

An Object-Oriented Approach to Banking Information Systems

Carlos Ferrán-Urdaneta

School of Management, Boston University
Boston, MA 02215, USA

Melanie L. Lenard

School of Management Boston University
Boston, MA 02215, USA

ABSTRACT

Financial institutions like banks make extensive use of information systems and information technology, but they are risk-averse and slow to adopt new technology. In this paper, we explore how the new paradigm of object-oriented analysis and design can be used to advantage in banking information systems. We identify the similarities among the different financial instruments offered by banks and then exploit these similarities in a proposal for an object-oriented approach to banking information systems. We discuss the need for a migration strategy to minimize the risk of adopting the new technology.

Keywords: Banking, Financial Instruments, Information Systems, Object-Oriented Analysis, Object-Oriented Design.

INTRODUCTION

The business of financial institutions such as banks and brokerage houses is a business of trust and information. To succeed, financial institutions must manage both assets properly. To manage their information assets, financial institutions make extensive use of information systems and information technology, and they have done so for a long time. However, because they are also a business of trust, they tend to be risk-averse in managing their information. Thus, they tend to continue using old systems and technologies and methods longer than other businesses. Here, we will focus on banks, although much of what we say is applicable to other financial institutions.

In this paper, we explore how the new paradigm of object-oriented analysis and design can be used to advantage in banking information systems. We will identify the similarities among the different financial instruments offered by banks. Then we exploit these similarities to propose an object-oriented approach to building integrated information systems for banks. In what follows, we begin by describing the current status of the banking information systems. Then we propose three primitive constructs that

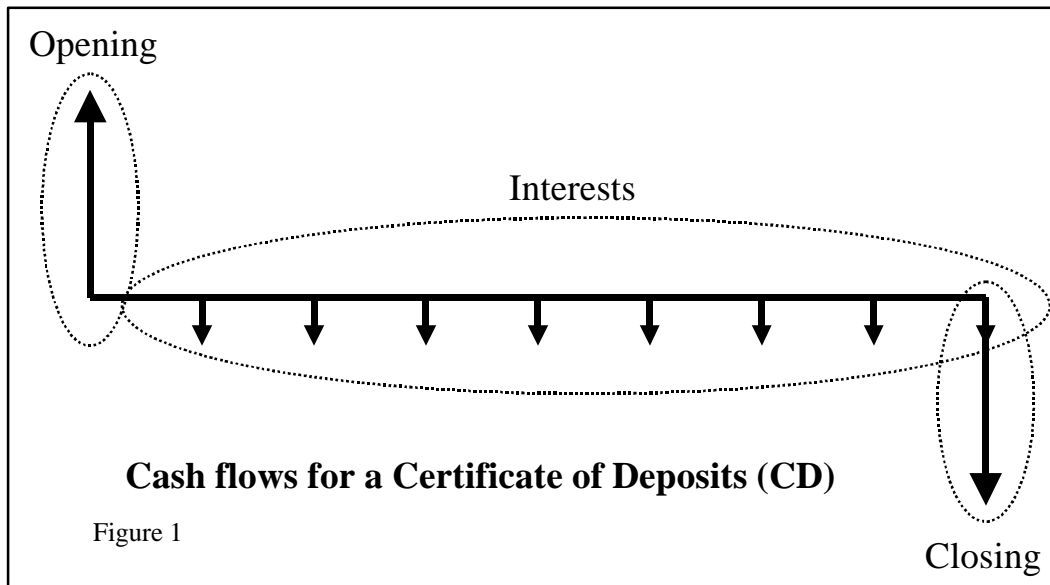
can be used in various combinations to describe a wide range of financial instruments. Next, we show how these primitives may be used to build an object-oriented model of banking information systems. In the last section, we discuss how the risk-averse nature of banking will affect the probable migration strategy for implementing this new approach.

CURRENT STATUS

Currently, most banks manage information with multiple legacy systems that run on one of their mainframe computers. Typically, each of those systems handles one and only one particular financial instrument. Then there are additional systems that try to consolidate the information created by the former. Whenever a new financial instrument is created or a variant of an old one is introduced, banks have to develop a new information system. Very little is reused, new code is written, and the data is structured separately. In general, there is very little integration between the different systems even though most financial instruments are very similar.

