How to Structure Difficult-to-Automate Business Processes

Summary

This article concerns designing automated systems with a high level of user interaction and exceptions. These ideas took shape during a rollout of a UK-based shared service center providing back-office support to more than 15 European subsidiary companies. Using the problems experienced there as a starting point, I describe how this was solved in R/3, and extrapolate the successful elements for similar problem resolution.

Author: Philip Kisloff

Company: AstraZeneca

Created on: November 20, 2006

Author Bio

Philip started with SAP R/2 but for the last 10 years has been a consultant for the SAP Business Workflow tool. Now he is a technical architect and business analyst for an SAP customer, working with other applications as well as the SAP NetWeaver platform.

Table of Contents

Summary 1
Author Bio 1
Introduction 3
The Problem 3
A Different Structural Approach 4
Shared Service Center Requirements 6
The Original Solution 6
Problems with Original Solution 7
Solution Based on ISR 7
The Object-Centric Approach 11
  Objects and Classes 11
  Inheritance 11
  Abstraction 11
  Encapsulation 12
  Polymorphism 12
Discussion 13
Disclaimer and Liability Notice 14

Prosperity is a great teacher; adversity is a greater. (William Hazlitt)

Introduction

What follows are my personal reflections on designing automated systems with a high level of user interaction and exceptions. These ideas took shape during a rollout of a UK-based shared service center (ESSC) providing back-office support to more than 15 European subsidiary companies. The project was a great success, but it was not without its challenges and compromises, in order to realize the business benefits in an acceptable time frame. Not to detract from the project's accomplishments, some of the difficulties that were endured included:

Major subsidiaries had a legacy of different solutions, requiring modeling of diversity with little influence on forcing the adoption of common procedures.

The best business-processes design had to take second place to meeting project deadlines. The bigger picture took precedence over the details, and the workability of some of the solutions suffered as a result.

A hard working and stretched ESSC staff had to deal with complicated procedures due to the lack of coordination in design and many manual procedures to be followed.

Halfway through the rollout, I advocated the abandonment of the first attempt to support a critical business process and a re-implementation with a different solution. The novel process-modeling methodology proposed here was created from the resulting successful implementation.

The Problem

With business process design, exceptions test the validity of an approach like no other circumstances. Repeatable, predictable processes are easy to model, but rarely is real life like that. The easiest cases are when one procedure is followed by a discrete set of following procedures; so it is then possible for conventional analysis to describe and realize the desired business process. Unfortunately, more complicated real world processes are harder to model.

Consider some abstraction of the problems involved in the diagram below. There are many processes in a business that are both human-centric and unstructured (lower-left quadrant). These difficult-to-automate areas challenge the business in three ways: No auditability, no real-time decision support, and no efficiency metrics. Faced with these challenges, there are two responses: press for compliance to policies and procedures (lower-right quadrant) or adopt a system to manage the process (upper-left quadrant).

Increasing system reliance ^

^ Adopt a system Impose a design to fit the system	Automated human-centric workflows?
No auditability No decision support No efficiency metrics ***Human-centric***	Policies and procedures Press for compliance >

> Increasing structure

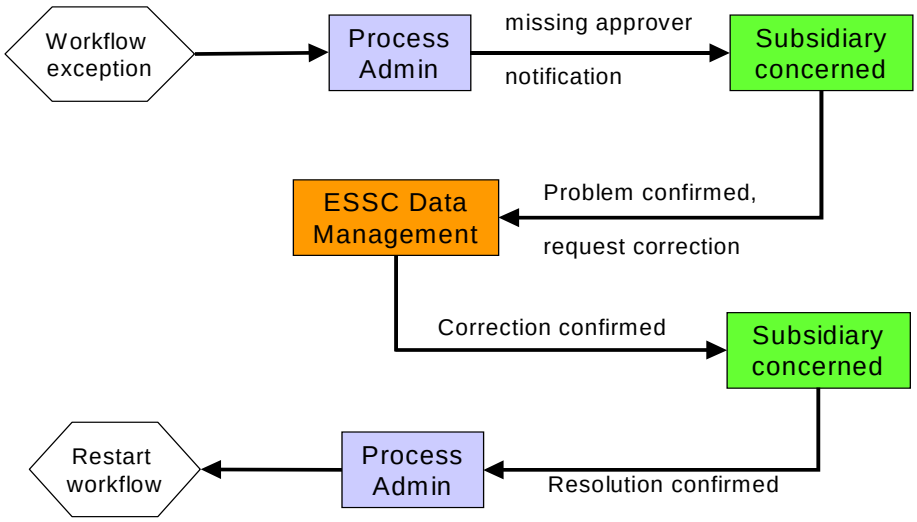
Pressing for compliance still leaves us without the benefits of an automated system solution. However, a disadvantage with the traditional approach to process automation is its inflexibility. If we adopt a (workflow) system, and there are a lot of exceptions requiring workflow flexibility, then these have to be catered for explicitly or dealt with outside the workflow. Similarly, when a system or application is called on to provide the necessary functionality to implement the business requirement, specific applications often require the business process to bend to fit the software.

