Technische Hochschule Ingolstadt

Fakultät für Elektrotechnik und Informatik

Zukunft in Bewegung

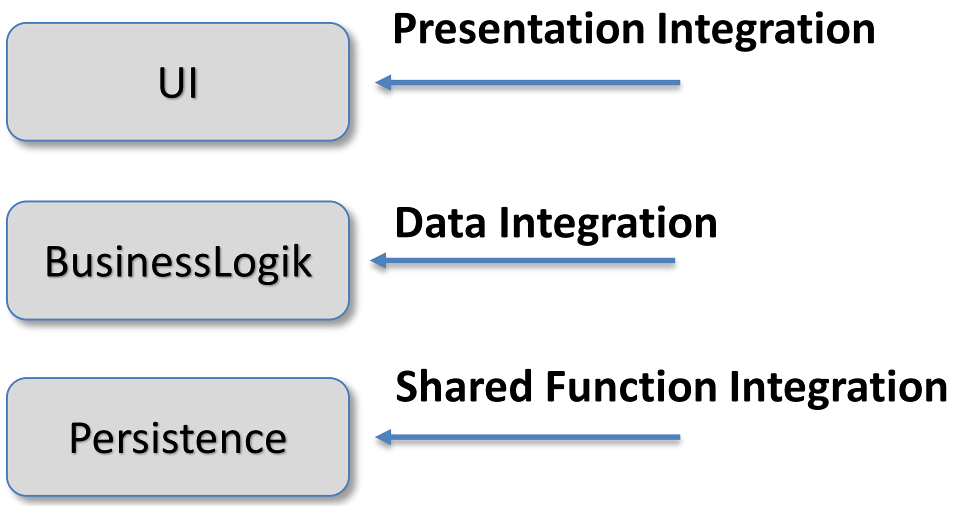
IT-Integrations- und Migrationstechnologien

Integrationsarten und Integrationsschichten

Prof. Dr. Bernd Hafenrichter 19.10.2023

Integrationsarten

- **Motivation:**
  - Um System integrieren zu können ist es notwendig eine Konnektivität zwischen den einzelnen Systemen herzustellen.
  - Im folgenden werden grundsätzliche Ansätze beschrieben welche sich an der klassischen 3-Schicht –Architektur orientieren



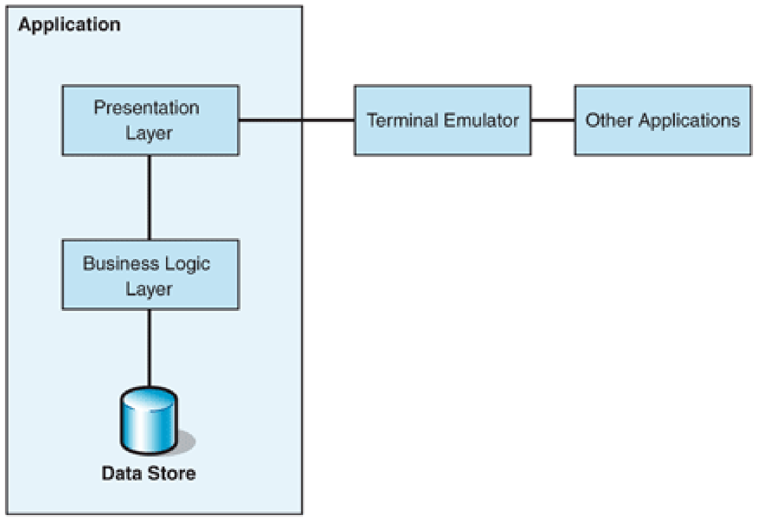
Integrationsarten – Presentation Integration

- **Motivation: Presentation Integration**
  - You have multiple independent applications that are organized as functional silos. Each application has a user interface.

Forces

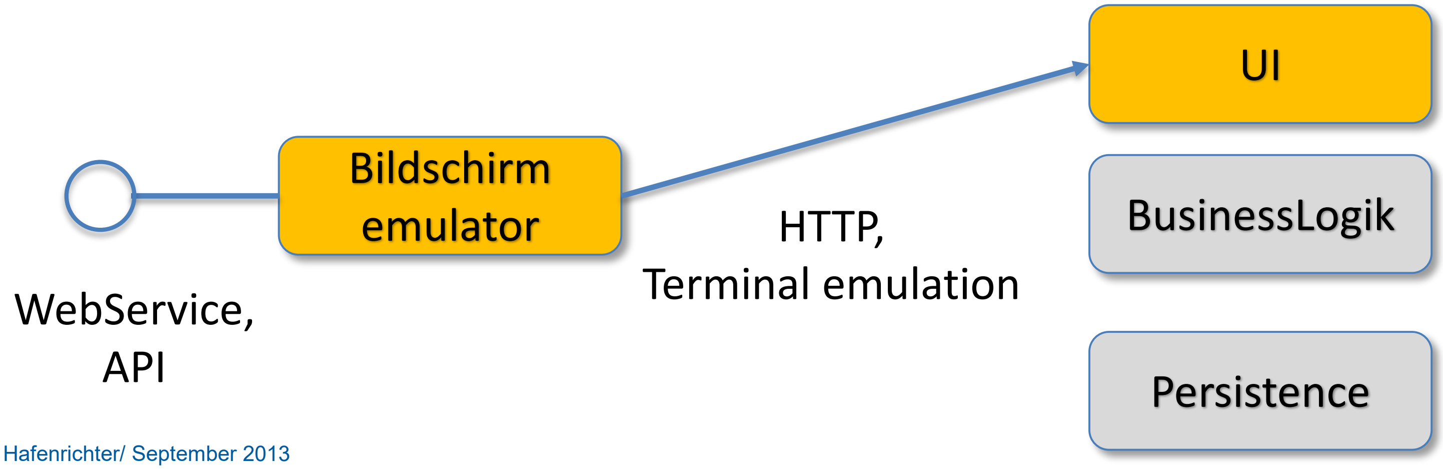
- When integrating multiple applications, you should minimize changes to existing systems because any change to an existing production system represents a risk and might require extensive regression testing.
- Many applications feature little or no separation between business and presentation logic. The only remote components of these applications are simple display and input devices. These display terminals access the central mainframe computer, and the central mainframe computer hosts all business logic and data storage.
- The business logic layer of many applications is programming-language specific and is not available remotely, unless it was developed on top of a specific remoting technology such as DCOM or Common Object Request Broker Architecture (CORBA).
- Directly accessing an application's database layers can cause corruption of application data or functionality.

