5. Aug. 2019 · 4 Min. Lesezeit

The SAP-Standard does not provide processes

There is much talk of "processes" in the SAP environment. There are business processes, "business process monitoring", software development, software maintenance, transport, migration processes and many more. Hardly anyone is worried about what this term means for software and its use. One of the most prominent business processes in SAP ERP is the goods receipt process. This consists of the following steps:

- An order is initiated which creates a receipt in SAP
- The order will be sent to the supplier
- The goods are delivered, collected, and a goods receipt document is created in SAP ERP

This example shows all the essential characteristics of a business process:

- The process has a temporal course
- The process has a direction, that means it is possible to measure progress
- The process has a defined state at all times
- The process has different editors. These can be serial or parallel
- The process carries a log with itPast process states may need to be recoverable
- SAP documents can be included or created (purchase order, goods receipt, possibly others)There are physical documents (forms) included (for example delivery note, order)

There are numerous similar business processes that are generally handled through SAP ERP. Examples include:

- Receipt of invoice
- Receipt of complaints
- Sales Process (Customer Inquiry -> Offer -> Order)
- Inventory

The SAP standard does not provide processes

As business processes, however, these processes are only available on an organizational level in SAP ERP. The SAP standard only delivers the documents between which the processes run. On receipt of goods for example the order and material documents are created during the process. The handling of the process is solely up to the user.
Appointments, progress, condition, involved processors, documents must be managed by the user. There is no central location where the user can gain an overview of the running processes. Monitoring the processes is time-consuming, cumbersome and unreliable, since many data sources (documents) must be used for the evaluation. In the process handling, there are frequent media disruptions: Communication via email, telephone, short message services. This communication remains undocumented. In retrospect, it is difficult to understand decisions.
The SAP standard apparently - intentionally or unintentionally - leaves a gap of astounding proportions.

The requirements for a process management tool

If you want to create a software tool to support the business process, it is expedient to abstract from specific use cases. That is, for example, to consider the use cases "invoice receipt" and "goods receipt" and to try to isolate the common properties and requirements.

Business object process

It has proven to be useful to consider the process itself as a business object. Process runtime
Processes can be advanced manually. That means they can be controlled by the user in a dialog transaction, but they can also progress automatically.
Common forms of business often involve both forms of "pushing ahead". It follows that a kind of "process runtime" is needed to automatically drive the processes.

Monitoring vs. Cockpit

Process status and progress must be able to be monitored. For this purpose, a monitoring transaction is required, which identifies all essential key figures and provides a view of the detailed data of the process. Allows monitoring transaction as well as the control of the process, in other words pushing the process forward and changing the process data, that's what we call Process Cockpit.
From a software point of view, monitor and cockpit can be the same transaction, which has different rights.
Software architecture of a process management tool
We assume a development in ABAP OO because the ABAP runtime offers decisive advantages.

Monitoring and Cockpit

Monitoring provides the user with all the necessary information about the progress, number and status of the processes. From the basic monitor, the log of each individual process should be visible. In addition, the base monitor should allow the reboot and debug of a single process and thus already fulfills a few cockpit functions. The monitoring is completely decoupled from duration and merchant class. It can be implemented in any UI technique (UI5, Mobile, SAPGUI, Windows).

Application Cockpit

Content-wise the application cockpit follows the monitor, but also provides views on application data as well as application-specific functions. The UI is an extension of the base monitor. The implementation of a cockpit can be accomplished in a very short time.

Handler Class

The basis of all application processes is an abstract process without application reference. This is implemented as a simple ABAP-OO class (handler class) with a few properties:

- The class provides its own persistence, that means it reads or writes itself into the database
- The class provides a locking mechanism so that only one editor is ever allowed
- The class writes its own log
- The class has a dedicated callback method, which is called for "dark" processing of the process runtime
- The class is inheritable

All application-specific classes derive from this abstract process class.

Late Binding

The process run time searches for the merchant class in a registration table associated with the particular process (e.g., merchant class for the goods receipt process). The merchant class is instantiated at run time and its callback method is called by the runtime.
This concept of late binding is based on the principle of the Component Object Model (COM), which was established by Microsoft back in 1992 and is still used today in the new Windows 10 Runtime (WinRT).

The solution: the universal process tool

For companies it can be worthwhile to close the "process gap" this way:

- From a process perspective, a central entry point is created, which often corresponds to the department view
- Process lead times are shortened and measurable
- Errors are completely logged
- Processes become transparent: Reporting can be implemented as a simple extension of monitoring
- The project implementation times are optimized by the high degree of reuse of existing software components
- The software adapts to the business process and not vice versa
- The concept is future-proof, as it adapts to new UI technologies with minimal effort.
- The principle can also be applied to technical processes such as migrations or asynchronous, parallelized bulk processes

Would you like to know more about our solution?