration phase, necessitated by the parallel operation of a legacy system and a new target system. The core challenge is that automatic rollback operations triggered between the synchronized ERP systems—due to technical errors—are systematically not communicated to the downstream PPC systems (Production Planning and Control systems). This data inconsistency leads to significant business risks, including production disruptions, delivery delays, and increased inventory costs. The developed solution implements an API-centric approach based on Microservices principles to ensure loose coupling between the systems. It utilizes a two-stage synchronization mechanism to proactively resolve these inconsistencies. First, a time-delayed individual validation employs a strategic 30-minute buffer time to guarantee that potential ERP rollback operations are completed before consistency checks with the PPC system begin. Second, a comprehensive nightly consistency validation systematically reviews all business-relevant delivery dates within a defined time window and initiates automatic corrections. The system's status-based processing model, combined with comprehensive logging and a "Live-Feed" Web Interface, guarantees transaction security and complete traceability. This approach successfully minimizes business risks during the critical transition and offers an effective design pattern for handling rollback situations in similar complex, heterogeneous integration projects.*

## Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>5</b>
<b>2</b>	<b>Theoretical Framework .....</b>	<b>6</b>
2.1	Fundamentals of ERP and PPC Systems.....	6
2.1.1	Historical Development and Functional Scope .....	6
2.1.2	Integration as a Core Feature.....	6
2.1.3	Production Planning and Control (PPC).....	7
2.1.4	ERP in the Context of SMEs .....	8
2.2	Challenges in ERP Migrations .....	8
2.2.1	Project Management and Resources .....	8
2.2.2	Organizational and Process Transformation .....	9
2.2.3	Technical Aspects and Data Management .....	9
2.3	Interface Development in Heterogeneous System Landscapes .....	10
2.3.1	Necessity and Challenges of Integration .....	10
2.3.2	Development of Integration Approaches in Complex System Landscapes.....	11
<b>3</b>	<b>Requirements Analysis of the Interface Solution .....</b>	<b>13</b>
3.1	System Landscape and Initial Situation .....	13
3.2	Problem Definition and Critical Data Flows.....	14
3.3	Functional Requirements .....	15
3.3.1	Consistency Monitoring and Automatic Data Correction .....	15
3.3.2	System Monitoring, Failover Safety, and Configurability.....	15
<b>4</b>	<b>Conception and Implementation of the Interface Solution.....</b>	<b>16</b>
4.1	Architecture Design and System Components .....	16
4.1.1	Architectural Principles .....	16
4.1.2	Central System Components .....	16
4.1.3	Integration into the Existing System Landscape .....	17
4.1.4	Technological Basis and Extensibility .....	17
4.2	Data Model and Status Management .....	18
4.2.1	Structure of Delivery Date Data .....	18
4.2.2	Transaction Status and Processing Logic.....	18
4.2.3	Data Flow and Processing Phases .....	19
4.3	Synchronization Mechanisms and Consistency Check .....	20
4.3.1	Time-Delayed Individual Validation of Staging Transactions .....	20
4.3.2	Comprehensive Nightly Consistency Validation.....	21
4.4	Monitoring, Logging, and Configuration.....	22
4.4.1	Logging Concept and Data Management.....	22
4.4.2	Monitoring Interface .....	22
4.4.3	System Configuration and Operational Parameters .....	23
<b>5</b>	<b>Summary and Outlook.....</b>	<b>24</b>
5.1	Summary.....	24

5.2 Outlook .....	24
<b>Bibliography.....</b>	<b>25</b>

# 1 Introduction

The phased implementation of a new ERP system presents companies with significant technical, organizational, and financial challenges. Often, a legacy system continues to run in parallel with the target system for an extended period to avoid interrupting the ongoing operations of a manufacturing company. Concurrently, many companies have specialized PPC (Production Planning and Control Systems) or MES (Manufacturing Execution System) systems that remain dependent on master data and, in particular, delivery dates from the legacy ERP system. If delivery dates need to be changed in the legacy system for operational reasons and then transferred to the new ERP system, technical errors in the migration process or the target environment can lead to the necessity of rolling back dates in both ERP systems. Due to insufficient synchronization mechanisms, these rollbacks of delivery dates are not transmitted to the PPC or MES system, which then continues to operate with outdated data. The resulting inconsistencies pose a risk of extensive production disruptions, delivery delays, and increased inventory costs.

Numerous studies confirm that ERP projects frequently fall behind schedule and budget. For instance, around 40 % of all implementations experience time delays, primarily due to unclear requirement specifications and inadequate budget planning [1]. A case study [2] by the University of Ulm on a medium-sized automotive supplier illustrates how poorly defined process interfaces between the legacy and new systems can lead to pauses of several months. The lack of goal prioritization and heterogeneous data models increased the migration effort by up to 30 %.

Cost control also proves to be a critical success factor: A meta-analysis [3] identifies training expenses (22–35 % of the budget), interface adjustments (15–25 %), data migration (10–18 %), and unforeseen customizing (12–20 %) as major cost drivers. Companies without a clear cloud migration path face average budget overruns of 17 %. Against this backdrop, phased migration approaches (the "wave principle") have proven more effective than the big-bang model [4], as they reduce failure risks by up to 40 %—for example, through comprehensive data cleansing and the parallel operation of hybrid test environments.

Particularly in a phase of parallel operation, the consistency of data such as delivery dates demands the utmost attention: incorrect dates can lead to production downtimes in 8–12 % of cases and a 15–20 % increase in capital tied up in excess inventory. A documented case at an automotive supplier resulted in losses of € 2.4 million due to a lack of delivery date synchronization [2]. With this in mind, this paper aims to design a robust, API-based (Application Programming Interface) interface that supplies a PPC system with current delivery dates and sustainably prevents inconsistencies through integrated error handling and monitoring.

## **2 Theoretical Framework**

### **2.1 Fundamentals of ERP and PPC Systems**

Enterprise Resource Planning (ERP) systems are now widespread in large, small, and medium-sized enterprises (SMEs) alike [4]. They are often regarded as the heart or backbone of many companies' IT landscapes. An ERP system is an integrated application software composed of multiple components, which encompasses the administration, scheduling, information, and analysis of all resources necessary for executing business processes. These resources include, for example, equipment, personnel, and capital.

The use of modern ERP systems not only supports internal, integrated business processes but also involves suppliers and customers in a company's value chain. Since the early 1990s, many companies have shifted their IT strategy from inhouse development of business application systems to purchasing standard business software, such as ERP systems. Key drivers for implementing an ERP system include potential cost optimization and the technical and process-related integration of all business areas within the company.

#### **2.1.1 Historical Development and Functional Scope**

The ERP concept adopts a cross-functional approach that evolved in the 1990s with the increasing focus on business process orientation and optimization. The concept originated from earlier systems, such as Material Requirements Planning (MRP) [5]. With the evolution to MRP II, the functional scope was expanded by integrating manufacturing with other functional areas, such as accounting and human resources. In this process, the focus shifted from planning business operations to control and management tasks.