Integrationsarten – Presentation Integration



Integrationsarten – Presentation Integration

- **Screenscraping/Terminalemulation**
  - Integration von Anwendungen durch Simulation eines Benutzers
  - Ein Bildschirmemulator liefert Eingabe an die zu steuernde Oberfläche
  - Die Ausgabe des integrierten Programms wird abgefangen und strukturiert bereitgestellt.
  - Einsatzbereich: Integration von Mainframes und Legacysystemen, Webanwendung
  - Zugriff wird über eine Standard API bereitgestellt



Integrationsarten - Presentation Integration

Vorteile

- **Low-risk.** In Presentation Integration, a source application is the application that the other applications extract data from. It is unlikely that the other applications that access the source application can corrupt it because the other applications access the data the same way that a user accesses the data. This means that all business logic and validations incorporated into the application logic protect the internal integrity of the source application's data store. This is particularly important with older applications that are poorly documented or understood.
- **Non-intrusive.** Because other applications appear to be a regular user to the source application, no changes to the source application are required. The only change that might be required is a new user account.
- **Works with monolithic applications.** Many applications do not cleanly separate the business and presentation logic. Presentation Integration works well in these situations because it executes the complete application logic regardless of where the logic is located.

Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsarten - Presentation Integration

Problems

- **Brittleness**. User interfaces tend to change more frequently than published programming interfaces or database schemas. Additionally, Presentation Integration may depend on the specific position of fields on the screen so that even minor changes such as moving a field can cause the integration solution to break. This effect is exacerbated by the relative wordiness of HTML.
- **Limited access to data**. Presentation Integration only provides data that is displayed in the user interface. In many cases, other applications are interested in internal codes and data elements such as primary keys that are not displayed in the user interface. Presentation Integration cannot provide access to these elements unless the source application is modified.
- **Unstructured information.** In most cases, the presentation layer displays all data values as a collection of string elements. Much of the internal metadata is lost in this conversion. The internal metadata that is lost includes data types, constraints, and the relationship between data fields and logical entities. To make the available data meaningful to other applications, a semantic enrichment layer has to be added. This layer is typically dependent on the specifics of the source application and may add to the brittleness of the solution.

Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsarten - Portalintegration

Problems

- **Inefficient**. Presentation Integration typically goes through a number of unnecessary steps. For example, the source application's presentation logic has to render a visual representation of the data even though it is never displayed. The terminal emulation software in turn has to parse the visual representation to turn it back into a data stream.
- **Slow**. In many cases, the information that you want to obtain is contained in multiple user screens because of limited screen space. For example, information may be displayed on summary and detail screens because of limited screen space. This requires the emulator to go to multiple screens to obtain a coherent set of information. Going to multiple screens to obtain information requires multiple requests to the source application and slows down the data access.
- **Complex**. Extracting information from a screen is relatively simple compared to locating the correct screen or screens. Because the integration solution simulates a live user, the solution has to authenticate to the system, change passwords regularly according to the system policy, use cursor keys and function keys to move between screens, and so on. This type of input typically has to be hard-coded or manually configured so that external systems can access the presentation integration as a meaningful business function, such as "Get Customer Address." This translation between business function and keystrokes can add a significant amount of overhead. The same issues of complexity also affect error handling and the control of atomic business transactions

Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsarten - Data Integration

Motivation

- Viele Applikationen innerhalb eines Unternehmens benötigen die gleiche Information (z.B. Kundendaten, Vertragsdaten, Materialstamm)
- Das redundante Erfassen dieser Daten ist zeitaufwendig, fehlerhaft und kostenintensive.

Ziel:

- Schaffung eines Datenpools der von mehreren Applikationen gleichzeitig genutzt werden kann.

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Data Integration

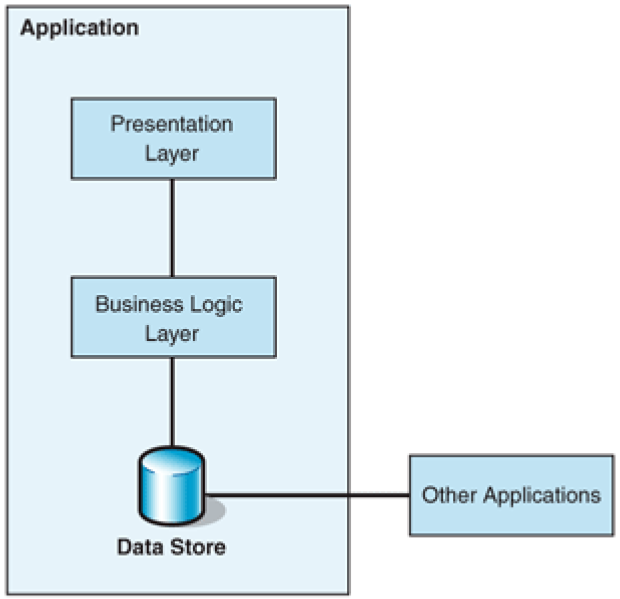
Forces

- When you integrate multiple systems, you usually want to be as noninvasive as possible. Any change to an existing production system is a risk, so it is wise to try to fulfill the needs of other systems and users while minimizing disturbance to the existing systems.
- Likewise, you usually want to isolate applications' internal data structures. Isolation means that changes to one application's internal structures or business logic do not affect other applications. Without isolated data structures, a small change inside an application could cause a ripple effect and require changes in many dependent applications.
- Reading data from a system usually requires little or no business logic or validation. In these cases, it can be more efficient to access raw data that a business layer has not modified.
- Many preexisting applications couple business and presentation logic so that the business logic is not accessible externally. In other cases, the business logic may be implemented in a specific programming language without support for remote access. Both scenarios limit the potential to connect to an application's business logic layer.

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Data Integration

Solution



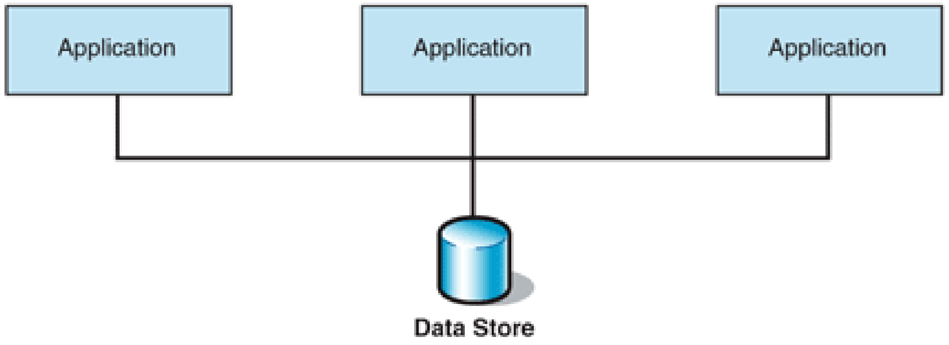
Integrate applications at the logical data layer by allowing the data in one application (the source) to be accessed by other applications (the target)