Now let us make a conceptual leap. How can structure be achieved from unstructured and human-centric activities? There must be a new approach that provides structure without being inflexible and is realized through a system but is responsive to complexity.

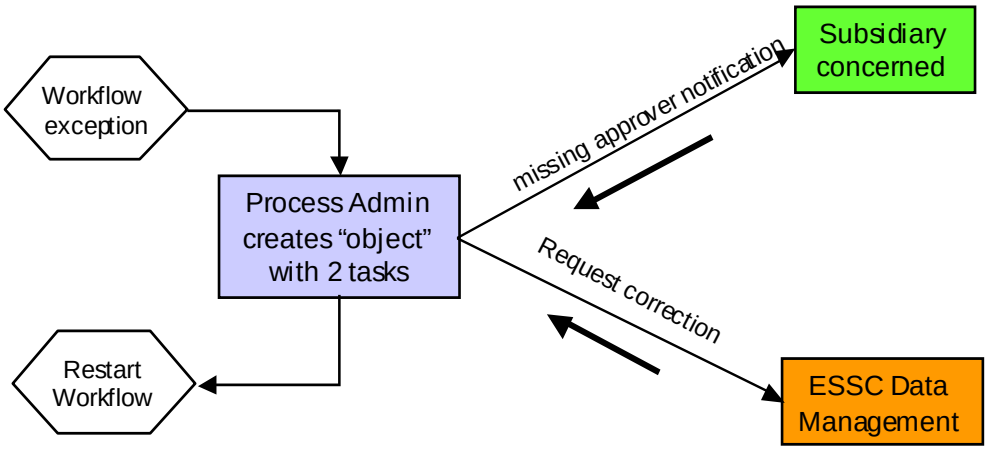
A Different Structural Approach

To illustrate the new approach, let us take a real-life example from ESSC that uses the traditional approach and see how it can be improved. Here, a separate workflow-approval process could not find an approver in the subsidiary's cost producer's department at the required authorization level, and therefore notifies the ESSC process administration staff of a workflow exception. Before that workflow can be restarted, the missing authorization level has to be filled by someone from the subsidiary's cost producer's department. Because there is both division of responsibility within ESSC and the subsidiary knows their people, this must be done through coordination of the subsidiary and a different part of ESSC that is responsible for the actual system changes.

In fact, this problem is a ubiquitous pattern for any coordination between three or more processing agents. Naturally, we are accustomed to viewing the process as a chain of predictable and sequential steps.



This process could be split into several steps, not linearly, but object-centrically. An example of how this looks is shown below:



On the face of it, the object-centric model appears to have done little new. Nevertheless, Process Admin now retains full visibility of each step, as they are the process owners. Also, the subsidiary concerned is relieved of the necessity of liaising with more than one ESSC department. Control is maintained by a dependency of the second step (calling ESSC data management) on authorization from the subsidiary concerned.

To explore more deeply the value of this approach, we need to see it work in a more sophisticated and challenging environment of the ESSC operating model. This is a multi-faceted example, because it takes the concept of a service on three levels: physical implementation, business processes, and corporate operations.