Some current projects are trying to address these issues. For example, Wells Fargo Bank designed an object-oriented system that provides an integrated interface to their legacy systems. Each instrument is still managed with a separate systems (and in some cases with a separate mainframe computer), but to the user they are presented as a single system. This allows the Bank to analyze the relationship with each customer based on all their accounts [1]. Furthermore, the use of object-oriented technology allows the bank's IT to have a faster response time to new systems and new instruments. It also allows for some cut reduction on future systems.

Another project that is trying to address these issues is the one taking place at Credit Agricole-Lazard Financial Products Ltd. They have already invested US\$10 million and plan to double that amount in the development of an object-oriented system that would be capable of



incorporating new investment instruments on the fly by adding new objects when required. [2].

ANALYSIS OF FINANCIAL INSTRUMENTS

We begin our analysis by noting that financial instruments may be described on the basis of the cash flows that they generate. There are those financial instruments whose entire cash flow is pre-determined when it is first created (e.g., a certificate of deposit or a bond), while there are others whose cash flow varies over time in a way which is not pre-determined (e.g., a savings account and a stock). This last group can be further divided in those for which part of their cash flow still follows certain rules (e.g., the interest payments in a savings account is calculated based on a pre-determined formula), and those whose cash flow is not at all pre-determined (e.g., dividends in a stock).

Furthermore, instruments are generally based on rules of cash flow calculation based on some principal amount. The present value of an instrument is based on that principal, its future cash flow, and its rules for redemption or cancellation.

A financial institution must be able to value each instance of any or all of the instruments, recall all the past cash flows, and plan for the future cash flows. All the required reports are based on different subsets of the total past (or future) cash flows and the different subsets of valuations.

Therefore, based on some earlier work by the first author [3], we propose the following three primitive constructs as the basis for banking information systems:

- Cash Flow
- Series
- Instruments

Cash Flow: A cash flow is a specific amount of money that moves (flows) from one account to another. (Here we are using the accounting definition of an account. Thus, an account could be the auxiliary in the bank's general ledger for a specific customer savings account, or the auxiliary for interest payments on certificate of deposits.) Following accounting principles, all cash flow must have an origin and a destination. All cash flows are in a determined currency and for a determined amount.

Figure 1 shows the cash flows as arrows. The size of the arrow represents the amount of the cash flow and the direction represents which way it goes (from the customer into the bank or out of the bank to the customer).

All cash flows have at least three dates: generation date, due date and actual date. The generation date is the date on which accrued interest is calculated. Due date is the date in which the cash flow is due to happen. And, actual date is the date in which it actually occurs. For example, in an instrument where the interests payments are due on the 15th of each month, there may be two cash flows to represent it: one that covers the interest from the 15th to the end of the month and another that covers the interest payments from the beginning of the month to the 15th. In the former, the generation date would be the last of the month, the due and actual date the 15th. In the latter, all three dates would be the 15th.

In general, whenever the bank is responsible for originating the cash flow, the due date and the actual date are the same. But when it is the customer who originates it, they tend to be different because the customer may pay in advance or may be delayed. (In the latter case, the bank normally imposes a penalty that generates an additional cash flow).

Series: A series is a collection of cash flows following a similar formula occurring one or more times at regular or

irregular time intervals. All of the cash flows of a financial instrument follow certain rules, and in many cases, several cash flows follow the same rule. When this is the case, we call the set of cash flows a “series”.

