Special Issue on Financial Systems: UDC 681.172.2-52 : 681.322

Software for AT-400N Automated Teller Machine

Tetsuji NITTA*, Kouji UEDA**, Osamu NAKAZAWA***

Abstract

We developed the AT-400N automated teller machine by reconstructing embedded ATM software, which was formerly developed with a dedicated OS / dedicated language, with a general purpose OS / general purpose language. This machine has WOSA/XFS, which can insure the portability of Oki software to ATM platforms made by other manufacturers. To improve productivity and maintainability, an object oriented development method was used for the middle software layer, where ATM core processing is executed.

1. Introduction

The conventional ATM (Automated Teller Machine) has been developed using a dedicated OS and dedicated programming language. As the performance of the ATM mechanism improves, the processing time taken per transaction is a critical factor to differentiate the product of a manufacturer from others, therefore a further increase in processing speed is demanded. A dedicated OS has been the contributing factor to maximizing the performance of the machine, however the conventional ATM is facing the following issues.

- Along with the expansion of banking services, the addition and change of specifications frequently occur. The life cycle of an ATM is considerably influenced by its customizing flexibility and extendibility of functions.
- Except for some standardized specifications, such as for the National Bank Association systems, most ATMs conform to the original specification of the respective bank. The critical issue is how to minimize the functional differences experienced by users.
- New demands are emerging that parallel the use of multi-vendor based information equipment, the internationalization and diversification of services, the further automatization of work at a teller window, and with such technological innovations as electronic commerce and multimedia. The information equipment used in financial institutions, including an ATM, must satisfy these demands. In practical terms, however, it is difficult to meet these needs under the current dedicated OS environment.[1]

As a result, it is inevitable that ATMs that have these features will move into an open systems environment. This time Oki developed software for the AT-400N Automated Teller Machine, aiming at ATMs suitable for the open systems era. We adopted Microsoft Windows NT [*1] as the general purpose OS of this machine. For the design, we placed importance on processing performance, and on the customizing flexibility and reuse of developed resources. We chose a development method based on object oriented design, which especially excels in terms of the reuse of resources and in customizing flexibility.

This paper describes the background that led us to choose object oriented design[2] , the models to which object oriented method was applied, and explains support of WOSA/XFS (Windows Open Services Architecture / eXtensions for Financial Services)[3], which established an extension specification for ATMs in Japan.

2. Basic Development Concept[4]

Figure 1 shows the software configuration of AT-400N. To construct an ATM using a general purpose OS, we established the following basic concepts.

1. Minimizing amount of application description:
   Control of the ATM includes reading cards, I/O control for a cash deposit and withdrawal, control of the

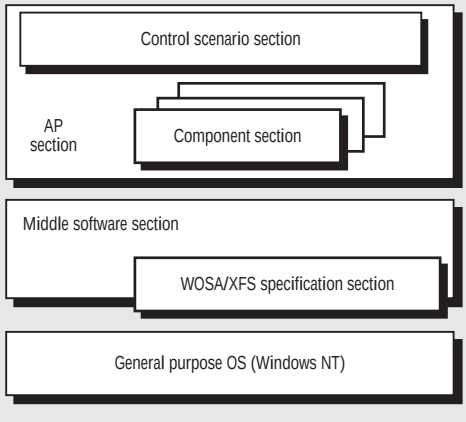

Figure 1: Software configuration of ATM

* General Manager, Platform Development Department, Terminal System Division, System Business Group
** Manager, Platform Development Department, Terminal System Division, System Business Group
*** Senior Research Manager, Media Network Laboratory, Research and Development Group

*1. Microsoft, Windows NT are registered trademarks of Microsoft Corporation in the USA and other countries.

customer screen, and host transaction control. With the application (AP), startup control for these functions and macro control for result judgment processing can be described. The operation for input and output data is encapsulated in a lower layer, so that the amount of application description can be minimized.

2. Insuring customizing flexibility throughout life cycle:
   To insure the universality of the interface between middle software and the AP, software is developed based on object oriented design.
3. Reuse of developed resources and selection of descriptive language:
   The AP consists of a group of components and a control scenario section to control each component. These are further divided into standard components, commonly used by users, and individually customized components, so that resources can be reused. Each component can have a hierarchical structure and can provide a simplified interface to the control scenario section. Each component and object can select a language suited for its purpose. For example, a GUI (Graphical User Interface) object consists of an individual screen and a section to control the sequence of the operation, such as a money transfer, and is described in Visual Basic*2, which is superb in describing GUIs.
4. EUC (End User Computing):
   The merits of using a general purpose OS for users are a decrease in the development period, a decrease in development costs, and the implementation of EUC. For the AT-400N, a GUI section and AP control scenario section were designed considering user access.
5. Insuring real-time capability:
   The processing time required per transaction will always be a factor to differentiate the product. The use of a general purpose OS and object oriented design generally tends to deteriorate the real-time capability of the product. In this development, we eliminated the capability deterioration factors by prototyping and by evaluating the results.
6. Providing solutions for a multi-vendor ATM:
   The ATM platform is constructed based on standard specifications, such as Windows NT and WOSA / XFS, so that Oki's middle software and AP section can be portable for ATMs made by other companies. This allows the distribution of ATM software for non-Oki hardware.