Shared Service Center Requirements

The chief area to support the ESSC ways of working centered on two main requirements:

The handling of queries indicated a need for a dedicated management system. The use of a common system in ESSC and in the subsidiaries would enable the distribution of tasks across ESSC staff. This would allow follow-up on query resolution from suppliers and customers through status tracking, productivity analysis, and performance management. This system would serve as the basis for call logging and recording of not just the type and quantity of complaints but also their resolution, to indicate where future investment should be directed.

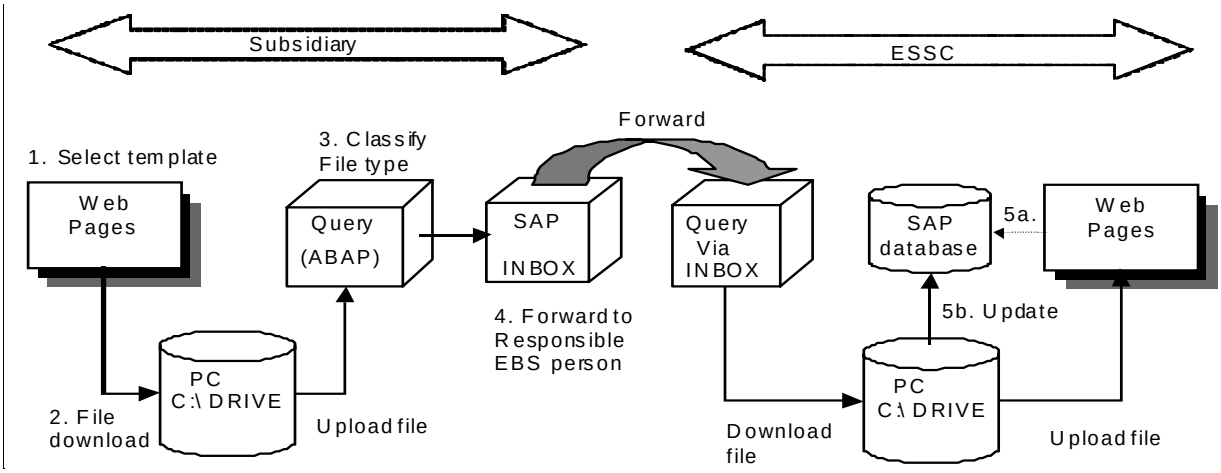
The ESSC business model mandated the prevention of posting journal entries and maintaining master data by the subsidiaries, which was only to be performed by ESSC operations. By front-ending the data entry, it allowed for simplification of inputs for the subsidiary end user (such as. context help and suppression of unwanted fields) and the opportunity for additional group-wide checks before creating or changing data. The model also enabled integration of received spreadsheets in a controlled upload to R/3, by categorization of data content for authorization purposes.

The Original Solution

An ABAP programming language query transaction was developed, which could be used in one of two ways:

Query mode sent by SAP office mail, with classification of the query nature and possible attachment of R/3 objects.

Data upload mode with the process requiring uploading and downloading data files (created from HTML templates) or Excel spreadsheets linked to this transaction. The transaction was then forwarded from the SAP inbox.



For data loading, the main steps involved were:

On the company intranet, the company subsidiary user enters data into the relevant HTML template (including creation of a material master record).

The data is saved or exported into a file located on the PC.

Within an SAP transaction, a record with the relevant subtype (such as Create Material) is created and attached to the file just created. When the record is saved, a workflow work item will be automatically created in the user's inbox.

User navigates to the SAP inbox, selects the relevant work item, and forwards it to a member of the ESSC team.

The ESSC team will review the request as defined in the individual procedures, and trigger the data to be automatically loaded (for example, auto load template).

Query is closed and optionally sent to the inbox of the originator via workflow.

The original subsidiary user can reprocess any failures by importing the data into the company intranet template, correcting and reperforming steps 2 to 4.

For call logging, the message is classified and forwarded without attachments.

Problems with the Original Solution

Despite meeting the stated business-model requirements, the original solution was not considered by the business to help ESSC add value because:

There was no process around when there was an error during upload by ESSC, other than to restart the whole process again. Templates, each with a new query ID, would be passed back and forth between ESSC and the subsidiary to get accurate completion before successful upload.

There was nothing to prevent anyone bypassing the system altogether with e-mails for spreadsheets and call logging. As a consequence, key performance indicators were difficult to measure and achieve.