For example, monthly interest payments are regular payments made at the end of the billing cycle with a pre-determined formula. Therefore the dates and the amounts can all be calculated with a common formula. These interest payments would constitute a series. Another series

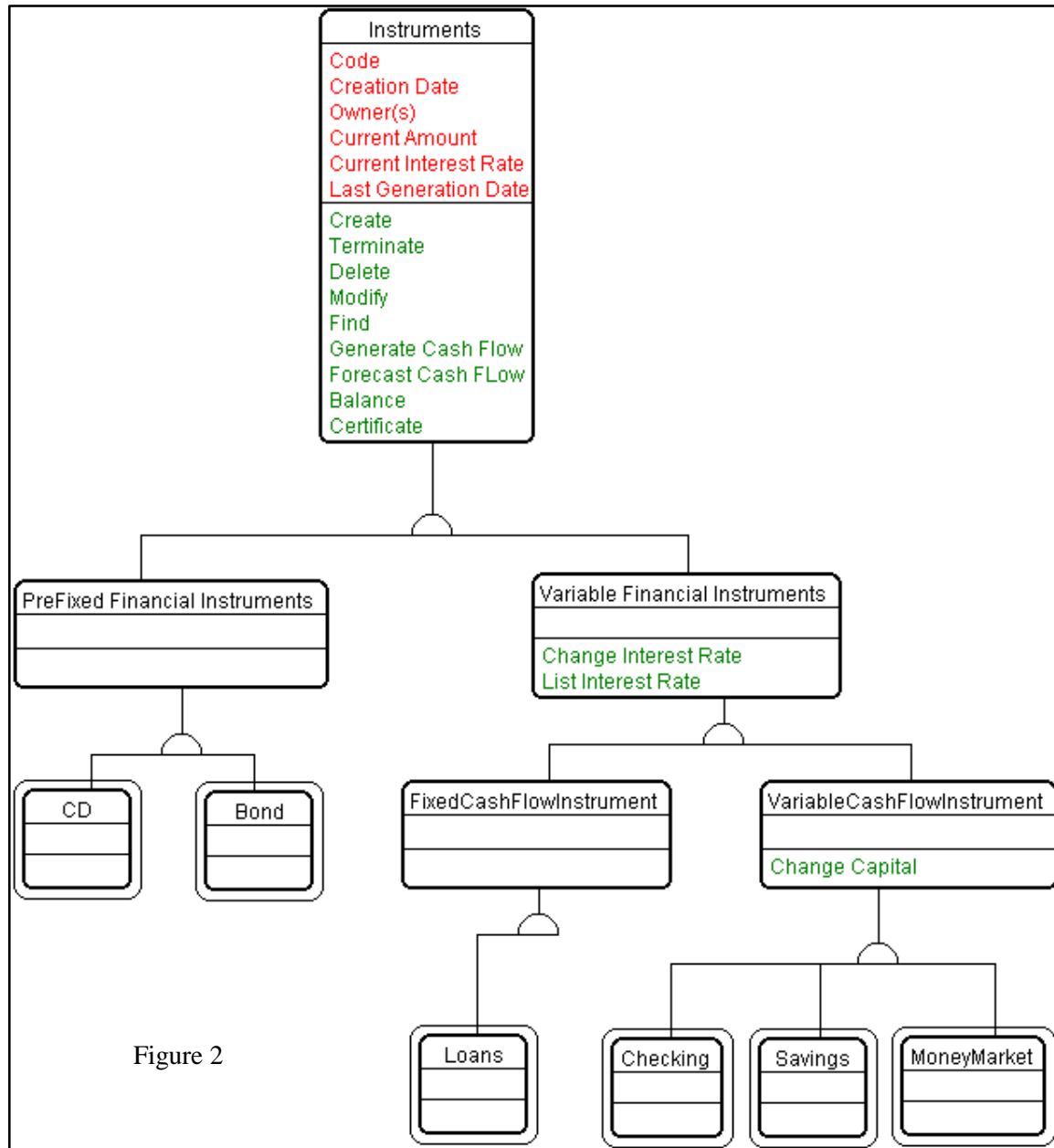


Figure 2

would be the commissions charged at the end of the month for accounts that have balances under the minimum required. Some series comprise only one cash flow, as is the case for the cash flow that is used to open or close an account. Figure 1 shows three series; one for the opening cash flow, another for the monthly interest payments, and a third for the repayment.