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Motivation

Direkter Zugriff verschiedener Applikationen auf die gleiche Datenbank



Anmerkungen:

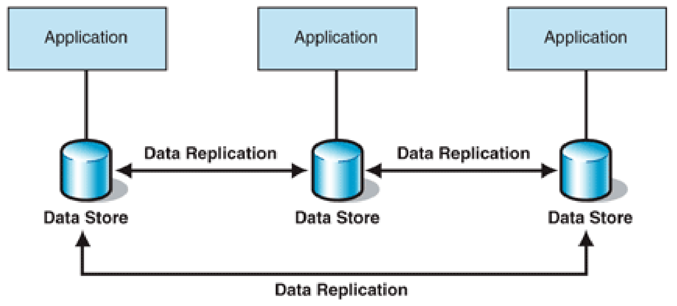
- Geringe Latenz
- Problem: Konsistenz

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Data Replication:

Replikation gemeinsam genutzter Daten zwischen verschiedenen Datenbanken



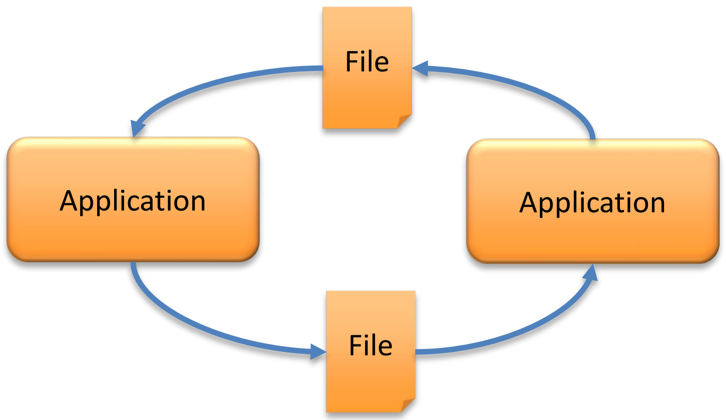
Anmerkungen :

- Kein direkter Zugriff die gleiche Datenbank.
- Replikation der benötigten Daten zwischen den verschiedenen Datenkbanken
- Problem: Latzen & Komplexität

Integrationsarten - Shared Data Integration

File Transfer:

Austausch gemeinsam genutzter Daten auf Basis von Export und Import von Dateien



File Transfer:

- Spezialfall der Daten Replikation
- Universell einsetzbar
- Kein internes Wissen über die Applikationen notwendig
- Latenz

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Kriterien für die Auswahl einer Variante

- Tolerance for data that is not current (*stale data*)
- Performance
- Complexity
- Platform infrastructure and tool support

Integrationsarten - Shared Data Integration

Vorteile

- **Nonintrusive.** Most databases support transactional multiuser access, ensuring that one user's transaction does not affect another user's transaction. This is accomplished by using the Isolation property of the Atomicity, Consistency, Isolation, and Durability (ACID) properties set. In addition, many applications permit you to produce and consume files for the purpose of data exchange. This makes Data Integration a natural choice for packaged applications that are difficult to modify.
- **High bandwidth**. Direct database connections are designed to handle large volumes of data. Likewise, reading files is a very efficient operation. High bandwidth can be very useful if the integration needs to access multiple entities at the same time. For example, high bandwidth is useful when you want to create summary reports or to replicate information to a data warehouse.
- **Access to raw data**. In most cases, data that is presented to an end user is transformed for the specific purpose of user display. For example, code values may be translated into display names for ease of use. In many integration scenarios, access to the internal code values is more useful because the codes tend to more stable than the display values, especially in situations where the software is localized. Also, the data store usually contains internal keys that uniquely identify entities. These keys are critical for robust integration, but they often are not accessible from the business or user interface layers of an application.

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Vorteile

- **Metadata**. Metadata is data that describes data. If the solution that you use for data integration connects to a commercial database, metadata is usually available through the same access mechanisms that are used to access application data. The metadata describes the names of data elements, their type, and the relationships between entities. Access to this information can greatly simplify the transformation from one application's data format to another.
- **Good tool support**. Most business applications need access to databases. As a result, many development and debugging tools are available to aid in connecting to a remote database. Almost every integration vendor provides a database adapter component that simplifies the conversion of data into messages. Also, Extract, Transform, and Load (ETL) tools allow the manipulation of larger sets of data and simplify the replication from one schema to another. If straight data replication is required, many database vendors integrate replication tools as part of their software platform.

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Probleme

- **Unpublished schemas**. Most packaged applications consider the database schema to be unpublished. This means that the software vendor reserves the right to make changes to the schema at will. A solution based on Data Integration is likely to be affected by these changes, making the integration solution unreliable. Also, many software vendors do not document their database schemas for packaged applications, which can make working with a large physical schema difficult.

- **Bypassed business logic**. Because data integration accesses the application data store directly, it bypasses most of the business logic and validation rules incorporated into the application logic. This means that direct updates to an application's database can corrupt the application's integrity and cause the application to malfunction. Even though databases enforce simple constraints such as uniqueness or foreign key relationships, it is usually very inefficient to implement all application-related rules and constraints inside the database. Therefore, the database may consider updates as valid even though they violate business rules that are encoded in the application logic. Use Functional Integration instead of Data Integration if you need the target application to enforce complex business rules.

http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Probleme

- **No encapsulation**. Accessing an application's data directly provides the advantage of immediate access to raw data. However, the disadvantage is that there is little or no encapsulation of the application's functionality. The data that is extracted is represented in the format of an application-internal physical database schema. This data very likely has to be transformed before other applications can use it. These transformations can become very complex because they often have to reconcile structural or semantic differences between applications. In extreme scenarios, the data store may contain information in a format that cannot be used by other systems. For example, the data store might contain byte streams representing serialized business objects that cannot be used by other systems.

Quelle: http://msdn.microsoft.com/en-us/library/ff647273.aspx, 10/2014

Integrationsarten - Shared Data Integration

Probleme

- **Simplistic semantics**. As the name suggests, Data Integration enables the integration of data only. For example, it enables the integration of data contained in entities such as "Customer XYZ's Address." It is not well-suited for richer command or event semantics such as "Customer XYZ Moved" or "Place Order." Instead, use Functional Integration for these types of integration.