The one-size-fits-all solution did not recognize and deal with the diversity of the different objects and their follow-on processing requirements.

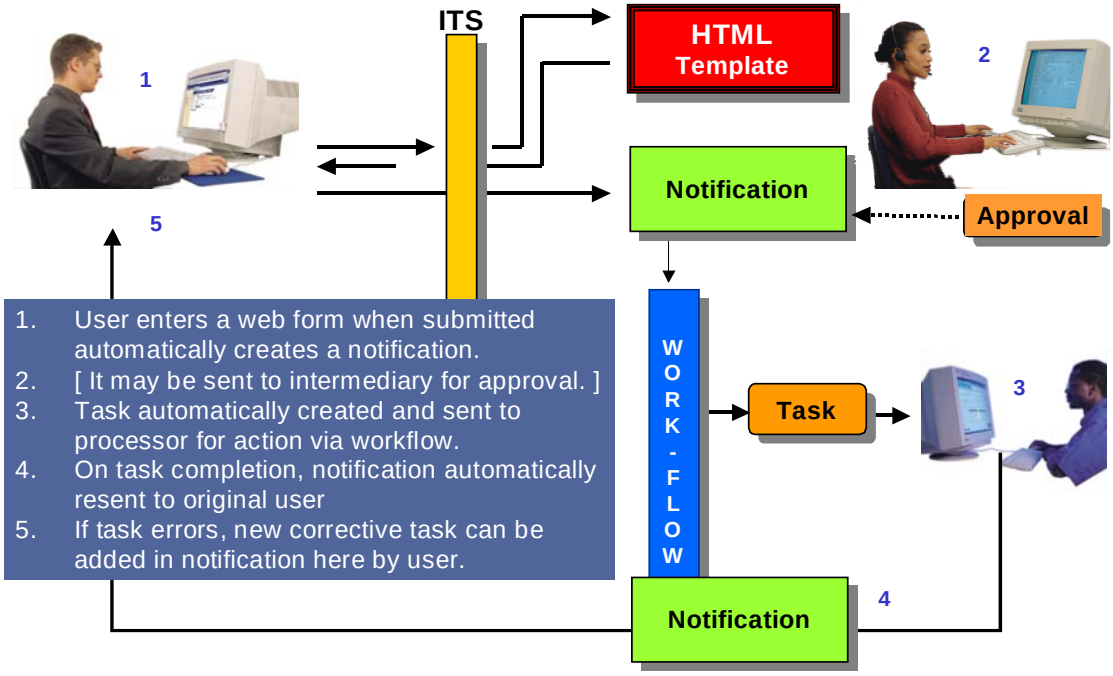
It did not leverage existing SAP functionality outside of the ABAP development environment.

Solution Based on Internal Service Requests (ISR)

Internal Service Requests, available from R/3 release 4.6C, are the basis for manager self-service and employee self-service modules in mySAP ERP. The SAP help documentation can be found here. What follows is a brief outline of the ESSC solution based on ISR. For the sake of brevity, a lot of interesting, but incidental, detail has been left out.

The characteristics of ISR include an HTML-form submission through ITS referencing data dictionary fields and accessing run-time BADI search help and validation. On submission of the HTML form, a CA-NO module generic notification is automatically created (similar to quality notifications). It can be configured that a task belonging to the notification is created and workflow routing can send this to a recipient. Both the notification and their sub-tasks share access to the submitted HTML data stored in the business document service in XML format[1],