3. Introduction of Object Oriented Development Method[5]

Object oriented development was adopted because the critical issue was just how smoothly specification changes could be handled during the 15 ~ 16 year life cycle of ATM software. During this time, a generation of development engineers may change. Critical here is whether over time new engineers can understand previous development procedures to manage and maintain ATM software. The conventional method of dividing into modules and interface specifications, which depends on the ability and experience of individuals, cannot satisfy this requirement. An object oriented design approach, on the other hand, can permanently implement a stable interface by modeling an interface with objects having individual physical characteristics. This concept matches well with the trend that sees object environments expanding network wide, with ATM services expanding accordingly.

3.1 Object Oriented Design and Real-Time Capability

Two problems of object oriented design are the deterioration of capability and engineers. For the former, we obtained a sufficient real-time capability by validating the product in advance through prototyping, including the general purpose OS itself. Noteworthy in this evaluation was the subdivision of objects. As for the latter, the use of object oriented design and C++ is a major challenge for ATM engineers, because they have been developing ATM using a dedicated OS and dedicated language, ASM/PLM, for a long time. Therefore, we decided to use object oriented design only for the middle software layer, where use of object oriented design is most effective. In this way the above problems were solved. We also taught object oriented design primarily to engineers engaged in basic design.

3.2 Efficient AP Development

Conventional ATM software has been developed by a structured programming method. The concept of this method is based on functions, where functions used for the AP layer are constructed as a package function module, and the module is used from the AP layer, as shown in Figure 2. This method, however, has problems in the way it handles different specifications, which depends on the financial institution, and in productivity when an AP is developed. This caused us to shift our attention from functions to data, and we adopted the method shown in Figure 3. With this method, an AP can be developed by requesting processing to the object modeling media (e.g. a card, passbook) of an ATM. To produce objects we decided on the following policy.

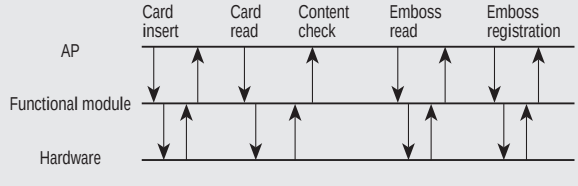

Figure 2: AP development based on conventional method

*2. Visual Basic is a trademark of Microsoft Corporation in the USA and other countries.

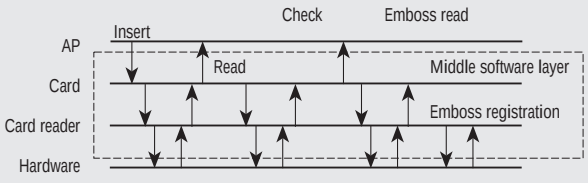

Figure 3: AP development based on object oriented method

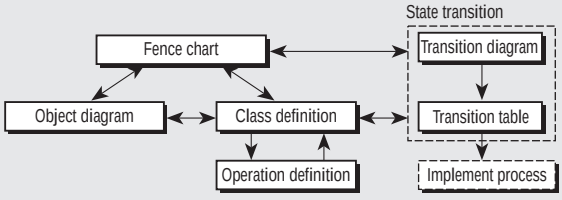

Figure 4: Object oriented analysis & design model

1. Making the development of a transaction AP efficient by capsulation:
   Information management / detailed processing, such as equipment management, is encapsulated inside an object.
2. Component based / reuse of components:
   Objects are produced according to the information to be handled: media information, equipment information, communication information, screen information and customer operation information. If customization for each financial institution is required, then the inheritance function and function replacement mechanism by dynamic linking are used. This decreases the number of processing requests that an AP must issue, and makes it possible to develop the basic structure of an AP by the following methods.
   A. Time and sequence to use middle software objects are controlled.
   B. The result of control from the middle software layer determines the next operation that is executed.

3.3 Object Oriented Analysis / Design Model

The analysis / design process that we adopted is based on the integration of top down operations and bottom up operations, which involved analyzing and designing ATM software according to the OMT[2] method, and rearranging the current ATM specifications. The development of the design document is described below using the model shown in Figure 4.

1. Object diagram: Arranges the relationships among objects (correlation in an operation request, data exchange, etc.).
2. Transition diagram: Arranges the behavior of objects based on the major events to be processed by each object.
3. Transition table: Arranges the transition diagram in a table format. The behavior of objects for all events, including abnormality processing, is arranged here.
4. Fence chart (timing chart): Arranges the role of each object and interface among objects in a major processing flow.
5. Class definition: Defines an attribute to be held and managed as a class, the relationship with other objects, and operations (and internal processing) to be accepted as a processing request.
6. Operation definition: Defines the content of processing and the input and output parameters of operation functions based on conventional ATM specifications.

