

Business Process Automation Approach to Branch Banking Software

The term Business Process Automation is replacing the term Work flow Management System for high end WFMS. While a normal WFMS system concentrates merely on the creation of documents and their routing, a BPA system attempts also to provide a powerful facility for incorporating the process logic and computerising variety of process models and their interactions.

Majority of the Indian banks have a history of several decades. Most of them have several hundreds of branches and carry the legacy of existing distinct and proven business processes. In order to be efficient and responsive, they may want to change their business processes but due to the legacy and skill level of available manpower they may prefer an evolutionary albeit fast approach rather than a revolutionary change. Using the Business Process Automation approach affords computerisation of the banking transactions in a way that conforms to the above requirement.

In the Indian banking system, each branch owns its database and enjoys a complete control on it. They interact with other branches, and other banks for certain transactions only. When the branch of a bank is interacting with other branch of the same bank, it may be subject to a similar business process. However, when a bank interacts with another bank, the transaction may be subject to a considerably different business process. This fact of real life strongly suggests that a distributed database model with native variants of Business Processes and powerful capabilities for inter-workgroup transactions would be an ideal solution to computerising them.

The modeling of the bank as an organisation will require clear definition of the following BP entities: the branch as a **workgroup**, the officials as the **transactors**, their interactions as **flows** and the processes they perform as **transactions**. These entities can be modeled elegantly using object techniques.

The workgroup is a collection of interacting transactors, who own their database and processes. The workgroup can be further characterised by a set of attributes. Examples are scale of the branch, designation of branch-in-charge, scale of the branch-in-charge, code of the branch etc. By transactors we mean roles such as clerk, officer, manager etc. Transactors can be modeled as instances of a class having certain attributes. For example there could be several managers (class) but one of them could be the Principal officer or in charge manager (attribute). Similarly there could be several officers (class) but of different levels (attribute).

In this article the term "*transaction*" or "*process*" will be used to mean banking transactions such as Account open, Withdrawal, Remittance etc.,

The privileges and responsibilities of transactors differ with the transaction. Hence a proper assignment of transactions to transactors provides both a convenient model of the business and the desired intrinsic security. For example an officer authorised to approve SB withdrawal may not participate in the loan sanction process. Similarly an officer authorised to approve withdrawal in routine case can perhaps only recommend it if it is TOD.

The banking transactions can also be categorised as belonging to certain distinct **classes** such as for example, the financial and non-financial or the ordinary and the exceptional. The transactions can be modeled with good transactional modularity and re-use capability. For example payment of cash through voucher, issue of PO/DD, viewing of pending Cr/Dr transactions of an account etc., are well defined re-usable transactions. This situation affords an elegant usage of the sub process or "*forking of a transaction*" concept where a main transaction may initiate another well-defined transaction in order to complete the function. This approach of modeling the transaction as independent processes and inter-linking them through "*fork a transaction*" capability will result in an optimal transactional granularity, easy upgradability and maintainability of the software.

Banking transactions also throw up situations where independently well-defined sub processes may sometimes have to be **dynamically combined** together to form a single transaction. For example, while issue of passbook and issue of chequebook can normally be looked upon as independent transactions, in certain situations such as account open, they may have to be combined together and treated as a single transaction.

When transactions move from one transactor to another and pend with a transactor, the concerned transactor has to be able to select a transaction for processing based on one of the many selection criteria such as value of certain field or time order or transaction name or who has sent the form etc. Such a flexible selection scheme is very important for achieving efficiency in operation. Further, a transactor should be able to process another pending transaction while viewing an already selected pending transaction. For example, having selected a withdrawal transaction of an account, he may find that it is necessary to pass a remittance transaction of the same account before processing the withdrawal.

Within a given bank, most of the transactions can be modeled as "*production processes*". This implies that the data carried in the transaction, the process logic applied, the transactors who interact to consummate a transaction and the flows are well defined and are repeatable for all instances of the transaction. Most of the transactions require a sequential flow with recursive routing capability. However modeling of such facts as "*delegation charts*" or "*passing powers*" are best handled with conditional routing and/or ad-hoc routing. Certain inter-workgroup (inter branch) transactions such as funds transfer or clearing can be modeled elegantly as parallel flows.

While there are many types of banking transactions, not all of them flow through multiple transactors i.e., some of the transactions are done by a single transactor, such as issue of a withdrawal slip or viewing the pass book. Therefore the ability to support both “*static*” transactions and “*flow enabled*” transactions concurrently can add considerably to the convenience of operations.

While participating in the consummation of a process, transactors perform various **actions** such as approve, reject, pay cash, view signature etc., An ability to model actions directly and to classify them as mandatory or optional and as manual or auto and assignment of actions to transactors will provide both efficiency and compactness in computerising such transactions. It also affords changing actions from manual to auto under different circumstances including introduction of mechanised peripheral devices.

The financial transactions in banking have a way by which their completion or termination condition can be judged by applying the basic rules of accounting. Incorporating this intrinsically into the transaction’s model will go a long way in guaranteeing the integrity of such transactions.

The BPA approach can also be seamlessly extended to inter-workgroup transactions. Anybranch banking, remote banking and outstation cheque clearing among branches of the same bank can all be modeled on this common framework. This incidentally also enables retaining the distributed database model which naturally fits the existing banking organisational structures and yet perform the inter-workgroup transactions elegantly. The availability of a robust “*transaction level*” protocol for such inter-workgroup transactions can be of great benefit in ensuring the integrity of such transactions.

Customer friendly facilities such as Telebanking can also be modeled under the general frame work of BPA, where the customer originates the process and inputs data remotely and the output data is a special data type viz. voice.

A facility to alter some of the parameters of the BP entities at site under suitable authority is also definitely needed to handle the dynamics of a branch.

Another important characteristic of banking is that transaction requirement may change due to the changes in the banking regulations. For example, consider the sudden introduction of TDS, which did not exist earlier, or the new stipulation to record whether the account opener is a bank employee. Further for practical reasons **all** the transactions in the bank or a given branch may not have been pre-coded either because the specs were not given or the transactions are performed rarely, such as a foreign exchange related transaction which is normally not done in that branch. In general the data to be captured may undergo a change, the process logic may undergo a sudden change or a new process may arise. Hence it would be extremely useful to provide a scheme whereby data addition/alteration, data processing as well as “*adding*” a process to the existing software at the branch by branch staff in a simple manner can be accomplished on an **ad hoc basis** when such a situation crops up without depending on the developer/vendor of the software. This empowerment would be a critical requirement to keep a computerised branch fully operational on the computer and not to revert to a part manual process.

Since auditability and traceability of whatever happens from origination to termination of a transaction is of critical importance in banking, a built-in support to log

the complete transaction history is an invaluable facility. Such an information can be used also for generating a variety of MIS reports which may not have been foreseen at the time of specifying the software requirements. It also enables meaningful analysis of processes and performances and makes it possible to reengineer the processes with understanding.

Since normally the end users of the software may have limited or no prior experience in using computers a simple and uniform interface to the entire software and GUI based forms as the carriers of transactions would make learning very easy and enable fast implementation in the branches.

As size of the branches varies from small to big, the solution should be cost effective across this range. This would imply choice in the selection of the deployment platform, particularly the RDBMS.

Over all, a Business Process Automation framework which can support all the above aspects efficiently will contribute not only to fast and efficient software development but will also ensure upgradability, manageability, maintainability, user friendliness, functionality, throughput and cost effectiveness.

Datanet has developed such a framework in its business process automation library called BPRO™ and has used it for its entire range of banking solutions and in the branch automation software.