Instruments: An instrument is an agreement between the bank and one or several customers which states the rights and obligations of each party concerning specific cash flows that may have to occur between the parties. Such cash flows represent principal, interest, or a combination of both. In general, one party receives funds from the other and it is then obliged to pay them back at a specific point in time as well as pay an agreed interest rate on the balance owed.

The main differences among instruments are based on which party owns the principal, the interest rate, the way the interest is calculated and paid, the way the principal is given and paid back, and the guarantees given. Some examples of instruments are a savings account, a checking account, a certificate of deposit, a money market account. An instrument comprises several series that in turn comprise one or more cash flows. For example, consider a certificate of deposit (CD). As shown in Figure 1, it is composed of 3 series: (1) an initial deposit, (2) regular payments of interest (which may be paid out or added to the principal), and (3) a final withdrawal.

As a second example, consider a savings account. This instrument has four series. Two of them are the same as for the certificate of deposit (the initial deposit series, and the closing withdrawal series). The series for interest payments is similar but not the same as for the CD (the monthly interest payments are based on the average balance on the account and the interest rate may be different every time). In addition, a savings account has a fourth series occurring at irregular times representing the customer deposits and withdrawals. Furthermore, there could be some additional series for commissions that may apply for balances under the minimum.

It should be noted that we can not claim that these constructs will adequately represent all financial instruments. We believe that it does, but we have not proved this formally.

OBJECT-ORIENTED ANALYSIS OF FINANCIAL INSTRUMENTS

Having identified the three primitive constructs underlying all financial instruments, we will exploit those constructs to build an object-oriented model of banking

information system. In doing so, we will use the Coad/Yourdon notation [4].

First, in Figure 2, we show the Generalization-Specialization (Gen-Spec) hierarchy of the object Class we call Financial Instrument. Attributes and methods of the object Classes at the higher levels of the hierarchy are inherited by those at lower levels and specialized if necessary. The basic object "Instruments" includes the attributes and methods that are common to all instruments. The attributes of an instrument are the specific data required to operate it, such as account number, customer code, opening date, duration, interest rate, opening amount, and balance. Specific instruments, like the checking account, may have additional attributes such as the minimum amount required to waive commissions, and the amount of the commission if it applies.

Most instruments will have methods with the same name, but the methods will execute differently for each instrument. Thus, most methods are defined at the top level, but then, they are redefined (polymorphism) at a lower level. For example, the "create" method will request different data from the user depending on the type of instrument being created. A large part of the instrument's "operations" are really embedded inside the methods of the series that are part of the instrument. Those "operations" are the formulae that determine the amount and periodicity of the cash flows.

Basic methods of an instrument are: create, modify, terminate, delete, find, generate cash flow, forecast cash flow, balance and print certificate. "Create", "modify", "find" and "terminate" are self-evident. "Delete" an instrument is meant to be used to correct an error (as long as accounting principles permit it). "Generate cash flow" is the method used to call the series objects and request the generation of any cash flows that may be due since the last generation. "Forecast cash flow" is similar except that the cash flows are not entered in the cash flow database and the instrument status is not modified. "Balance" returns the current value of an instrument and "print certificate" allows for printing of a certificate or title for the instrument.

The main difference from one instrument to another is whether its behavior is completely established at the time of its creation or if its behavior varies over time. So for example, bonds and certificate of deposits are instruments whose behavior is completely defined at the time of their creation. (There may be later changes, as in the case that a company goes into chapter 11 and is forced to refinance all of its debt, but such changes on the instruments are not expected, and can be handled by exception.) Checking and saving accounts are examples of instruments whose