Around the turn of the millennium, a new generation of ERP systems known as ERP II [4] emerged. ERP II systems shifted the focus from internal business processes to the entire value chain of a company. Intercompany processes, in the sense of Supply Chain Management (SCM) on the partner and supplier side, as well as Customer Relationship Management (CRM) on the customer side, gained importance. Today's ERP systems adapt these open and cross-company system characteristics, even though the term ERP II has not become widely established. ERP systems are sometimes considered hybrids that combine features of different system types.

The functional scope of ERP systems essentially covers core functional areas such as materials management, production, finance and accounting, controlling, research and development (R&D), marketing, sales, human resources (HR), and master data management. They cover, for example, tasks and processes in procurement (requirements planning, materials management, warehousing, purchasing, goods receipt, SCM, SRM), production (resource scheduling, manufacturing coordination), and sales (order entry, availability check, shipping, CRM).

#### **2.1.2 Integration as a Core Feature**

A crucial feature of ERP systems is the integration of various functions, tasks, and data into a common information system with central data storage [4]. This enables integrated business processes and avoids data redundancy (*Fig. 1*).

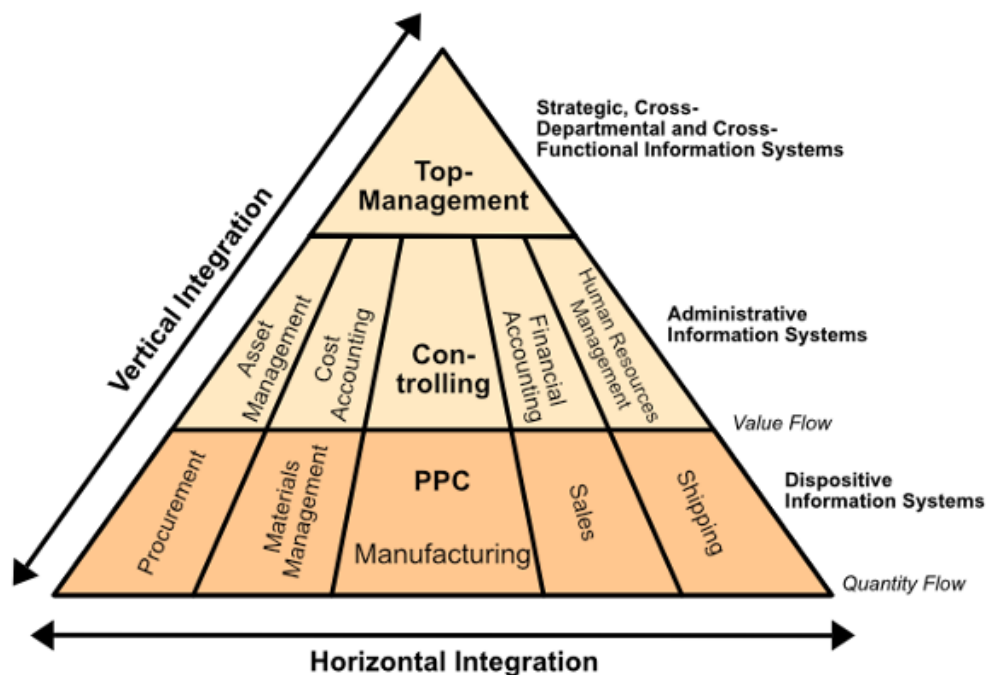


Figure 1: Integration system pyramid [6]

Integration can occur at different levels:

- **Horizontal Integration:** Merging information between functional areas like production and procurement to make it available to different departments.
- **Vertical Integration:** The ability to merge and network information across different levels of aggregation, e.g., between operational activities and management. However, ERP systems are enormously complex due to the integrated database that structures many process variants.

### 2.1.3 Production Planning and Control (PPC)

Production Planning and Control (PPC) is an important topic in business administration that deals with organizing business thinking in production [7]. Conventional PPC systems are often product-oriented and designed to accommodate specific production structures such as job-shop, flow-shop, or JIT organizations. These segment-specific planning and control concepts are ideally embedded in a higher-level framework plan.

During the historical development of business software, which led from MRP to MRP II and finally to ERP, the functional scope was expanded by integrating manufacturing with other functional areas. This means that PPC functionalities are now typically part of ERP systems as integrated modules, covering essential operational processes in production.

Nevertheless, fragmented legacy systems, isolated solutions, or custom IT solutions are still found in companies, especially in the SME sector. These may include specific functionalities for production planning and control and often provide the impetus for introducing an integrated ERP system to resolve system and media breaks and to optimize processes. Although the goal of an ERP implementation is often a single leading system, it may be necessary to integrate additional applications if requirements are not fully supported by the ERP system.

#### **2.1.4 ERP in the Context of SMEs**

Small and medium-sized enterprises (SMEs) have specific characteristics, such as often limited resources, especially in personnel, which often has a direct impact on the project course of an ERP system implementation or migration. Furthermore, a lack of process knowledge and business know-how can be a characteristic feature. For this reason, SMEs often have shied away from implementing classic (on-premise) ERP systems due to high implementation and license costs, as well as the risk that they might be unsuitable. However, the Software-as-a-Service (SaaS) delivery model allows for renting the software and promises a simpler and more affordable implementation. Nevertheless, there is still a need for SMEs to catch up in the market penetration of standard ERP software, as some still use custom IT solutions.

The motivation for SMEs to implement ERP systems is diverse. Common internal triggers are the replacement of fragmented legacy systems or isolated solutions and the long-term integration of a standardized and extensible IT platform. External initiators, such as demands from business partners or legal requirements, can also play a role. A key requirement for SMEs in ERP implementation is the greatest possible alignment between the company's business processes and the processes mapped in the software, to minimize the effort for business process restructuring or technical adjustments.

### **2.2 Challenges in ERP Migrations**

The implementation or switch to a new ERP system is a complex and risky project that ties up significant human and financial resources. The motivation for such a change often lies in outdated systems, missing functionality, lack of vendor support, or the demands of digitalization. Successful implementation requires a structured and controlled approach. Various central challenges arise that go beyond purely technical conversion.

#### **2.2.1 Project Management and Resources**

ERP migrations are inherently large-scale projects that place high demands on project management. Requirements gathering is a critical but often inadequately performed step, which can lead to an incomplete functional scope and necessitate rework or scope changes during the project. Project management activities such as scope, human resources, risk, procurement, and integration management are often underestimated.

One of the biggest challenges, especially for SMEs, are resource constraints. Personnel availability is often low because employees are already heavily burdened by daily operational tasks. However, sufficient release from operational duties is essential for successful project execution. This is particularly true for Key Users, whose time commitment is often underestimated. Besides human resources, tight financial resources are also a typical weakness of SMEs in a project context. The lack of experience with ERP systems or IT projects, as well as a deficiency in process knowledge and business knowledge, further complicates the implementation. The team composition and the decision-making autonomy of those involved also play an important role in project success. The successful Go-Live itself is a complex undertaking that requires intensive preparation and troubleshooting.



### **2.2.2 Organizational and Process Transformation**

The introduction of a new ERP system almost always involves a restructuring of company processes, especially with standardized solutions. This often requires a change in mindset within the company and can lead to internal resistance from employees who want to stick to established processes. Inefficient Business Process Reengineering (BPR) and inadequate Change Management are therefore critical success factors that persist regardless of the technology delivery model (conventional vs. SaaS). The adaptation of processes to the software, rather than the reverse, is a fundamental yet challenging requirement for standardized systems.