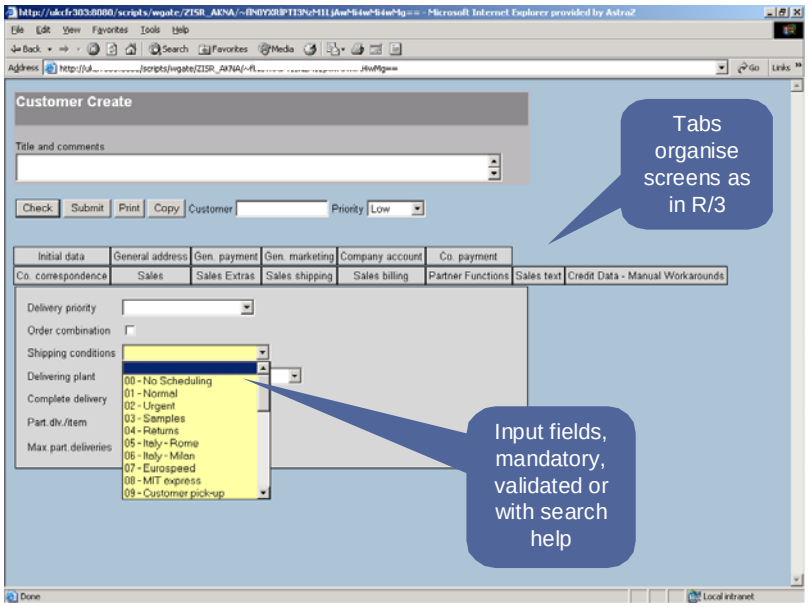
[1] ISR now supports SAP Interactive Forms Software by Adobe. In a future iteration, a more paper-like user interface could just as easily be achieved. Also, the SAP Business Workflow tool, now part of the SAP NetWeaver platform, is used by ISR to do the workflow automation, and the tasks deployed are also part of SAP NetWeaver.



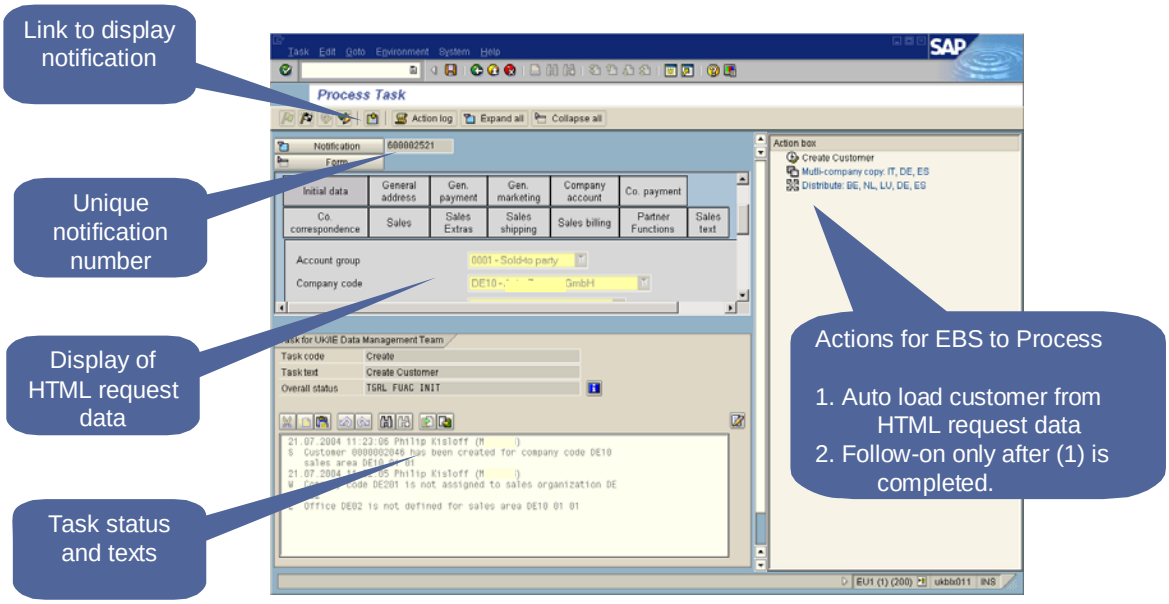
Through a division of security authorizations, subsidiary users could only create and change notifications (either directly in R/3 or incidentally through HTML form submission). This meant only they could change the XML data within and create tasks to send to ESSC. Emulating a client/server model for shared-service operations, they could not process the tasks sent; only ESSC had this authorization.

Now, the CA-NO notification can give a self-contained overview of the process history and subsequent required steps from within the notification itself. All notifications have statuses to track the progress of a process and catalogs to categorize the type of process. Each notification and task has an action box with a list of executable functions saying what should happen and in what order. This allows the process to be self-documenting. Such a to-do list can also aid control of steps outside of the system, whereby steps might not have a start or end communicated automatically by a business event.