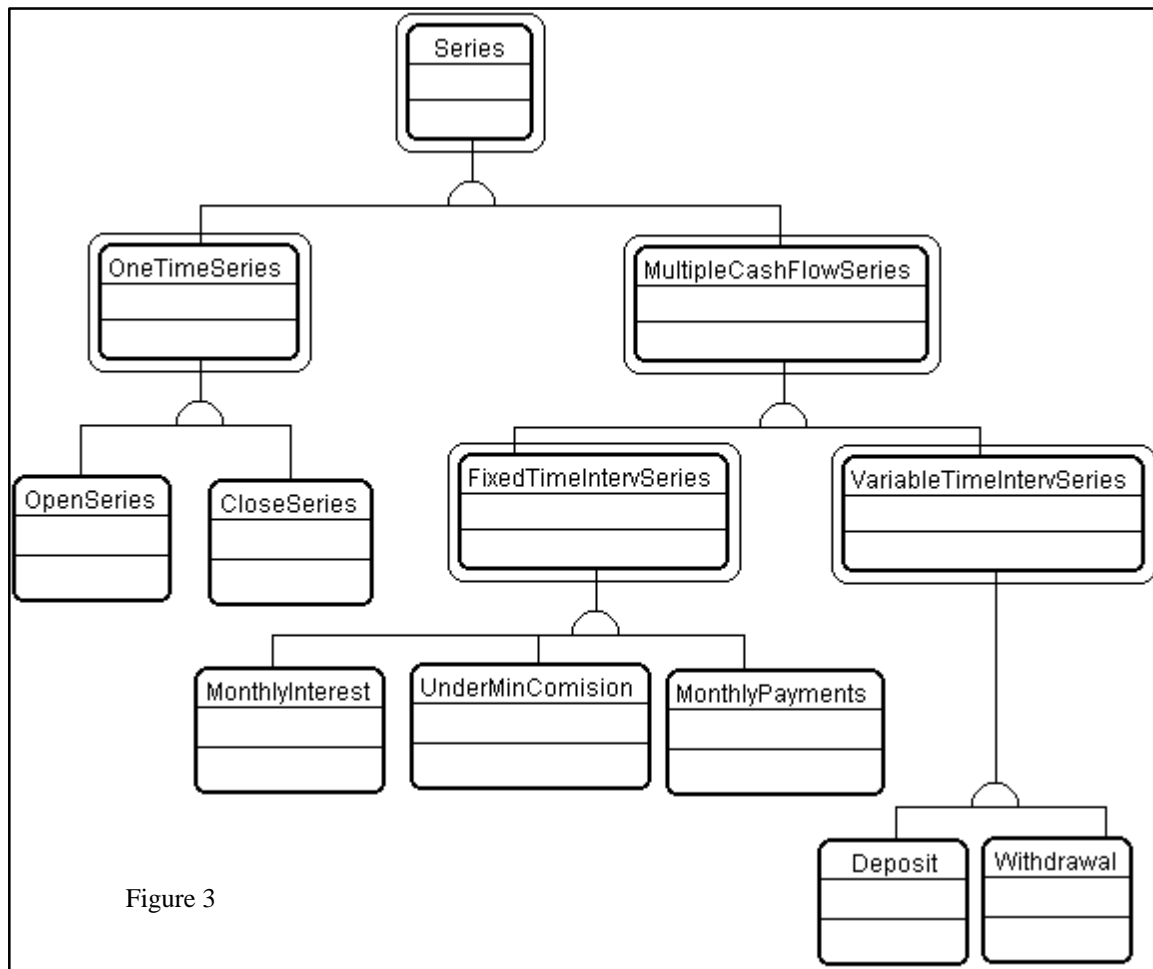


Figure 3

behavior is not totally pre-determined. The interest rate may change from one month to the next, and the balance moves up and down following a function that is defined by the needs and desires of the customer. These latter instruments need additional methods like one to change the current interest rate and another to list the historic interest rates.

Within the instruments with variable behaviors we can also make subdivisions. The first subdivision that comes to mind is of those whose recurrence of cash flow may be determined at the time of opening, but not so the amounts of each cash flow. For some loans, the payment dates are determined at the initiation of the loan but the amount is dependent on the current interest rate. In the other group, we have the saving and checking accounts. These have a changing interest rate and a fixed date for paying interest, but the deposits and withdrawals do not follow any rule. So these instruments require an additional method that

creates a deposit or withdrawal series whenever a customer requests one.