Another central point is knowledge transfer and employee training. Challenges here lie in the quality of documentation and training materials. With modern systems, especially SaaS solutions, a high degree of customer self-service is often expected, which requires corresponding motivation and time allocation. Employees may have anxieties or fear making mistakes in the new system. Management plays a crucial role in communicating the strategic goals and the overall benefits of the new system to minimize resistance and motivate users. Organizational changes, such as clarifying responsibilities and reporting lines, are also necessary and pose a challenge. Many of these organizational challenges are, according to [4], highly dependent on the specific characteristics of the company and the application context, rather than the technology itself.

### **2.2.3 Technical Aspects and Data Management**

Data migration from legacy systems is a critical and often time-consuming step. Challenges arise in the identification, extraction, cleaning, and consolidation of data that often exists in heterogeneous sources (such as spreadsheets). Manual entry carries the risk of errors and media breaks. The use of migration tools and templates can also be complicated by insufficient manageability, discrepancies between system structure and template, and missing options for data maintenance. The use of improved tools, such as API-based extraction capabilities and mass processing functions, is recommended.

If the new ERP system does not replace all previous applications, integration with remaining legacy systems or other specialized solutions is necessary. This requires the design and implementation of new interfaces, which increases the complexity of the project. System configuration, particularly the adaptation of standard processes, also poses a technical challenge that requires deep process understanding and expertise. Inexperienced consultants or insufficient knowledge on the client side can lead to difficulties here. Although modern, standardized systems rely on predetermined "Best Practices", the "fine-tuning" to the specific corporate conditions requires careful work.

It can be concluded that the challenges in ERP migrations are diverse and encompass project management, organization, and employees, as well as technical and data-related aspects. Many of the critical factors are closely linked to the specific circumstances and maturity of the migrating company.

## **2.3 Interface Development in Heterogeneous System Landscapes**

### **2.3.1 Necessity and Challenges of Integration**

When replacing an existing ERP system with a new solution, the company faces the complex task of ensuring a seamless migration without jeopardizing ongoing business operations. This challenge becomes particularly apparent when the legacy system and the new ERP system must operate in parallel for an extended period. In this critical phase of ERP migration, the development and operation of suitable interfaces between the systems are not only necessary but business-critical.

The actual system conversion occurs in phases, with individual modules or business areas migrated successively. During this transition phase, both systems must be kept synchronized to ensure data consistency and maintain business processes. For instance, a sales application in the new system must be able to access current inventory levels still managed in the legacy system. Simultaneously, delivery dates must be synchronized between both systems to enable reliable customer statements.

The challenges of this integration are diverse and complex. A central problem lies in the diversity and heterogeneity of the systems. The legacy system to be replaced and the new ERP system are often developed in different programming languages, are based on different technological platforms, and follow divergent paradigms in modeling business processes [8]. Connecting such heterogeneous applications requires a deep understanding of both the technical architecture and the underlying business logic of both systems.

The limited control over the existing systems is particularly challenging. Migration projects frequently face the problem that the legacy system can only be modified slightly or not at all, often because it is a legacy, poorly documented system or standard software that must not be altered. This restriction forces the integration solution to compensate for deficits, peculiarities, or design decisions of the legacy system. Often, complex workarounds must be developed to utilize inadequate or poorly documented interfaces.

Another significant problem is the lack of standards and semantic differences [8]. Despite the widespread need for integration, few universal standards have been established. While technical standards like XML or JSON exist for data representation, a common syntax does not necessarily imply common semantics. The legacy system and the new ERP system may have different definitions for seemingly simple concepts like "customer," "order," or "delivery date". Harmonizing these semantic differences proves to be a particularly difficult and time-consuming task, requiring both business and technical decisions and often involving protracted coordination processes between different specialized departments.

The complexity and organizational impact of integration further intensify these challenges. Migration requires significant adjustments in corporate culture and IT departments, as the customary isolated way of working must be abandoned. The networking between the old and new systems increases the complexity of the overall system exponentially and can lead to unpredictable interactions. Flawed integration solutions can have a cascading effect on both systems and cause significant financial losses, especially when critical business processes like delivery date planning are affected.

Finally, the operation and maintenance of the integration solutions are particularly complex due to the mixing of different technologies and the distributed nature of the solution. Monitoring, error diagnosis, and performance optimization must be carried out across systems, requiring specialized tools and

expertise. Additionally, integration solutions must be synchronized with the different life cycles of both systems, which can lead to significant coordination effort during updates or adjustments.

A central goal in overcoming these challenges is achieving "loose coupling" between the legacy system and the new ERP system. Loosely coupled systems minimize their dependencies on each other, allowing each system to develop autonomously without causing issues for the other. This architectural principle helps increase flexibility and maintainability during the migration phase while reducing the risks of system failures and the complexity of maintenance.

### **2.3.2 Development of Integration Approaches in Complex System Landscapes**

The history of system integration is characterized by a continuous paradigm shift away from rigid, centralized approaches towards flexible, decentralized architectures. This evolution reflects not only technological progress but also the changing demands of modern companies for agility, scalability, and speed of innovation. Early data exchange mechanisms like EDI (Electronic Data Interchange) typically led to complex point-to-point integrations that were functional but entailed significant maintenance challenges [9]. Each new system connection required individual development work, resulting in an exponentially growing network of dependencies. This rigidity became an obstacle for companies that needed to react quickly to market changes.

In the late 1990s, Service-Oriented Architecture (SOA) marked a fundamental turning point in integration strategy [10]. SOA introduced the revolutionary concept of loose coupling, where software components communicate via reusable and interoperable service interfaces. These services encapsulate discrete business functions and use standardized communication protocols like WSDL or SOAP/HTTP. This significantly reduced application dependencies and noticeably accelerated both development and integration processes.

In the ERP environment, the Enterprise Service Bus (ESB) established itself as a central complement to SOA principles. As a middleware component, the ESB took over critical functions such as data model transformations, message communication, and intelligent routing between different applications. Despite their success in standardizing communication, ESBs revealed their structural weaknesses with increasing integration volume. The central architecture led to bottlenecks, and changes to the ESB middleware could impact other integrations. This necessitated complex testing cycles and impaired developer productivity, leading to the need for even more flexible approaches.

These challenges fostered the emergence of the Microservices architecture, which was further facilitated by the rise of cloud computing and agile development methodologies [11]. While monolithic applications bundle all functions into a single system and ESBs create a central integration point, the Microservices architecture follows a decentralized approach. Complex applications are broken down into smaller, purpose-specific, and loosely coupled services, each operating as an autonomous unit with its own technology stack, specific data management methods, and potentially dedicated databases.

This decentralized structure enables exceptional agility and developer productivity, as teams can selectively integrate new technologies into individual services without affecting the overall system. At the same time, granular scalability allows for optimal resource utilization, as each service can be dimensioned independently based on its specific workload. The decoupling of services leads to increased resilience, as the failure of one microservice does not jeopardize the entire system.