3.4 Configuration of Middle Software Layer

Figure 5 shows the configuration of the middle software layer. The AP interface layer is used by AP developers, and consists of objects to request processing to an individual object of the middle software. The information object layer consists of objects that hold, manage and operate various information handled by an ATM (checking, editing, collection, etc.), and objects that process an operation request to IO objects. The IO object layer consists of objects that request an operation to each IO equipment (driver), and objects which process attention and response from that equipment.

4. Supporting WOSA/XFS[6]

4.1 Implementing Portability by WOSA/XFS

ATMs in Europe and America have completely different functional specifications than ATMs in Japan, for example, outside Japan a deposit can be made at an ATM using an envelope. The extension specification of WOSA/XFS for ATMs in Japan was established by extending the specifications created by vendors in Europe and America, so that the specification could be applied to ATMs in Japan. The aim of this was to insure the portability of software on WOSA/XFS. If merely improving portability is important, then a

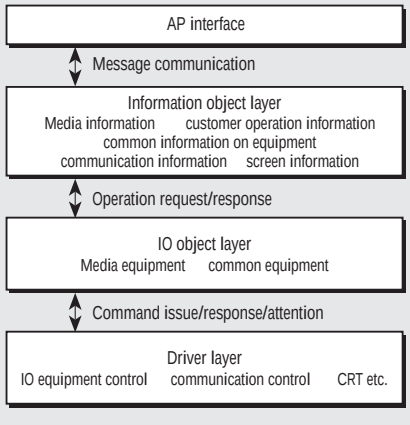

Figure 5: Configuration of middle software

rigid standardization (less range of selections) is best. However, ATMs in Japan have special features, such as the method of circulating coins and paper currency, and these features are the result of the efforts of respective vendors to differentiate their product from others. Therefore, the specification of WOSA/XFS for ATMs in Japan had to be defined such that respective vendors could implement their own features within the specification, not just by adding functions to the original specification. To meet these conflicting demands, special functions depending on the vendor were defined as "selectable" (this function is called "Capability"), so that portability could still be insured.

4.2 Support of WOSA/XFS for Middle Software

The concept of supporting WOSA/XFS for AT-400N and the results are as follows.

- Assigning different roles to AP section and middle software section:
  Functions that differ, depending on the vendor, are encapsulated in the middle software layer as much as possible, except for functions where the physical sequence is critical to control the movement of two or more media (i.e. ID card and money transfer card) due to the mechanism of the hardware.
- Supporting standard "Capabilities":
  Since the WOSA/XFS specification was recently established, supporting all "Capabilities" is impractical. Only the standard "Capabilities" are supported.
- Matching objects in middle software layer and WOSA/XFS object class:
  The WOSA/XFS specification was established after we completed the design of the middle software. Still, the object class defined for the middle software section matched well with the class of the control target equipment of WOSA/XFS. Therefore, we did not have to perform any major modifications on the middle software to support WOSA/XFS.

5. Future Development Plans

The major purpose of the development this time was to shift conventional operations to a general purpose OS environment. In the future, we will continue development as follows so that advanced services provided by a general purpose OS can be applied.

1. Organizing development support environment for EUC:
   We will provide visual languages to describe the control scenario of AP. For this, a test environment must also be provided to users.
2. Use of intranet (Internet):
   Since automation and 24 hour operation for teller window services cause problems on how to distribute software programs which increases in number. In conventional operations, a system to download and to switch software according to the transaction service is required. However, software for services rarely requested or for services that do not require a quick response need not be installed on a local disk. A dynamic download mechanism, such as HTML (Hyper Text Markup Language), will be more practical for such software. We are planning to access to an intranet (Internet) and to prepare an execution environment for this purpose.
3. Defacto standardization for higher layer API:
   To implement true EUC on an AP layer, not only an OS and development language environment but also a higher interface provided by the middle software must be standardized. It takes time to solve this problem, but discussions will probably begin among users who use multi-media ATMs.

6. Conclusion

We developed software for the AT-400N Automated Teller Machine, aiming at an ATM suitable for the open systems era. Through this development we confirmed many achievements to provide solutions to users, which included the portability of ATM software and the implementation of EUC. We also confronted many new problems, a kind of baptism of fire in entering the open systems era. For example, issues on licensing costs for software to be installed, insuring quality (accenting performance and avoiding bugs), updating versions, and security problems. Formidable experiences like these will help us in our future work.

7. References

1. Yokoyama: AT-400 series Automated Teller Machine, *Oki Kenkyu Kaihatsu*, 168, 62, 4, (1995): 7~14.
2. J. Rumbaugh: Object-Oriented Modeling and Design, Prentice Hall, (1992).
3. Japan WOSA/XFS Council, Working Group, WOSA/XFS Programmers Reference, (1996).
4. Tetsuji Nitta et al: Development of software for banking equipment based on object oriented method—Development strategy / outline - Proc. of 53rd Convention IPS, 4D-10, (1996).
5. Nakazawa et al: Development of software for banking equipment based on object oriented method - Configuration of middle software - Proc. of 53rd Convention IPS, 4D-11, (1996).
6. Ueda et al: Development of software for banking equipment based on object oriented method - Support of WOSA standardization, Proc. of 53rd Convention IPS, 4D-12, (1996).