Second, in Figure 3, we show another Gen-Spec hierarchy, this one for the object Class called Series. It represents one of the primitive constructs (i.e. series) described above and its various specializations. Within the methods of the series are the formulae required to calculate the magnitude (amount) and date of the cash flows. All the series objects will need to have one formula (or procedure) to calculate the amount of the cash flows and another to calculate the due date, but the formula will most probably be different for each object. For example, the monthly interest series for a CD may have as the amount formula, " = Principal*Days of the month*Irate / 360" ; while the amount formula for the opening series will be just " = Principal". Furthermore, there may be two types of CDs; one which pays interest based on a year of 360 days and another based on a year of 365 days. The only difference between this two separate objects will be

on the interest payment series, since one will have the number 360 while the other the number 365, but all the other series will be common to both instruments.

Our model of a banking information system will include two additional object Classes, namely Owner and Cash Flow. The object Class Owner could also be a Gen-Spec hierarchy with subclasses such as Individual, Corporate, Trust, but we have not shown this since it is not central to our main thesis. The object Class Cash Flow, of course, represents the other of our primitive constructs.

Figure 4 shows the relationships among the four primary object Classes. Owner objects are connected to Financial Instrument objects by a many-to-many instance association (each Owner may own several Financial Instruments and each Financial Instrument may be owned by several entities). The diagram in Figure 4 also shows that Financial Instrument is at the top of a Whole-Part hierarchy where Financial Instrument is a collection of possibly many Series and a Series is a collection of possibly many Cash Flows.

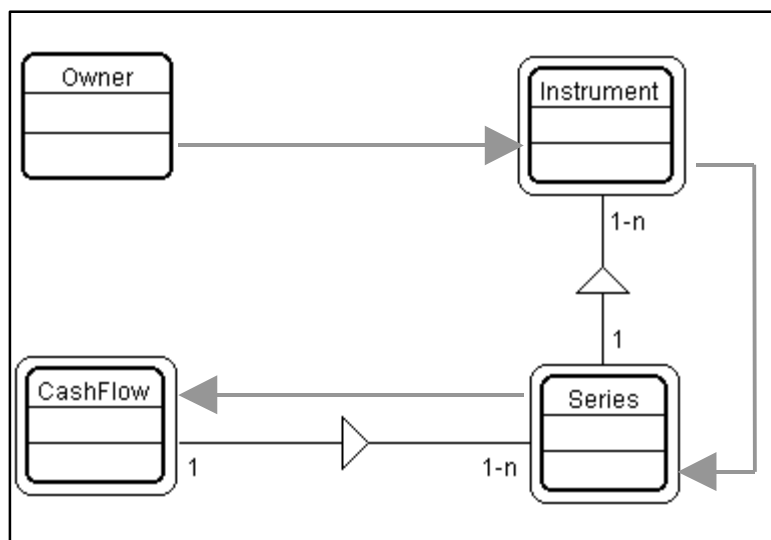
Furthermore, the diagram shows service connections between these classes. For example, at the end of the month, the system will need to generate all the interest payments for the savings accounts (financial instrument). In this case, the “generate interest” method will be invoked on the financial instrument. In its turn, the instrument will call the “advance time” method on the pertinent series (end of the month interest payment series). And this series object will in turn send a message to the cash flow object requesting the “creation” of specific cash flows.

Another example of these services is on the “Flows” method of the instruments. The “Flows” method belongs to the instrument and it returns a set of cash flow object instances that belong to the specific instrument instance. The instrument sends a message to the series, and the series send messages to the cash flow objects. The resulting set are the cash flows that apply to a specific instrument instance in a given period of time. This method will be extensively used every month in producing the statements of account.

Using this approach would greatly simplify the process of providing information systems support for financial instruments. For example, a new instrument could be added to the system with a minimum of programming by defining a new “Financial Instrument” Object (which would inherit many of its attributes and methods) and then specifying its connections to “Series” and “Cash Flow” objects. Furthermore, many instruments could reuse the same series, if applicable.