Communication between these decentralized services primarily occurs via RESTful APIs, which have established themselves as the dominant architectural style for modern system integration. REST (Representational State Transfer) is based on fundamental HTTP principles and utilizes standardized HTTP methods for various operations. Core characteristics such as statelessness, uniform interfaces, and resource orientation ensure consistent and intuitive API interactions. Each API request contains all necessary information for its processing without relying on server-side sessions, while APIs organize functionalities around clearly defined resources identified via unique URIs.

In this decentralized ecosystem, the API Gateway takes on a central, yet deliberately lightweight, orchestration role. As an intelligent entry point for client requests, it differs fundamentally from the heavyweight ESB due to its focus on the efficient management and provision of external APIs. The API Gateway handles critical functions such as intelligent request routing, load balancing, and security checks, but it prioritizes the real-time processing of synchronous HTTP-based communication instead of complex internal integration and transformation scenarios.

This evolutionary development clarifies that APIs and Microservices, while complementary, represent conceptually different elements. While Microservices represent an architectural design style, APIs form the technical "nervous system" that seamlessly connects these services. APIs function as digital interfaces through which decentralized components communicate and interact, enabling companies to leverage the full flexibility and speed of Microservices while controlling complexity in development and deployment.

### 3 Requirements Analysis of the Interface Solution

#### 3.1 System Landscape and Initial Situation

The company under consideration is in a critical phase of ERP migration, which was initiated by the approaching end of vendor support for the existing ERP system. The announcement of the discontinuation of support for the currently implemented system version presented the company with the strategic necessity of migrating to the latest generation of the ERP system. To minimize business risks and ensure continuous operation, a controlled parallel approach was chosen instead of a Big Bang migration.

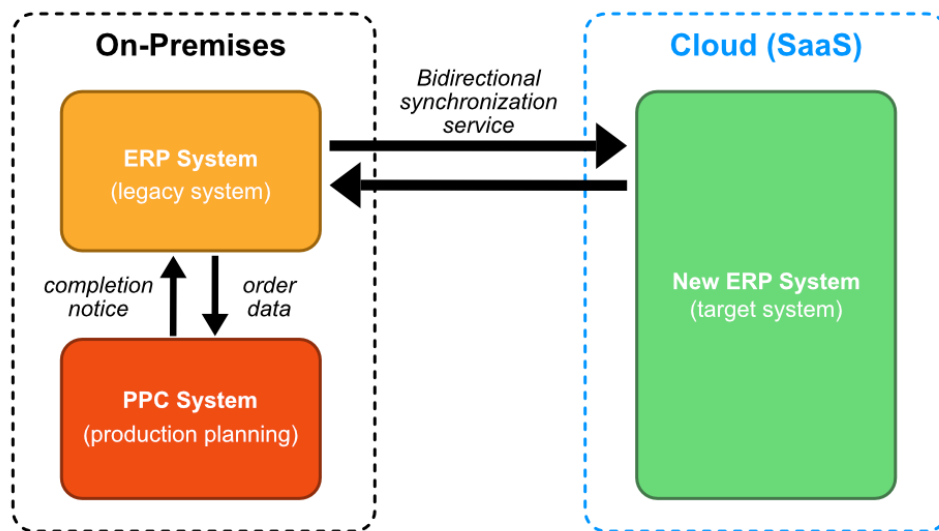


Figure 2: Simplified schematic representation of the ERP system landscape

The current system landscape (*Fig. 2*) is characterized by the coexistence of three central information systems, whose parallel operation during the migration phase is essential for maintaining business processes. This constellation generates both synergy potential and significant challenges regarding data consistency and process stability. The existing ERP system currently functions as the leading system and covers the essential corporate functions, including logistical processes. This system has already replaced the original PPC system in this functional area in the past and assumes a central role as the primary data supplier for the downstream PPC system. Data transmission to the PPC system is unidirectional and includes critical production data such as master data, order information, and delivery dates.

The PPC system is solely responsible for production control and represents a highly specialized solution for production-related processes. Communication with the existing ERP system is unidirectional, with the PPC system merely transmitting completion notifications back to the leading system. These feedback messages are necessary for updating production progress and adjusting planning data accordingly.

The new ERP system is operated as a cloud-based Software-as-a-Service (SaaS) solution and is designed to completely replace both the existing ERP system and the PPC system in the long term. Functionally, the scope of performance corresponds to that of the existing ERP system and ensures all required business functions for the various corporate divisions.

A significant advantage of the current system constellation lies in the structural compatibility between the legacy and target systems. Since the new ERP system represents an evolutionary development of the existing system by the same vendor, both systems exhibit similar internal data structures. This architectural compatibility enabled the implementation of a bidirectional synchronization service that ensures data consistency between both ERP systems. This ensures that critical business data such as delivery dates remain consistent across systems, and the successive migration of business units can be realized with calculable effort.

### **3.2 Problem Definition and Critical Data Flows**

The described system constellation during the migration phase generates a complex dependency structure between the three information systems, which can lead to considerable inconsistencies during the synchronization of critical business data. The handling of delivery dates proves to be particularly problematic, as they are of fundamental importance for production planning and control.

The critical data flow for delivery dates currently follows the following pattern: Delivery dates are primarily managed in the existing ERP system and, when changed, are transferred to both the new ERP system and the PPC system. The bidirectional synchronization service between the two ERP systems ensures that date changes are kept consistent in both systems. The PPC system receives this information exclusively from the existing ERP system and uses it for production planning and resource disposition.

The central problem arises in handling error situations during synchronization between the ERP systems. When technical errors occur in the migration process or in the target environment—for example, due to blocked data records in the new ERP system—the affected delivery dates are automatically rolled back in both ERP systems to ensure data consistency. This rollback mechanism ensures that transactions are executed consistently on both ERP systems within the scope of synchronization, and both systems always have the same data status, thereby preventing divergent date specifications. However, it is critical that this rollback process is not communicated to the PPC system. The PPC system continues to operate with the originally transferred, but now outdated, delivery dates, while both ERP systems already operate with the reset dates. This inconsistency leads to production planning being based on false assumptions, and production resources possibly being allocated inefficiently.

The consequences of these data consistency problems are diverse and business-critical. Outdated delivery dates in the PPC system can lead to production stoppages if planned material deliveries do not arrive on time or if production capacities are reserved for orders that are no longer relevant. Furthermore, incorrect scheduling results in increased warehousing costs due to overstocking or additional procurement costs due to rush orders. It is particularly problematic that these inconsistencies are often only recognized once operational disruptions have already occurred. The lack of a systematic monitoring and validation mechanism for data consistency between all three systems complicates the early identification and resolution of such problems. Manually checking and potentially correcting data discrepancies ties up considerable personnel resources and is prone to errors.

The problem described highlights the necessity of a robust interface solution that detects any rollback operations between the ERP systems and triggers corresponding corrections in the PPC system. Such a solution must encompass both technical integration and the business logic for handling error situations to ensure data consistency across all system boundaries.

### **3.3 Functional Requirements**

Based on the identified problem, the functional requirements for the interface solution to be developed must be defined. These requirements aim to eliminate the described inconsistencies between the systems and ensure robust data integration.