- **Different storage paradigms**. Most data stores use a representation that is different from the underlying programming model of the application layer. For example, both relational databases and flat-file formats usually represent entities and their relationships in a very different way from an object-oriented application. The difference in representation requires a translation between the two paradigms.
  - --> Sie Kapitel_2a_Persistenz_03_OO_REL.pdf

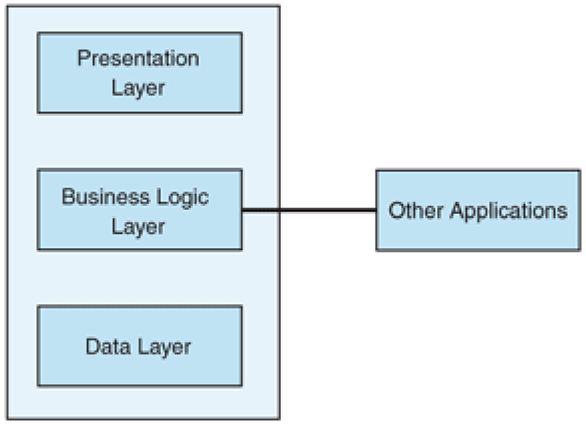
- **Semantic dissonance**. Semantic dissonance is a common problem in integration. Even though you can easily resolve syntactic differences between systems, resolving semantic differences can be much more difficult. For example, although it is easy to convert a number into a string to resolve a syntactic difference, it is more difficult to resolve semantic differences between two systems that have slightly different meanings for the Time, Customer, and Region entities. Even though two databases might contain a Customer entity, the semantic context of both entities might be dramatically different.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

```java
public class BusinessLogic {

    private static final String ROLE_REMOVE_MATERIAL = null;
    private OrderService orderService;
    private MaterialRepo repository;

    public void onRemoveMaterial( String materialNr, long amount ) {

        SecurityContext.checkAccessRights( ROLE_REMOVE_MATERIAL );

        if( amount <= 0 ) {
            throw new RuntimeException( "Amout has to be a positive value" );
        }

        if( materialNr == null || materialNr.isBlank() ) {
            throw new RuntimeException( "Invalid material number specified" );
        }

        MaterialType type = repository.loadMaterialType( materialNr );

        if( type == null ) {
            throw new RuntimeException( "Invalid material number specified" );
        }

        if( type.getCurrentAmount() < amount ) {
            throw new RuntimeException( "Not enougph material on stock" );
        }

        type.setCurrentAmount( type.getCurrentAmount() - amount );

        if( type.reachedCriticalLimit() ) {
            orderService.orderNewMaterial( type );
        }

        repository.save( type );
    }
}
```

Integrationsarten - Shared Function Integration

- **Shared Function Integration**
  - Zugriff auf gemeinsam genutzte Funktionen



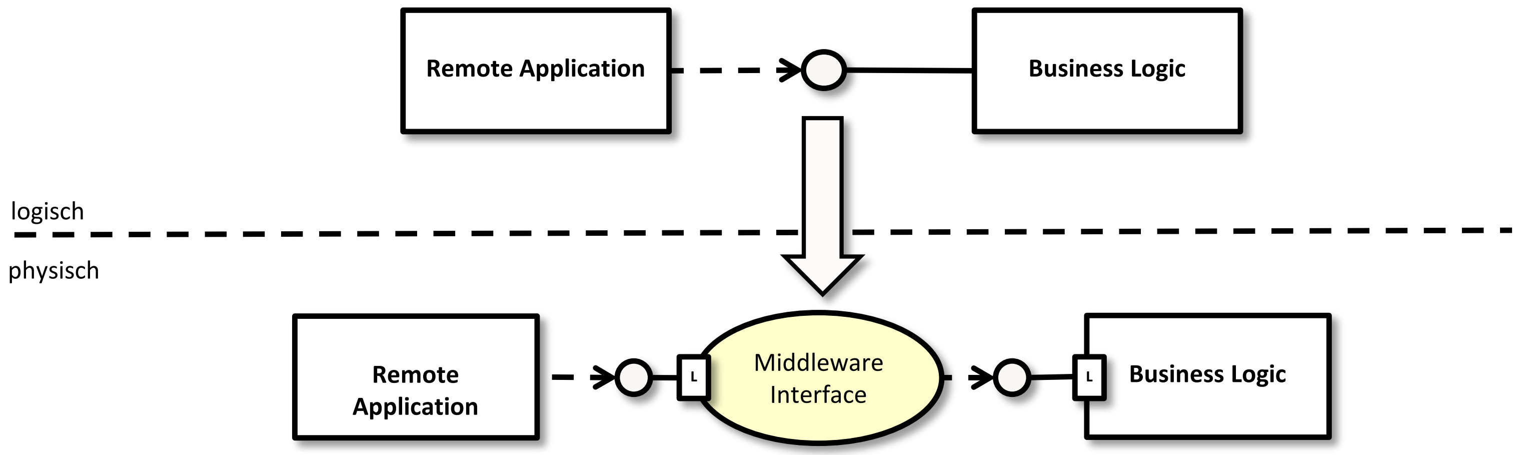
Voraussetzungen:

- Die benötigten Funktionen müssen in der Quellapplikation verfügbar sein
- Direkter (remote) Zugriff auf die API

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsarten - Shared Function Integration

- **Prinzipieller Aufbau**



Komponente	Aufgabe
Business Logic	Lokale Business Logik
Remote Application	Konsumiert Funktionen die in der Bussiness Logic bereitgestellt werden
Middleware Interface	Kommunikation zwischen den Partner herstellen.

Integrationsarten - Shared Function Integration

- **Realisierung**
  - Distributed Object Integration (z.B. DCOM, CORBA, RMI)
  - Message-Oriented Middleware Integration (z.B. Message Queuing)
  - Service-Oriented Integration (z.B. XML und SOAP)

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsarten - Shared Function Integration

Kriterien für die Auswahl einer Variante

- Reliability and latency of the network between endpoints
- Interfaces exposed by your current systems
- Need for interoperability between disparate technical architectures
- Performance
- Fragility, if incompatible updates are introduced to any participant
- Expertise of the technical team
- Existing infrastructure

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsarten - Shared Function Integration

Vorteile

- **Flexibility**. Functional Integration is very flexible. The abstraction, in the form of a function call, permits many different types of interchanges, such as data replication, shared business functions, or business process integration. This also implies that a generic mechanism for Functional Integration can be used in many different integration scenarios.
- **Encapsulation**. A functional interface provides an abstraction from an application's internal workings. This isolates users of the programming interface from variations in the application's internal workings or data representations.
- **Robust**. Executing a function through an application's business logic layer ensures that only well defined functions can be executed. It also ensures that all validations built into the application logic are executed. This ensures that other applications cannot perform invalid functions or corrupt the application's internal state.
- **Familiar programming model**. Functional Integration provides a programming model that is more aligned with widespread application programming models. Functional Integration is therefore familiar to most application developers. Other styles of integration, such as Data Integration or Presentation Integration, require developers to learn a new style of development.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsarten - Shared Function Integration

Problembereiche

- **Tighter coupling**. One application that sends a command directly to another application results in tighter coupling compared to publishing data to a common Message Bus because the requesting application requires information about the location of the application that is providing the service. However, inserting a Message Broker can alleviate this type of coupling.