MIGRATION STRATEGY

As we have shown, object-oriented analysis is a powerful method for modeling banking information systems. However, implementing systems based on this model is an inherently risky undertaking. It may require platforms that are not yet commercially reliable (for example, an Object-Oriented Database) and it will require re-training of systems development personnel. This brings us back to the point that we made in the beginning: banks are in the business of trust and must avoid all non-financial risks.



Banking information systems should maintain and possibly increase the existing level of trust. An unreliable system, say one that goes down for even a day, will damage the public trust in the institution (and may cause interest to be lost on idle funds). Banks are expected to take lending risk; some are also expected to take additional financial risks like interest and exchange rates fluctuations. However, banks are expected not to take other risks. (For example, a bank can lend money to third parties but it should not embark on its own enterprise.) In fact, most banking regulations prohibit banks from taking other risks.

As a consequence, banks must be very cautious about adopting a new information technology. One way to reduce the risk is to adopt a migration strategy of “divide and conquer” [5]. In the long run, the bank will have to replace all of its systems anyway. The bank could begin to migrate to object-oriented technology by using the new approach when it next replaces one of its existing systems. As trust and organizational expertise grow, the bank would move on to replacing other systems. Eventually, the bank could adopt a “unite and conquer” strategy, wherein all systems are brought together under one object-oriented framework. A legacy system may be included by placing a “wrapper” around it to make it look like an object (the wrapper accepts messages from other objects and returns results to the calling object).

CONCLUSIONS AND RECOMMENDATIONS

Despite the risk, adopting an object-oriented approach to information systems may provide a competitive advantage to the bank, so it is an alternative that must be considered. In addition to the migration strategy just described, a variety of other procedures [6, 7] are available to minimize the risk involved in developing object-oriented information systems.

Another option is for several banking institutions to pool their resources in an object-oriented project. This will reduce the risk and also limit the strategic advantage of the project. In this case, the group will benefit from the variance of the institutions. The more distinct the institutions the more comprehensive the system will be. Similar institutions only serve the purpose of dividing the cost, but dissimilar institution not only divide the cost but enrich the system since it has to be designed with a broader point of view. We recommend that banking institutions start an object-oriented project for a complete overhaul of their systems. Those that do not start to gain expertise in this area will most probably pay a very high price on the long run. The new systems should exploit the commonalities among the different financial instruments

and build from them. The analysts will be required to be experts on the new tools, but more important, they should understand the business as a whole and not just try to automate small and separate segments.

These projects are long term project. They must be sponsored by the most senior hierarchy (preferably the CEO) and it should be understood from the start that the systems will not go “live” for several years. The project requires an initial phase which allows for the IT personnel to experiment with and gain expertise on the new tools.

It is essential that object-oriented projects be undertaken slowly and cautiously. Projects that are rolled out too soon most probably will fail, thus creating even more problems for later initiatives, since management will be more demanding and more distrusting. Failure of live systems may cause large damages to the institution as a whole, not only in terms of cost but also in lost trust from the client base.

REFERENCES

- [1] Townsend, E.S., *Wells Fargo's "Object Express"*. Distributed Object Computing, 1997. 1(1): p. 18-27.
- [2] Hoffman, T., *Object-Oriented Financial Package Tames Transactions*. Computerworld, 1995. 29(19): p. 75.
- [3] Ferrán-Urdaneta, C., *Series Multidimensionales: Diseño de una Metodología Generalizada para el Control Automatizado de Instrumentos Financieros en Funcion de una Institucion Financiera*, in *School of Management*. 1990, Universidad Metropolitana: Caracas.
- [4] Coad, P. and E. Yourdon, *Object-Oriented Analysis*. 1991, Englewood Cliffs, N.J.: Prentice-Hall.
- [5] Taylor, D.A., *Object-Oriented Information Systems: Planning and Implementation*. 1992, New York: John Wiley & Sons.
- [6] Fayad, M.E., W.-t. Tsai, and M.L. Fulghum, *Transition to Object-Oriented Software Development*. Communications of the ACM, 1996. 39(2): p. 108-121.
- [7] Sadr, B. and P.J. Dousette, *An OO Project Management Strategy*. IEEE Computer, 1996. 29(9): p. 33-38.