#### **3.3.1 Consistency Monitoring and Automatic Data Correction**

The interface solution must be able to identify and automatically correct inconsistencies in delivery dates between the existing ERP system and the PPC system. This includes, in particular, the detection of rollback situations where dates were reset between the ERP systems, but these changes were not forwarded to the PPC system. The solution must support both timely and periodic consistency checks to cover various error scenarios. It must be ensured that all relevant delivery dates can be systematically checked for consistency. When inconsistencies are identified, the solution must be able to initiate automatic corrective measures, whereby the existing ERP system functions as the leading system, whose data status serves as a reference for the correction in the PPC system. The correction processes must be able to occur without manual intervention in order to minimize reaction time and reduce operational disruptions. Furthermore, it must be ensured that corrections are only carried out when inconsistencies are actually identified, and that the entire process runs deterministically and traceably.

#### **3.3.2 System Monitoring, Failover Safety, and Configurability**

The solution must fully log all performed consistency checks and corrective measures, including the documentation of successful synchronizations and the recording of identified discrepancies and corrections carried out. It must be possible to track the status of consistency checks and distinguish between already processed data records and those yet to be checked. The interface solution must offer transparency regarding its activities and grant authorized users insight into the operations performed. This includes both current processing statuses and historical data on consistency checks and corrections. The system must provide features that allow for the evaluation of the frequency of inconsistencies.

The solution must be designed to be robust against various failure scenarios and exhibit deterministic behavior in the event of system errors. In the event of temporary failures of individual components, the system must be able to automatically resume operation after recovery. There must be no data loss, even if failures occur during processing. The system must manage its internal state so that incompletely processed operations can be identified and correctly concluded.

Furthermore, the interface solution must support configurable parameters to meet various operational requirements. This includes, in particular, the adjustment of inspection intervals according to specific business requirements. The system must be flexible enough to be able to react to changed framework conditions without requiring extensive system modifications.

## 4 Conception and Implementation of the Interface Solution

### 4.1 Architecture Design and System Components

The development of a robust interface solution for the synchronization of delivery dates requires a well-thought-out architecture that meets both the technical requirements of the heterogeneous system landscape and the operational requirements for reliability and transparency. The conceived solution is based on an API-centric approach that is oriented toward modern Microservices principles and ensures loose coupling between the participating systems.

#### 4.1.1 Architectural Principles

The architectural design follows the principle of loose coupling between the existing ERP system and the PPC system. Instead of creating direct system connections that would lead to strong dependencies, the interface solution functions as an intelligent intermediary that encapsulates the complexity of system integration while maintaining the autonomy of the participating systems. This decoupling makes it possible to make changes to individual systems without affecting the overall integration logic. A central aspect of the architecture is the implementation of an event-driven approach for consistency monitoring. The solution responds both to explicit change notifications from the ERP system and to time-controlled consistency checks to cover various error scenarios. This hybrid strategy ensures that both immediate inconsistencies and creeping data deviations can be detected and corrected.

#### 4.1.2 Central System Components

The interface solution consists of three complementary components, each performing specific functions in the overall architecture. The service of the interface solution forms the technical core and implements the entire business logic for consistency monitoring and data validation. This RESTful API acts as the central coordinator between the ERP system and the PPC system and performs all necessary data validations and transformations.

The service processes incoming POST requests from the ERP system containing information about delivery date updates (*Fig. 3*). In parallel, the ERP system logs all delivery date changes in its own logging table, thereby ensuring a complete audit trail for all date modifications. Every incoming request is analyzed, validated, and documented by the service. Subsequently, an SQL insert into the staging transaction table of the PPC system is executed with the status "CREATED". This staging table functions as an intermediate buffer and allows for the controlled processing of delivery date changes without directly interfering with the productive delivery date tables of the PPC system. This component implements transaction security and ensures that no information about completed operations is lost, even in the event of system failures.



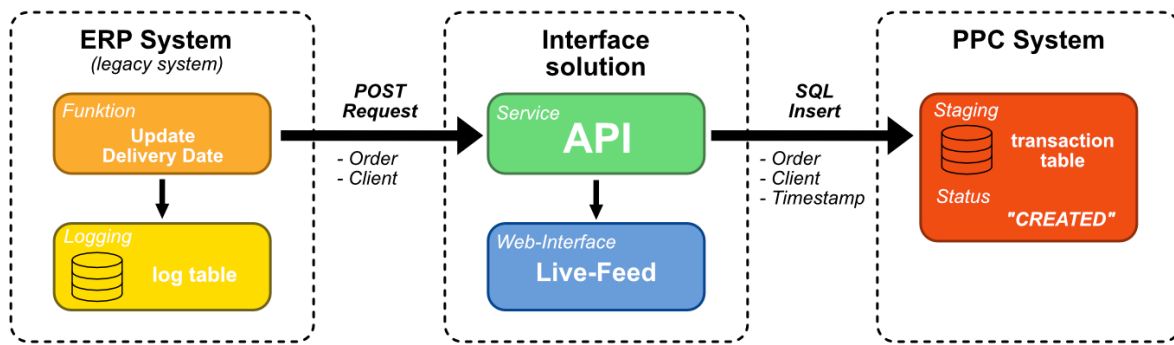


Figure 3: Update notification of the delivery date to the interface solution

The "Live-Feed" web interface represents a user-friendly monitoring interface that grants authorized users real-time insight into the activities of the interface solution. This web application visualizes all delivery date updates in chronological order. The interface allows administrators and specialized users to continuously monitor the status of data integration and make manual interventions if necessary.

#### 4.1.3 Integration into the Existing System Landscape

The interface solution was specifically designed to be integrated into the existing system architecture in a minimally invasive manner. The ERP system is expanded by three specific API functions that enable complete integration. The first function is a POST request endpoint that is automatically activated upon delivery date changes and notifies the interface solution about relevant date modifications. The second API function allows the interface solution service to query current delivery dates directly from the ERP system, utilizing them for subsequent consistency comparisons with the PPC system. The third function represents a callable API interface through which the service can trigger the established standard process for delivery date transfer to the PPC system

Integration with the PPC system is achieved via direct database access to the staging transaction table, thereby eliminating the necessity for extensive interface development on the PPC side. This decision is based on the analysis that the PPC system already provides structured database tables for delivery dates and uses SQL-based operations by default. The transaction table serves as a temporary time buffer and ensures a controlled processing flow, featuring a strategic time delay of 30 minutes between the initial staging and the subsequent consistency check.

This time delay is necessary for handling rollback scenarios between the existing and the new ERP system. Should a rollback operation be triggered due to technical problems in the ERP synchronization, 30 minutes are available to restore data consistency between both ERP systems. After this buffer time has elapsed, the interface solution can assume that the delivery date present in the ERP system is the valid and final date status and use it for the consistency comparison with the PPC system.

#### 4.1.4 Technological Basis and Extensibility

The implementation is based on proven web standards and uses HTTP/HTTPS as the primary communication protocol between the ERP system and the interface solution. JSON is used as the data format for exchanging structured information, as it is both machine-readable and well-suited for debugging purposes. The RESTful API architecture ensures that the solution is compatible with various

client implementations and supports future extensions. For database operations, SQL is used as the standardized query language to ensure maximum compatibility with different database management systems. The architecture also considers the temporary nature of the current system constellation. Upon completion of the ERP migration, the interface solution can be gradually decommissioned or reconfigured for other integration purposes without requiring extensive refactoring measures.