- **Requires business layer to be exposed**. Naturally, Functional Integration is limited to scenarios where the affected applications expose a suitable functional interface. This is the case for most contemporary business applications, but it may not be true for older monolithic applications or for Web applications hosted by external entities. In those cases, Presentation Integration may be the only option.

- **Limited to available functions**. In most cases, Functional Integration is limited to the functions that are already implemented in the application's business logic. Extending the application's logic with new functions usually involves significant development and may outweigh the other benefits of Functional Integration. In such cases, implementing the new function externally might be a more efficient approach.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsarten - Shared Function Integration

Problembereiche

- **Inefficient with large datasets.** Existing functional interfaces are generally designed to execute individual functions. This typically makes them inefficient for the transmission of large datasets because many individual requests must be made.

- **Programming-language specific.** Many functional interfaces are tied to a specific programming language or technology because the stronger semantics of the conversation require a more detailed contract that includes the representation of method names, complex data structures, return values, exceptions, and other elements.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsschicht

- **Motivation:**
  - Um eine komplexe Landschaft von verteilten Systemen zu integrieren ist ein Infrastruktur notwendig welche die Daten zwischen den Systemen austauscht
  - Neben dem eigentlichen Austausch von Daten ist oftmals eine zusätzliche Schicht notwendig welche zusätzlich Aufgaben über den verschiedenen Daten realisiert

"Any problem in computer science can be solved with another layer of indirection."—David Wheeler (chief programmer for the EDSAC project in the early 1950s

https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff647962(v=pandp.10)

Integrationsschicht

Maintaining State

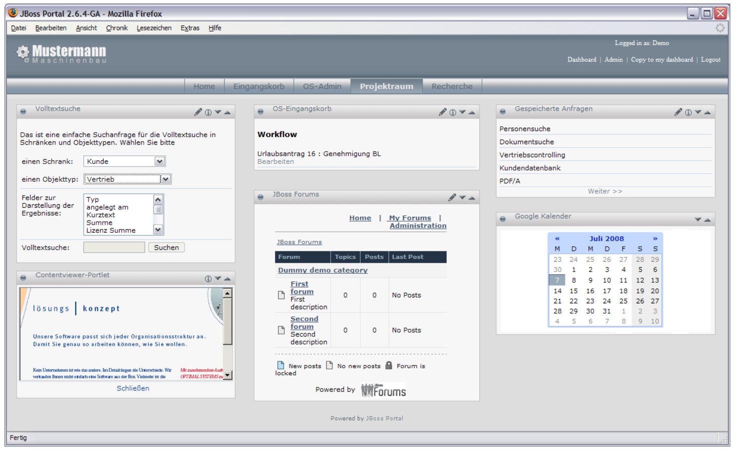
- Most integration solutions aim to streamline business processes such as placing an order or making a payment. These processes rely on the coordinated completion of a series of steps. When an order arrives, the order has to be validated, the inventory has to be checked, the sales tax has to be computed, and an invoice has to be generated. This type of integration requires some mechanism to track the current *state* of an order.

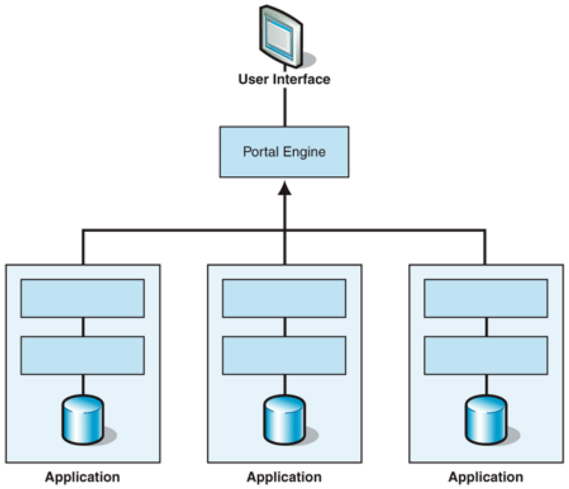
- **Manually**. State can reside in the user's head or in a manual on the user's desk
- **Inside existing applications**. The state can be kept inside existing applications
- **In an integration layer**. You can insert a new integration layer to orchestrate the activities across multiple applications and to keep track of state.

https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff647962(v=pandp.10)

Integrationsschicht - Portalintegration

- **Motivation: Portalintegration**
  - Integration von Anwendungen und Systemen auf Ebene des User-Interfaces
  - Einfachste Form der Integration
  - Geschäftsprozess existiert nicht explizit sondern ist dem Benutzer bekannt
  - Bereitstellung eines zentralen UI‘s welches die Informationen verschiedenster Systeme aggregiert und bereitstellt.