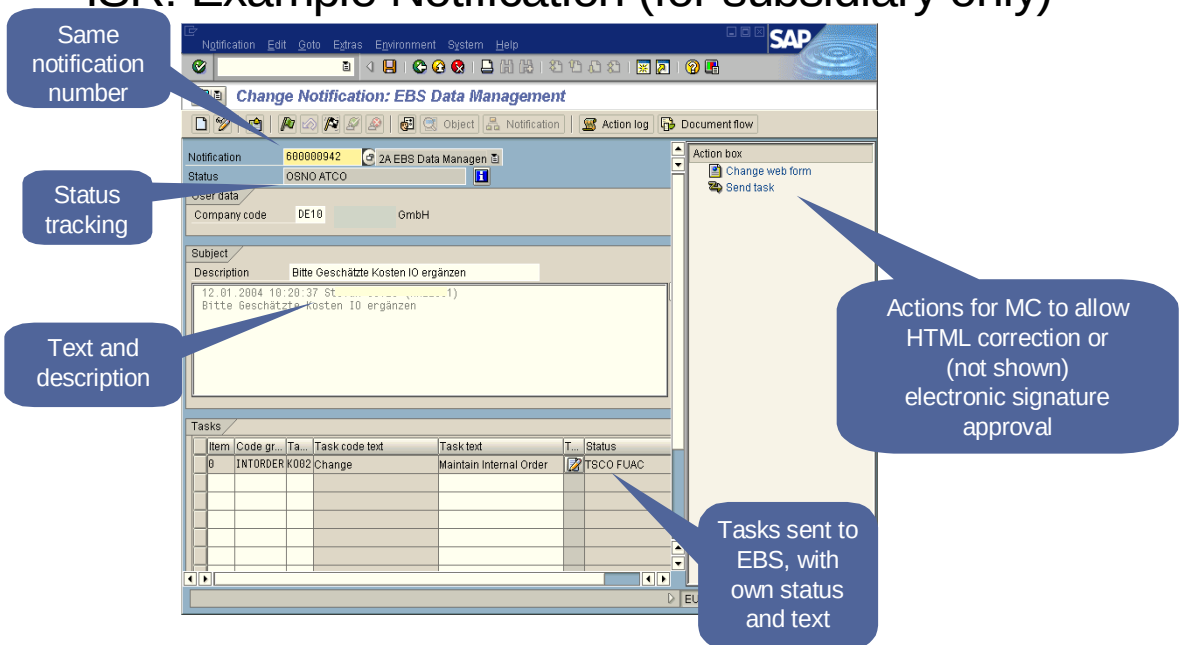
ISR: Example HTML Request Form



ISR: Example task (for ESSC only)



ISR: Example Notification (for subsidiary only)



For the new structural approach, the object-centric part of the process is defined by the automatically created notification, and the first task to be sent to ESSC represents the first step in the process. Actually, steps in the workflow are now meaningless, as everything is now apparent as either sub-tasks in the notification or as background follow-on activities. As notifications and tasks are coordinated using status management, all workflows are modeled as services triggered by state transitions. Two workflows for notifications and tasks, respectively, cope with all different scenarios, with one extra for ER/ES approvals[2].

One of the potential benefits of this approach is that workflows are now services used for state transition events, and exceptions are easier to model without altering any predefined process flow.

The design for the error-correction process must create as little unnecessary overhead as possible. For master data maintenance, the needs of an error-correction process were built in from the start. Although the original system had no validation of input at HTML form entry (because the Web server was independent of ITS and could not read the backend system) full immunity from upload error is impossible to achieve. The business rules in R/3 are so complex that it would be futile to attempt to replicate them all at the front end. While we could catch a higher proportion of errors with ISR validation through the BADI supplied, the ultimate backstop was when ESSC staff autoloaded XML data into R/3, and BDC/BAPI processing reported return messages.

[2] ER/ES (Electronic Records/Electronic Signatures) was provided for approval of some data creation/changes prior to upload. While the original task for uploading data is created on submission of the Web form, it is not sent via workflow to ESSC and it is in a status that prevents use, pending creation and completion of another task. This second task is called within the notification action box and requires a password entry to complete. Once done, the original task is released and can be processed by ESSC. The second task also becomes a permanent record of the approval/rejection for compliance purposes.