## **4.2 Data Model and Status Management**

Defining a unified data model for the interface solution forms the basis for consistent and reliable synchronization of delivery dates between the involved systems. Due to the structural compatibility between the existing and the new ERP system, and the deliberate focus on delivery dates, the data model proves to be relatively manageable and easy to implement.

### **4.2.1 Structure of Delivery Date Data**

The central data object of the interface solution is the delivery date, which is characterized by three essential attributes. The order number acts as the primary identifier, enabling the unambiguous assignment of delivery dates to specific business transactions. This key ensures cross-system referenceability and forms the basis for all consistency checks between the ERP and PPC systems. The client represents an organizational dimension relevant exclusively for the ERP-side operations. This identifier allows the interface solution to identify the correct corporate context and, if necessary, trigger the corresponding standard process for the delivery date transfer in the correct client of the ERP system. The delivery date is managed as a day-specific value without time information. This granularity corresponds to the practical requirements of the PPC system's production planning, as manufacturing processes are typically planned on a daily basis, and a higher time resolution is not required for delivery date coordination. Using day-specific dates simplifies both data processing and consistency checks, as timezone issues and second-exact synchronization questions are eliminated.

### **4.2.2 Transaction Status and Processing Logic**

The staging transaction table of the PPC system implements a status-based processing model that allows for the controlled and traceable execution of delivery date updates. Newly inserted datasets initially receive the status "CREATED", which signals that the delivery date change was received by the service and is marked for further processing. This status marks the beginning of the 30-minute waiting period intended for potential rollback operations between the ERP systems. Upon successful completion of the consistency check and eventual corrective actions, the status is updated to "COMPLETED", documenting that the delivery date update has been fully processed. Should technical problems or data inconsistencies occur during the processing phase that cannot be automatically resolved, the status is set to "ERROR" to allow for manual postprocessing.

### **4.2.3 Data Flow and Processing Phases**

All datasets in the interface solution are provided with precise timestamps, ensuring complete traceability of the processing sequence and timings. This time information is important for implementing the 30-minute waiting period and allows the system to automatically recognize when staging entries are ready for consistency checking. The combination of order number and timestamp also enables the identification and handling of multiple updates for the same order within short periods. If further date changes for the same order are received during the waiting period, the system can intelligently consolidate them and only use the latest change for the final consistency check.

The data flow through the interface solution occurs in clearly defined phases that ensure both technical robustness and operational traceability. In the first phase, the service receives delivery date changes from the ERP system and creates corresponding entries in the staging table. This phase is deliberately kept lean to enable quick processing of incoming requests and avoid burdening the ERP system with complex validation processes. The second phase involves the time-controlled consistency check, in which the system, after the waiting period has elapsed, retrieves the current delivery dates from the ERP system and compares them with the corresponding data in the PPC system. This separation of processing phases allows the system's immediate responsiveness to be decoupled from the more computationally intensive consistency operations, while simultaneously ensuring the necessary stability in ERP rollback scenarios.

### 4.3 Synchronization Mechanisms and Consistency Check

The synchronization mechanisms of the interface solution are based on a two-stage approach, which includes both timely individual checks and comprehensive periodic validations. This combination ensures continuous data consistency between the ERP and PPC system and covers various error scenarios that may occur during the critical migration phase.

#### 4.3.1 Time-Delayed Individual Validation of Staging Transactions

The first synchronization mechanism (Fig. 4) is based on the time-delayed processing of entries in the staging transaction table of the PPC system.

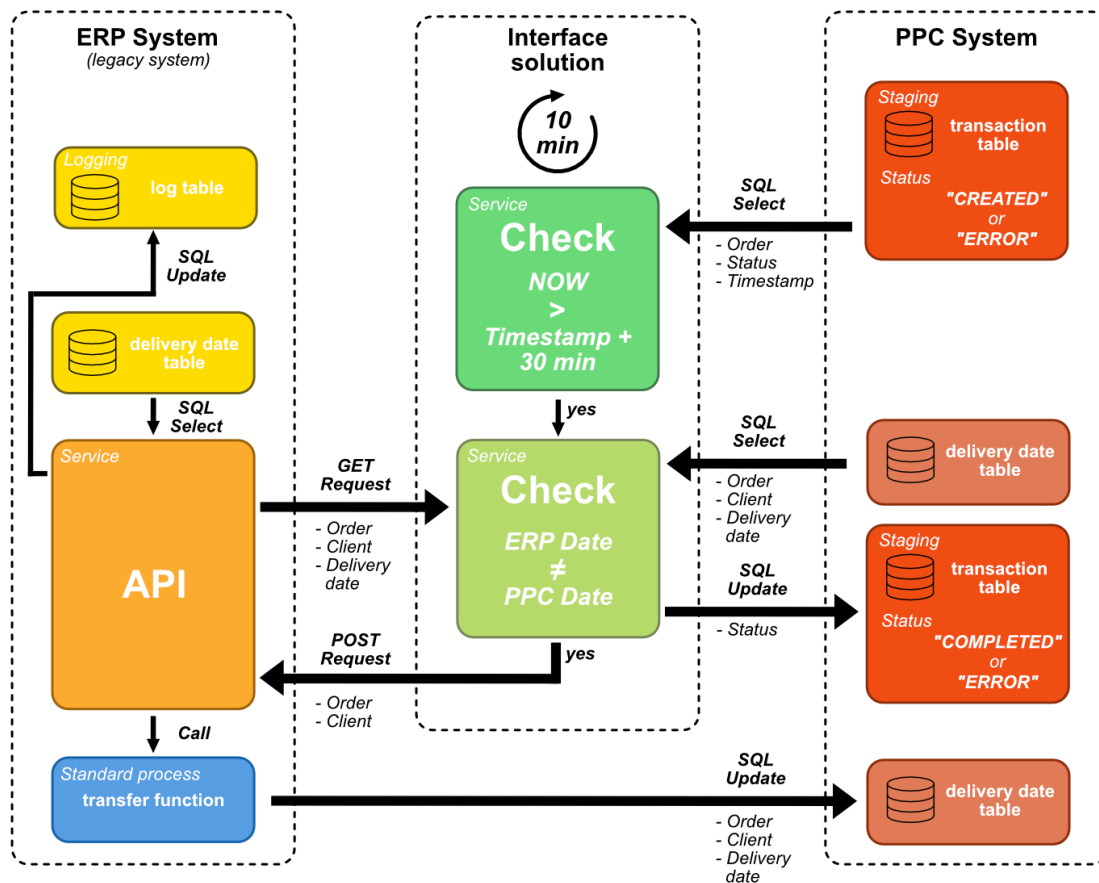


Figure 4: Schematic representation of the first synchronization mechanism of the interface solution

The interface solution performs automated checks at regular 10-minute intervals to identify transactions with the status "CREATED" whose timestamp is at least 30 minutes in the past. This strategic time delay ensures that any rollback operations between the existing and new ERP system are fully completed before the consistency validation is initiated. For each identified staging transaction, the solution initiates a multi-stage validation process. First, the current delivery date for the affected order and client is retrieved via the corresponding API function of the ERP system. Simultaneously, an SQL SELECT command extracts the corresponding delivery date from the

productive delivery date table of the PPC system. These two date values are then compared to identify potential inconsistencies.