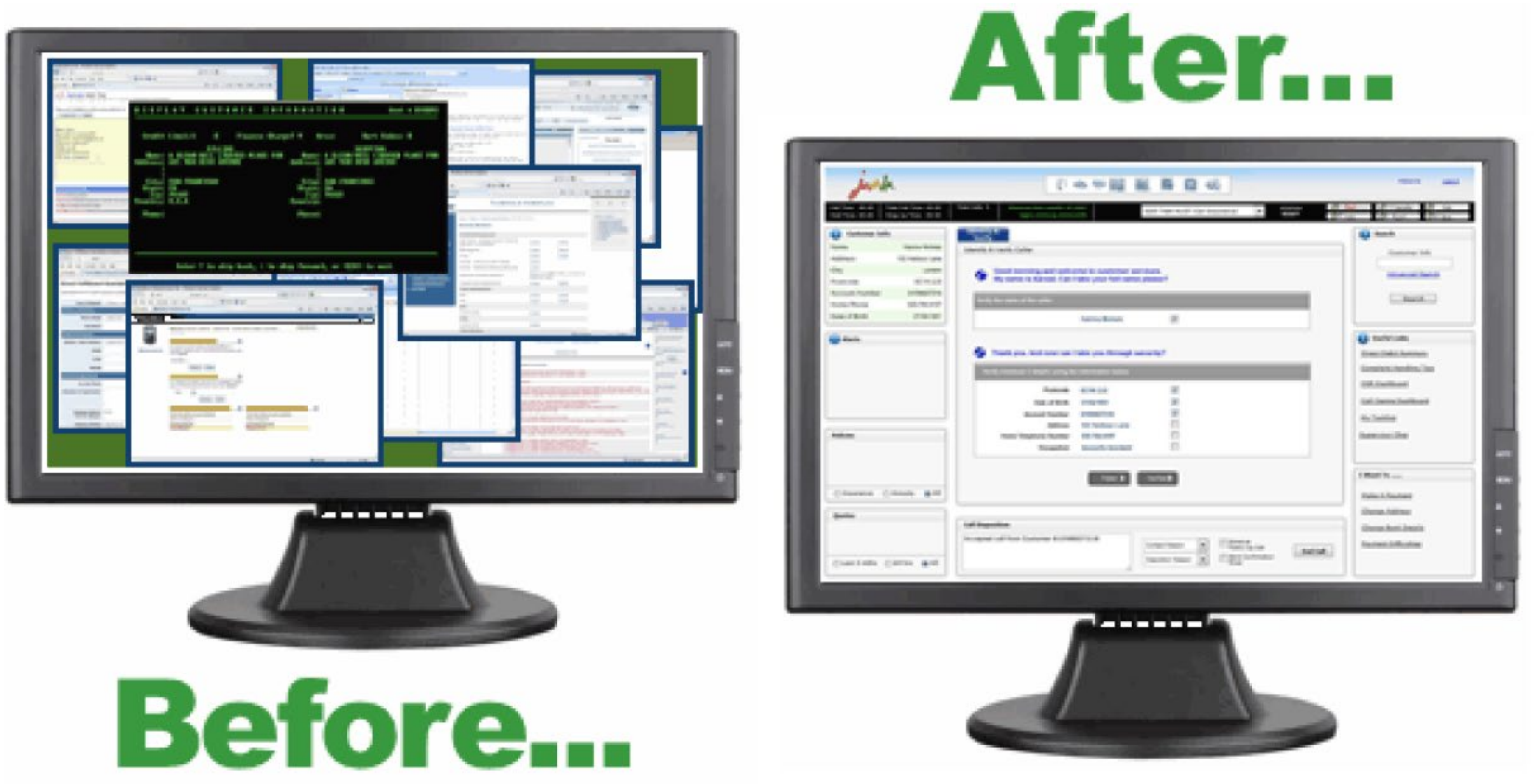




Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsschicht

- **Portalintegration**



Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsschicht

- **Portalintegration**
  - **Display-only**

    Einfachste Form der Protalintegration. In unterschiedlichen Bereichen des Bildschirms wird verschiedene Information angezeigt
  - **Simple post-processing**

    Definition von zusätzlichen Regeln/Logik durch das Portalsystem welche auf die Daten der einzelnen Applikationen reagieren und Folgeaktionen anstoßen
  - **Single application interaction**

    Anzeige von unterschiedlichen Daten und Informationen. Die Interaktion des Benutzers erfolgt immer genau mit einer Applikation.
  - **Cross-pane interactivity:**

    Interaktion zwischen verschiedenen Elementen eines Portals zum Austausch gemeinsam genutzter Funktionalität

Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsschicht - Portalintegration

Vorteile

- **Non-intrusiveness**. Because its primary function is the retrieval and display of information, Portal Integration can typically be added to existing systems without disturbing the existing functionality. This is particularly true if Portal Integration uses Data Integration for the retrieval of data from the individual systems.
- **Speed of implementation**. Adding a portal to existing applications can often be accomplished in a matter of days or weeks, whereas a full integration of the systems could take many months. Many vendors offer Portal Integration platforms in combination with a suite of prefabricated panes that integrate with many popular enterprise applications and data sources.
- **Flexibility**. Because the user makes the decisions, Portal Integration can be used in situations where business rules are not well understood or not agreed upon.

Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsschicht - Portalintegration

Problemfelder

- **Inefficient**. Portal integration is best suited to the automation of simple tasks and processes because it still requires manual interaction by the user. For example, because most portals allow the user to interact with only one application at a time, the user might have to perform multiple manual actions in sequence to complete a complex business function. This makes Portal Integration unsuitable for high-volume applications. In these situations, Process Integration is generally a better solution because the sequence of steps can be encoded in a process model and run automatically by the process manager.

- **Error-prone.** When using Portal Integration, the user makes business decisions and determines the sequence of tasks to be performed. While this provides flexibility, it also introduces the risk of human error. Therefore, Portal Integration is not well suited to integrations that require strict repetition of a defined business process. Process Integration is generally a better choice in those scenarios.

Quelle: http://msdn.microsoft.com/en-us/library/ff647030.aspx, 10/2014

Integrationsschicht – Entity Integration

- **Motivation:**
  - Daten auf Unternehmensebene sind in uneinheitlicher Weise über mehrere Repositories verteilt. Bestehende Anwendungen benötigen eine einzige konsistente Darstellung von Schlüsselentitäten, bei denen es sich um logische Gruppen verwandter Datenelemente wie Kunde, Produkt, Auftrag oder Konto handelt. Das Replizieren von Daten zwischen diesen Repositories ist möglicherweise keine praktikable Option.

- **Problem**
  - Wie können Unternehmensdaten, die redundant über mehrere Repositories verteilt sind, von Anwendungen effektiv gepflegt werden?

Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

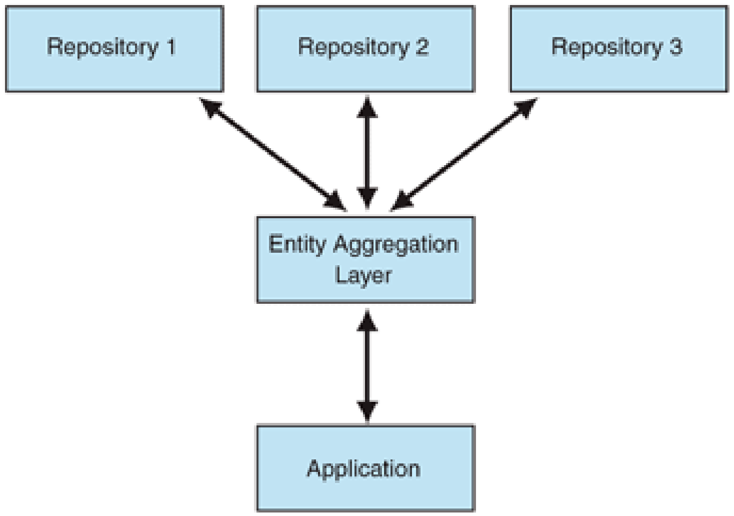
Integrationsschicht – Entity Integration

- **Forces:**
  - Für ein und dieselbe Entität kann es mehrere Datenbanken geben. Geschäftsregeln und -prozesse könnten die Art und Weise vorgeben, in der die Datenbank über eine bestimmte Entität bestimmt
  - Oftmals besteht eine semantische Dissonanz zwischen den Datenwerten, die in den verschiedenen Repositories für dieselbe Entität dargestellt werden.
  - Selbst wenn die Datenelemente semantisch konsistent sind, können die Informationen, die sie darstellen, bei parallelen Instanzen des Datenelements variieren.
  - Anwendungen benötigen unter Umständen logische Teilmengen der Datenelemente, die nicht in einem einzigen Repository verfügbar sind.
  - Möglicherweise gibt es bereits Datensynchronisationsprozesse zwischen Repositories, die es ermöglichen, dass ein Repository als Front-End für das andere fungiert.

Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

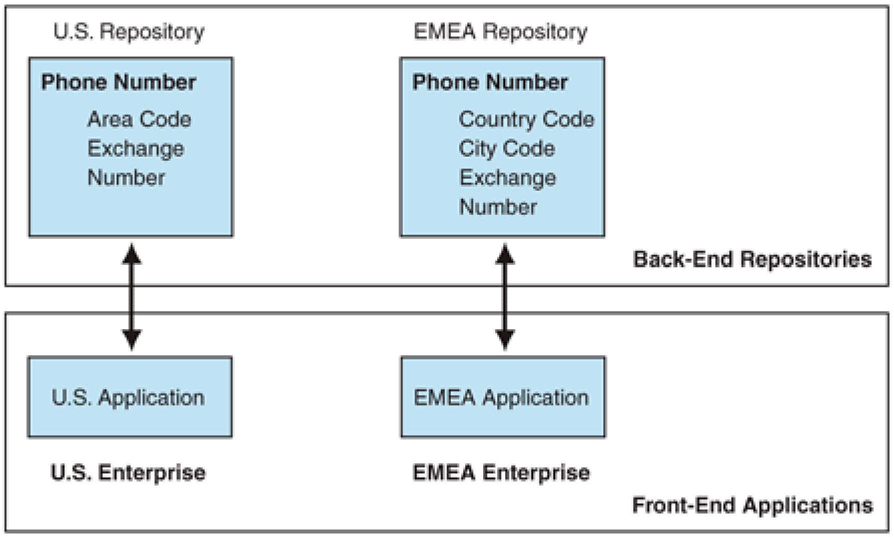
Integrationsschicht – Entity Integration

- **Solution:**
  - Einführung einer Entitätsaggregationsschicht, die eine logische Darstellung der Entitäten auf Unternehmensebene mit physischen Verbindungen bietet, die den Zugriff auf ihre jeweiligen Instanzen in Back-End-Repositories unterstützen und diese aktualisieren.



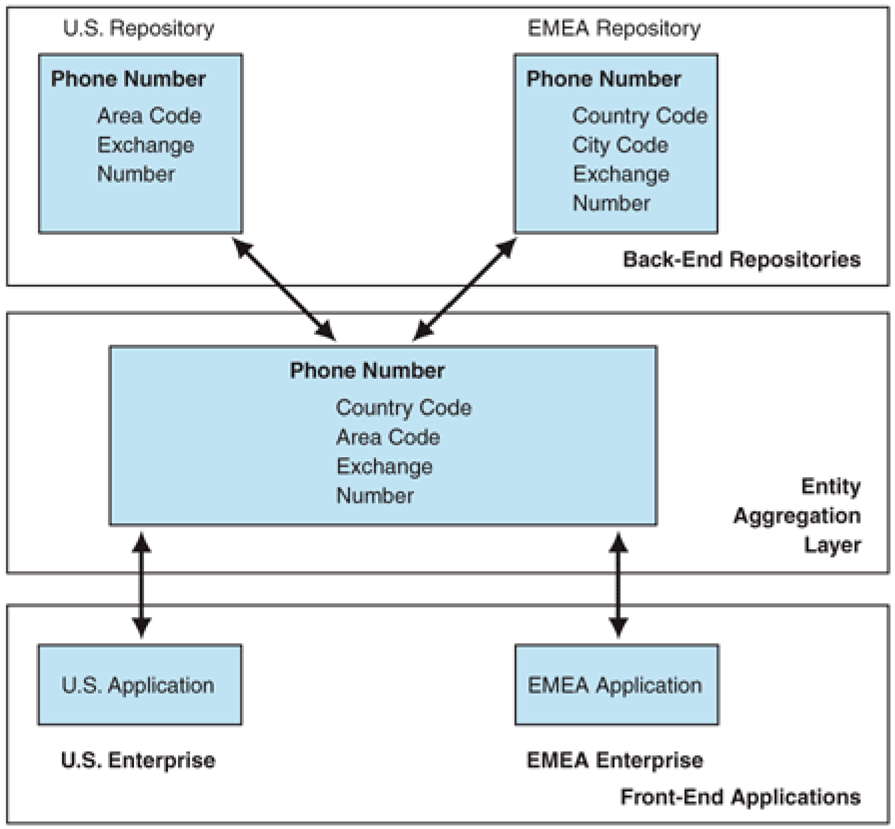
Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

Integrationsschicht – Entity Integration



Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

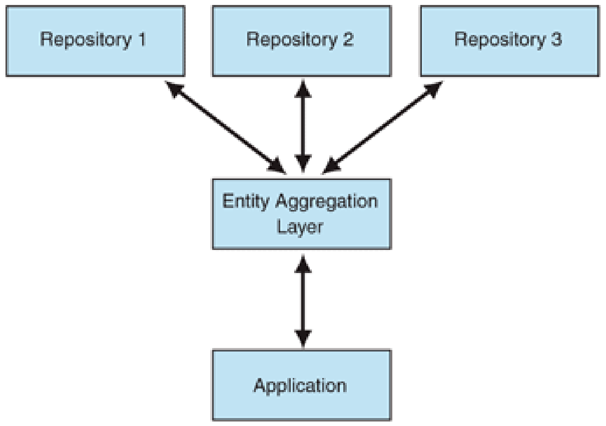
Integrationsschicht – Entity Integration



Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

Integrationsschicht – Entity Integration

- **Solution: Straight-Through Processing**
  - Bei einem Straight-Through-Processing-Ansatz werden die Informationen in Echtzeit aus den jeweiligen Back-End-Repositories abgerufen und zu einer einzigen, einheitlichen Ansicht zusammengeführt. Dies setzt voraus, dass die Entitätsaggregationsschicht in Echtzeit mit den Repositories verbunden ist und in der Lage sein sollte, die verschiedenen Instanzen der Entität zu verknüpfen.