Thus, a seamless way dealing of with errors was essential. Such a mechanism was this:

When an error occurs during data upload by ESSC, the task is manually completed (but without additional indication of success) together with an error text, and the notification automatically returned to the subsidiary's user through a workflow triggered by the status change.

The message entered by ESSC in the task is also available to be viewed from within the notification by the subsidiary user. If appropriate, the XML data could be amended through a Web form contained in the notification, and a new task within the notification submitted to ESSC. Repeated errors can be tracked under the same notification object instance.

When the automated actions required by the task are completed (and they can only be performed once), the task is flagged as completed and successful. This also locks the notification as completed.

It occurred to me that an anthropomorphic perception of a computer's operation (imagine a little man inside performing some work in response to key strokes) became a reality to a subsidiary's user when ESSC staff processed their request and transparently attempted autoloading before typing back an explanation of the failure reason. For various reasons, some autoload attempts failed several times before success, but even in these dire cases, the typical end-to-end time to completion was only several minutes.

A slightly different approach was taken for spreadsheet attachments and journal postings. Both required the notification to be created directly within R/3, and the attachment or dummy journal posting (or sample document) attached by object relationship services inside the notification. The task is then created via the action box and sent automatically to ESSC for processing.

The Object-Centric Approach

Bearing in mind ISR and CA-NO notifications only approximate the desired characteristics of an object-centric approach, this new model has brought about the following changes:

Processes have become operations (ISR tasks) belonging to objects (CA-NO notifications).

Data flows have become messages (or status management event changes).

Information can be hidden inside the object (with object relationship services).

If we continue along this perspective, we can speculate how the four pillars of object orientation (which comprise inheritance, abstraction, encapsulation, and polymorphism) have relevance to the object-centric approach for business process analysis and design.

Objects and Classes

Classes are defined, and objects are an instance of a class at runtime.

Subsidiary requests are now all dealt with by notifications, but these notifications can be of different types. These notification types each represent a different business processes. For example, there is one for HTML uploads, one for sending attachments, one for processes that require electronic signatures, and one for logging calls and sending queries. So each notification belongs to a notification type analogous to a class. When created, a notification becomes identified by a unique ID (which is an instantiated object).

Inheritance

We can divide a thing into objects, where base class represents the common functionality and derived class represents its unique functionality.

Inheritance can be used to break up a process into manageable chunks. Derived classes can be thought of as blocks in a larger, all-encompassing scenario. With the ESSC/ISR example, I have only managed to deal with single blocks. The behavior would be something like a notification's ability to have its type manually changed from parent to child as the circumstances required. This might be necessary by unpredictable, but predefined requirements to process something as a special case.

Abstraction

Abstraction in simple terms means implementation hiding.

Follow-up actions, user exits executed on saving, and subsequent workflows triggered in background provide a level of abstraction for additional process functionality.

Encapsulation

Binding data and methods together is encapsulation. Methods and data can be accessed only after creating the object of that particular class. Only objects that are permitted can access the data of the other objects.

The synergy here is that object relationships defined in CA-NO notification document flow are acted upon with action-box functions. Also, the statuses of a notification can be set by functions in the action box[3].

However, at a more general level, CA-NO notifications can act as a wrapper for R/3 data objects. With a binary relation created between the notification and R/3 data object, generic object services allows us to trace back to the notifications that have acted upon the object, where the notification(s) provide a detailed audit trail of all changes performed. This function was particularly useful for ESSC for immediate response to queries about their actions creating and changing master data.

Polymorphism

In object-oriented programming, there are two categories of polymorphism. Ad-hoc polymorphism is where one thing appears in many forms depending on context, and universal polymorphism is where different things appear the same in a certain context (that is, one thing used differently or different things used the same way).

Well, that's Java for you. Suppose we leave the programming environment and imagine what relevance polymorphism has for business processes with human interaction. One way to look at it is the provision of a set of consistent tools supplied for user manipulation – whatever the different scenario or processing step, there is a polymorphism in communication and interaction.