If there is no discrepancy between the dates, the status of the staging transaction is updated to "COMPLETED", and processing is concluded. In the event of identified date differences, the solution automatically initiates correction measures. The standard delivery date transfer process in the ERP system is triggered via the corresponding API function, which effects the correct update of the delivery date in the PPC system. After successful transmission, the transaction status is also set to "COMPLETED". Should the ERP system be unreachable due to technical problems, maintenance, or network failures, the transaction status is set to "ERROR". These erroneous transactions are retried during the next 10-minute cycle, ensuring automatic retry in case of temporary failures.

#### 4.3.2 Comprehensive Nightly Consistency Validation

The second synchronization mechanism implements a comprehensive consistency check of all delivery dates between the ERP and PPC systems (Fig. 5).

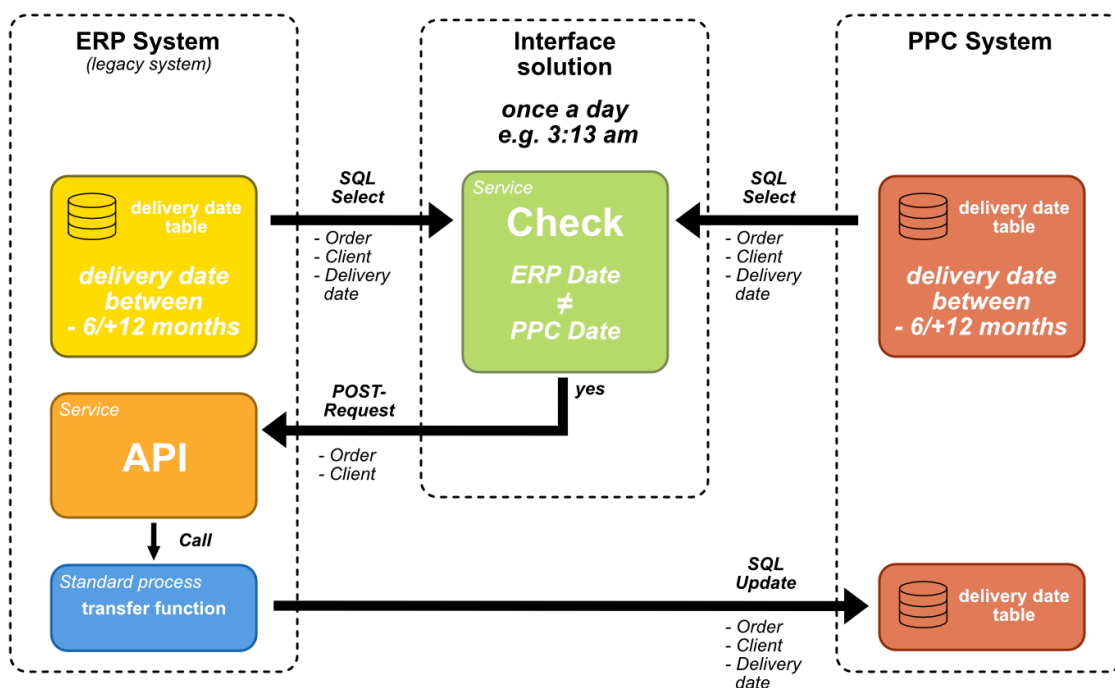


Figure 5: Schematic representation of the second synchronization mechanism of the interface solution

This mass validation is performed daily outside business hours to avoid interfering with operational processes. The nightly execution utilizes time windows with low system load and minimizes potential impacts on the performance of productive systems. The comprehensive consistency check systematically compares all delivery dates within a defined time window of 6 months in the past up to 12 months into the future. This timeframe covers both already processed orders and future plans, ensuring a complete validation of all business-relevant dates. Unlike the individual validation, data access occurs via direct SQL queries to the underlying databases of both systems to optimize performance for large data volumes and circumvent API limitations.

The mass comparison algorithm first extracts all relevant delivery dates from both systems and then performs a systematic reconciliation. In the event of identified inconsistencies, automatic corrections are initiated via the established standard process for delivery date transfer. All adjustments made are documented in a structured CSV file (Comma-Separated Values), which is generated at the end of the nightly process. The CSV file contains detailed information on all identified discrepancies and corrections performed, including the order number, client, and original and corrected delivery dates. This documentation allows administrators and specialized users to obtain a complete overview of all nightly corrections on the next working day and initiate further measures if necessary.

## **4.4 Monitoring, Logging, and Configuration**

### **4.4.1 Logging Concept and Data Management**

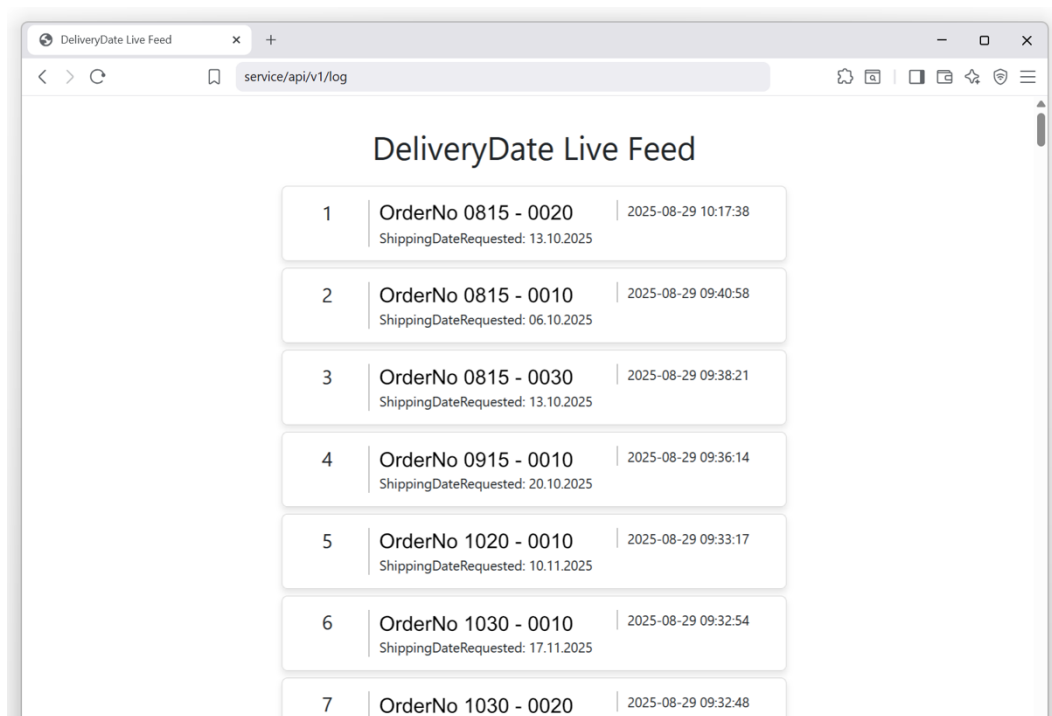
The logging of the interface solution is based on a multi-stage approach that implements various levels of documentation detail on the participating systems. On the ERP side, all delivery date updates are recorded in the system's internal logging table, thereby ensuring complete traceability of all date changes in the leading system. This ERP-side logging forms the basis for subsequent audit processes and enables seamless documentation of data origin.