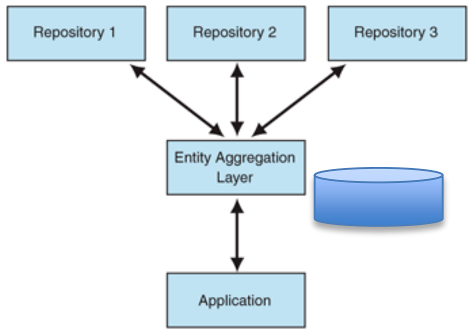


Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

Integrationsschicht – Entity Integration

Solution: Replication

- Die Replikation von Entitäten zur Verwendung durch die Entitätsaggregationsschicht ist erforderlich, wenn die folgenden Bedingungen erfüllt sind:
  - Die Echtzeit-Konnektivität zu den Repositories ist nicht vorhanden.
  - Komplizierte Verknüpfungen zwischen mehreren Instanzen der gleichen Entität in verschiedenen Repositories sind erforderlich, um eine konsistente Darstellung zu gewährleisten.
  - Eine hohe Leistung ist für die Lösung unerlässlich.



Quelle: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649505(v=pandp.10), 10/2023

Integrationsschicht - Prozessintegration

- **Motivation: Prozessintegration**
  - Verschiedene Systeme nehmen an einem übergreifenden Geschäftsprozess teil
  - Geschäftsprozesse müssen permanent an neue Gegebenheiten angepasst werden
  - Problem: Wie können lang laufende Prozesse über mehre Applikationen hinweg gesteuert werden?

Quelle: http://msdn.microsoft.com/en-us/library/ff647433.aspx, 10/2014

Integrationsschicht - Prozessintegration

- **Forces**
    - To implement the overall business function, you could have one system directly call the system that performs the next step, as defined in the business function. However, this approach encodes the sequence of interactions into the individual systems. Creating this dependency in each system makes changing the sequence more error-prone and also limits your ability to reuse systems in multiple contexts.
    - The change and maintenance cycle of a complex business function is likely to be different from the change cycle of the individual business functions that reside inside the applications. For example, financial functions such as computing sales tax or posting revenues are typically subject to infrequent, but mandatory, legal or regulatory changes. In contrast, the overall execution of a business function might be changed much more frequently based on business and marketing strategies.
    - Complex business functions can often take days or weeks to complete. However, most functions that are available in existing applications are synchronous; that is, the caller has to wait while the application performs the requested function. This type of synchronous interaction is not well-suited to long-running business functions because one application could spend a significant amount of time waiting for another application to complete the requested function.

Quelle: http://msdn.microsoft.com/en-us/library/ff647433.aspx, 10/2014

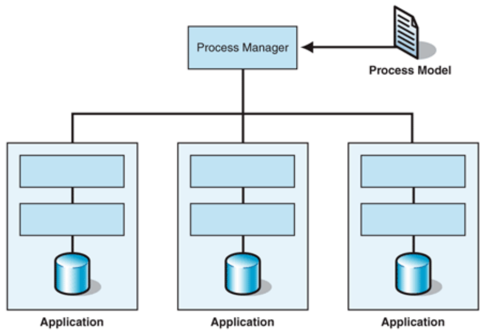
Integrationsschicht - Prozessintegration

- **Forces**
  - A longer time span of execution increases the likelihood that an application might fail during the execution of the business function. If one application fails, portions of the overall function may have to be reverted so that all systems can be returned to a consistent state.
  - Because a complex business function can take a long time to execute, it is likely that a new request will arrive while the previous one is still being serviced. To improve response times, the solution should be able to handle multiple concurrent executions of the function. Many applications are not designed for this type of concurrent execution.
  - Modeling business processes inside a software component can be difficult if the processes are not well understood or documented. In many businesses, users make decisions based on experience rather than documented business rules.

Quelle: http://msdn.microsoft.com/en-us/library/ff647433.aspx, 10/2014

Integrationsschicht - Prozessintegration

- **Lösung:**
  - Definiere einen ausführbaren Geschäftsprozess welcher den komplexen Ablauf und Schritte zwischen den verschiedenen Akteuren (Systeme/User) beschreibt



Komponente	Aufgabe
Process Modell	Definiert den logischen Ablauf einer Geschäftsprozesses
Process manager	Interagiert mit den verschiedene Applikationen und steuert für jede Prozessinstanz die einzelnen Verarbeitungsschritte
Application	Führt eine spezifische Businessfunktion gesteuert durch den Prozessmanager aus.

Quelle: http://msdn.microsoft.com/en-us/library/ff647433.aspx, 10/2014

Integrationsschicht - Prozessintegration

Vorteile

Maintainability. Creating a separate process integration layer allows users to define and maintain the business process independent from the implementation of the individual functions. This increases maintainability and reduces the skill set requirements for the personnel who maintain the process definition.

Reusability. Because the existing applications are not dependent on the process management layer, the functions inside these applications can be reused in multiple process definitions.

Flexibility. The process manager can support a variety of configurations that would be difficult to implement in many traditional programming models. For example, parallel execution of multiple tasks, synchronization points, timeouts, and escalations are all configurations that would be difficult to implement in a traditional programming model. Supporting a variety of configurations gives the process manager the flexibility to adapt to many different business requirements.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsschicht - Prozessintegration

Vorteile

Reporting capability. Because the process instances are maintained centrally, it becomes feasible to extract statistical information that spans multiple process steps and process instances. Such reports can provide answers to questions such as "How long does it take on average to fulfill an order?" or "How many orders are on hold due to insufficient inventory?" This type of reporting can be instrumental in making informed business decisions.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014

Integrationsschicht - Prozessintegration

Problembereiche

Potential bottleneck. Managing a large number of process instances in a central process manager component can present a run-time bottleneck. However, in most cases, the parallel execution of multiple process manager instances can alleviate this threat while maintaining the benefit of central configurability.

Temptation to overuse. This liability is the flip side of the flexibility inherent in Process Integration solutions. Because Process Integration can be used to solve nearly any integration problem, some people take this as an indication that each and every integration problem should be solved using Process Integration. This is not true. Using Process Integration frivolously can lead to overarchitected and sometimes inefficient solutions—for example, in cases where a Message Broker would be perfectly appropriate to address the requirements.

Complexity. A generic process manager can be difficult to build because it has to deal with concurrency, checkpoints, and recoverability. For these reasons, a commercial process manager should be used whenever possible. If a commercial component is not available, the cost of building the process manager should be carefully weighed against the cost of making changes to the process model.

Quelle: http://msdn.microsoft.com/en-us/library/ff649730.aspx, 10/2014