This concept of polymorphism as shown in the use of different notifications types (quality notifications, customer services, and service management, and so on) within SAP R/3. But what would be the advantage of restricting the user interaction with business processes in a rigid format?

Reduced training overhead is one reason. I am reminded in this respect of the Microsoft Windows desktop environment. A familiarity and standardization with the Windows user interface has helped improve productivity. This idea of a homogeneous layout would have benefits in the old workflows of the SAP R/3 4.6C software, where apart from a generic decision task, too many times unique and initially bewildering varieties of screen options would be presented to users switching from one bespoke developed workflow to another.

Further, such commonality could also improve the measurements of efficiency metrics and other areas of business intelligence. If all attributes of a process are catalogued in a polymorphic format, and the progress tracked through a universal system of status changes, then the data gathering at source would become a lot more consistent, thus aiding summation and comparison from a wide variety of scenarios[4].

[3] Objects have both behavior (they do things) and state (which changes when they do things). In OO, a class encapsulates state (attributes) and offers services to manipulate that state (behavior).

[4] The workflow information system transfer structure MCWF_TRANS in user exit WISEXIT has additional fields for CA-NO notifications. After populating with report RMCDATA, business warehouse can be used for analysis instead of WIS.

Discussion

For this part, I wish to return to the original question of how to structure difficult-to-automate business processes. As I said at the beginning, this direction was born out of using ISR to realize a shared-service-center way of working, but it would be a mistake to think of ISR and CA-NO notifications as a solution – they are only artifacts of a deeper understanding.

It is true that business processes should be a reflection on the way the business has chosen to operate, rather than the technology used to implement it. Some processes will be linear, chronological, and cumulative and some non-linear, asynchronous, and collaborative – for good business reasons. One not-so-good reason for the former process is because of people's finite capacity to manage complexity. Arguably all processes have to break down to this structure because of the way human and non-human resources are managed.

To some extent, this is begging the requirement. While starting out to understand and document a business's activities with the aim of improving them, we fall back on the viewpoint of data transformations, which may be performed by either, or a combination of, manual activities and automated systems. The model in vogue is the data-flow diagram, originally developed for the purpose of describing how data is transformed and stored by a system. Early notations were based on the Yourdon method, and served as a basis for business process execution language (BPEL) now used for service-oriented architecture (SOA).

The advantages of data-flow diagrams are that they are user friendly and can be easily understood by business users who have to verify their validity. As a model of the business's process, they depict separate chunks of work and show the information flow between them. The problem arises when the solution to the business requirement is not linear, chronological, and cumulative, or at least not satisfactorily described by such an approach. Such examples are unstructured human-centric activities, which feature parallel processes with collaboration. Managing the administration of starters or leavers within a company usually has no automatic triggers from business events, either to start or end the process. There are many more examples when you start looking.

The data-flow diagramming disadvantages may be overcome when an alternative approach is taken, borrowing the concept of the object from system analysis and design as a focus for modeling the process. Compared with the data-flow approach, the object model provides a more direct analogy with business tasks and activities. *The key insight is that the process in its entirety is an object.* The object provides a way to partition the complexity of a business scenario into processes that are easier to model real world behaviors.

Instead of representing a process as a flow diagram, we can use a collection of rules (or triggers) to represent the process without explicitly delineating the paths. Such a system is more flexible in capturing the dynamic behavior. I foresee three approaches to such a system:

1. Strong use of objects uses rules to model processes. However, executions are difficult to analyze or explain because logic is scattered over many rules.
2. Weak use of objects uses a model, whereby logic of business process is explicitly defined, or scripted, and rules are used for checking constraints, handling exceptions, and so on.
3. A combination approach uses scripting for process of major workflow blocks, and rules are used to dynamically change the flow at execution time within these blocks.

I've had some gratifying success with the use of CA-NO notifications in control issues around approval of financial invoices presented without prior PO approval, and the ideas behind the object-centric methodology have also helped design several other complex processes. I believe there is a great potential to have these ideas adopted as a fully fledged modeling approach to compliment an SOA landscape.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

The opinions expressed in this article do not necessarily represent the opinions, policies or strategies of AstraZeneca.