PPC-side logging occurs via the staging transaction table, which documents all delivery date updates transmitted by the interface, along with corresponding status markers. This table manages the processing status of each transaction and allows for tracking the entire lifecycle of a delivery date update from the initial "CREATED" status up to the final "COMPLETED" or "ERROR" status. The staging table serves not only for logging but also as an operational buffer for time-delayed processing.

On the interface solution side itself, a lean in-memory logging is implemented that exclusively holds the data required for the Live Feed. The in-memory table contains the essential transaction information, such as order number, client, and timestamp, which are needed for real-time monitoring via a web interface.

### **4.4.2 Monitoring Interface**

The web interface "Live Feed" (*Fig. 6*) represents a central monitoring component that grants authorized users continuous insight into the activities of the interface solution. The web application visualizes all processed delivery date updates in chronological order, allowing for an immediate assessment of the system status. Each entry in the Live Feed displays the relevant business data, such as the order number and requested delivery date, along with technical metadata like the processing time. The interface was deliberately designed as a simple yet effective monitoring solution that meets operational requirements given a moderate data volume. The real-time display enables users to immediately recognize processing anomalies without relying on complex monitoring infrastructures.



	OrderNo	ShippingDateRequested	Timestamp
1	0815 - 0020	13.10.2025	2025-08-29 10:17:38
2	0815 - 0010	06.10.2025	2025-08-29 09:40:58
3	0815 - 0030	13.10.2025	2025-08-29 09:38:21
4	0915 - 0010	20.10.2025	2025-08-29 09:36:14
5	1020 - 0010	10.11.2025	2025-08-29 09:33:17
6	1030 - 0010	17.11.2025	2025-08-29 09:32:54
7	1030 - 0020		2025-08-29 09:32:48

Figure 6: Web interface for tracking delivery date updates

#### 4.4.3 System Configuration and Operational Parameters

The interface solution supports various configurable parameters to allow adaptation to specific operational requirements. The central configuration parameter is the time window between staging and consistency check, which is set to 30 minutes by default. This value sufficiently accounts for the processing times needed for rollback operations between the ERP systems and can be adjusted if necessary to accommodate altered system runtimes. Other configurable aspects include the check intervals for time-controlled consistency monitoring as well as the definition of time periods for delivery dates that are included in the consistency comparison. By default, delivery dates are monitored within a time window of 6 to 12 months, although these boundaries can be adjusted according to the company's planning horizons. Configuration is managed via structured configuration files, enabling flexible adjustments without requiring system restarts. This parameterization ensures that the interface solution can be adapted to changing business requirements or system environments without necessitating extensive code modifications.

## 5 Summary and Outlook

### 5.1 Summary

This paper addressed the development of an interface solution for synchronizing delivery dates during a critical ERP migration phase. The investigation clarified that the parallel operation of legacy and target systems entails significant challenges for data consistency, particularly when specialized PPC systems continue to rely on data from the legacy system. The theoretical analysis confirmed that ERP migrations in SMEs are characterized by limited resources, missing standards, and inadequate interface integration. The identified problem showed that rollback operations between synchronized ERP systems can lead to systematic inconsistencies with downstream PPC systems. These data discrepancies pose significant business risks, as they can result in production stoppages and increased storage costs.

The developed interface solution addresses these challenges through a two-stage synchronization mechanism. The time-delayed single validation with a 30-minute buffer time allows for the complete processing of ERP rollback scenarios before consistency checks with the PPC system occur. In addition, the nightly mass validation ensures a systematic review of all business-relevant delivery dates. The API-centric approach is based on modern Microservices principles and ensures loose coupling between the participating systems. The minimal invasiveness of the solution, achieved by utilizing existing API functions and direct database access, significantly reduced the development effort. The status-based processing model, combined with comprehensive logging, ensures transaction security and complete traceability of all synchronization operations.

### 5.2 Outlook

The successful implementation of the interface solution opens various perspectives for further developments and methodical insights that are relevant beyond the immediate use case. For instance, implementation with a parallel processing logic could significantly boost system performance as data volumes increase during the migration expansion to other business areas. The work carried out also confirms the relevance of event-driven architecture approaches for complex integration problems in migration scenarios. The time-delayed validation approach proves to be an effective strategy for handling rollback situations and could serve as a design pattern for similar integration projects. The combination of minimal invasiveness and maximum automation points to a possible path for resource-efficient integration solutions in SMEs. The developed concepts can be transferred to other critical data categories, such as inventory information, and offer companies in similar migration situations a template for their own integration projects.

This paper demonstrates that, through systematic requirements analysis and well-considered architectural design, even complex integration problems in heterogeneous system landscapes can be solved with manageable effort. The developed solution contributes to minimizing business risks during critical migration phases and provides a solid foundation for the company's continued digital transformation.



## Bibliography

- [1] Panorama Consulting Solutions (2013): 2013 ERP Report – A Panorama Consulting Solutions Research Report.
- [2] Rüger, M. (2015): Analyse der Einführung eines ERP-Systems bei einem mittelständischen Unternehmen. Masters thesis. Ulm University
- [3] Kusters, R, Heemstra, F., Jonker, A. (2007): Determining the Costs of ERP Implementation. In Proceedings of the Ninth International Conference on Enterprise Information Systems – DISI. 102-110
- [4] Kienegger, H. (2015) Kritische Erfolgsfaktoren und Herausforderungen von Software-as-a-Service basierten Enterprise Resource Planning Einführungsprojekten – Eine explorative Studie am Beispiel von SAP Business ByDesign. Dissertation. Technische Universität München
- [5] Klaus, H., Rosemann, M., Gable, G. (2000) What is ERP? Information Systems Frontiers, 2(2), pp. 141-162.
- [6] Scheer, A.-W. (1997): Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse (7. Aufl.), Springer, Berlin
- [7] Drexel, A., Fleischmann, B., Günther, H.-O., Stadtler, H.; Tempelmeier, H. (1993). Konzeptionelle Grundlagen kapazitätsorientierter PPS-Systeme, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 315, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel
- [8] Hohpe, G., Woolf, B. (2004): Enterprise Integration Patterns (1. Aufl.), Addison-Wesley, Boston
- [9] Gronau, N., Eggert, S. (2010). Betriebliche Anwendungssysteme - Architektur von ERP-Systemen. Vorlesungsfolien, Lehrstuhl für Wirtschaftsinformatik und Electronic Government, Universität Potsdam
- [10] IBM. Was ist serviceorientierte Architektur (SOA)?  
<https://www.ibm.com/de-de/think/topics/soa>
- [11] API7.ai. (2024). ESB vs. API Gateway: Was ist der Unterschied?  
<https://api7.ai/de/blog/esb-vs-api-gateway>