How to cite: Ridwan, A.Y., Kamili, I. (2015). Designing an Integrated Core Banking System for a Medium-Scale Sharia Banking in Indonesia. Proceedings of the 3rd International Conference on Information and Communication Technology (ICoICT)

Designing an Integrated Core Banking System For a Medium-Scale Sharia Bank In Indonesia

Ari Yanuar Ridwan
Industrial and System Engineering
Telkom University
Bandung, Indonesia
ariyanuar@telkomuniversity.ac.id

Ismir Kamili
Lead System Architect
PT. Ihsan Solusi Informatika
Bandung, Indonesia
Ismir.kamili@ihsansolusi.co.id

***Abstract*— The implementation of an Islamic or Sharia core banking system (SCBS) must transform information technology into an enabler of business by providing an agile platform to achieve corporate objectives. It often forms an integral part of a business transformation program addressing key banking issues. The objective of this paper is to propose the development and implementation of an integrated sharia core banking system. The development of the system is based on a core banking system that is already implemented in a medium-scale sharia bank in Indonesia (with nearly 3 million customer saving accounts). The system is developed using DAF (Database Application Framework) for multi-tier application development and deployment. DAF is a set of development library, development tool and runtime environment that is used to develop the SCBS and is required by the SCBS to run. The result of this development activity is a solid application that supports a sharia bank operation from end to end and is a highly valuable asset for dissemination for sharia economic concepts. The system shall also be available for implementation in a wide range of banking operation scale, from small cooperative bank to large commercial bank. The system shall also be enabled to easily operate in application-hosting business model , to reach the maximum availability of economic scale that will empower small-scale sharia-based business entities to participate in an integrated payment and transaction processing system.**

Keywords*— *sharia, Core Banking application, DAF (Database Application Framework)

I. INTRODUCTION

Today, the Islamic banking system in Indonesia is attaining enormous development. Several modern international and national conventional banks were also enchanting significant concern and starting Islamic banking branches in their organizations, which work in compliance with the specific Islamic Shariah principles. The growth in Islamic banking system in Indonesia can be attributed to Rising discernment among Muslim customers who want to invest and borrow according to Shariah principles while enjoying a full range of banking products and services.

There are two types of Islamic banking institutions in Indonesia: (1) The "purely" Islamic bank which only offers Islamic products and services. (2) Islamic windows set up by conventional banks.

In technology perspective, the implementation of an Islamic core banking solution must transform information technology into an enabler of business by providing an agile platform to achieve corporate objectives. It often forms an integral part of a business transformation program addressing key banking issues.

Innovative new technology solutions have enabled banks to meet the increased demand for these services. Now, virtually every product and service offered by conventional financial institutions have a Shariah compliant equivalent, from loans to mutual funds; electronic payment systems to stock indices

Most Sharia banks today deploy core banking systems to support their business processes and increase operational efficiencies. This requires enterprise-wide planning, commitment and resource. This is more complex in the case of Islamic windows where conventional core banking systems are tweaked to support Islamic banking activities. In these cases, the banks are required to maintain separate entity books for customers and reporting.

An Islamic banking solution must provide financial institutions with the following capabilities [2]: (1) Cover all lines of business such as investments, financing, payments, cards, treasury management, trade finance and anti-money laundering services. (2) Provide customers with a consistent experience across multiple channels and total visibility to the bank's sales and servicing agents, enabling "anywhere anytime" banking. (3)Provide banks with a single view of the customer's activities acrosslines of business and access channels while providing customers with a single view of the bank's products and services. (4) Provide enterprise-wide information relating to regulatory reporting, risk management and bank-wide reporting. (5) Have a profit distribution engine which provides flexibility in pool definition, profit-sharing schemes, revenue reserves allocation and distribution, alongside the ability to perform "what if" scenarios prior to the distribution of profits. (6) Comply with Shariah law, AAOIFI (Accounting and Auditing Organization for Islamic Financial Institutions) accounting standards and international accounting standards (IAS), among others.

The objective of this paper is to propose the development and implementation of an integrated sharia core banking system (SCBS). The development of the system is based on a core banking system that is already implemented in a medium-scale sharia bank in Indonesia (with nearly 3 million customer saving accounts) and will be integrated with proven secure payment / card technology.

978-1-4799-7752-9/15/$31.00 ©2015 IEEE

The layers of functionality in the core banking system can be summarized in the following picture[3]:

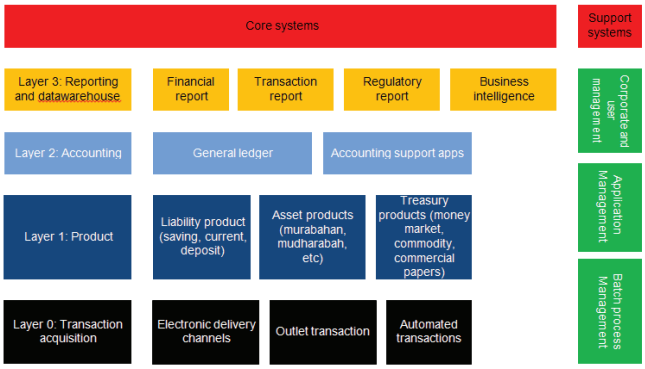

Fig 1. The Layers of Functionality in The Core Banking System[3]

Transaction acquisition layer: Functionalities that are provided to capture data entry / transaction from electronic delivery channels. Some data entry will occur in the back office operation so it is cohesive with product / accounting layer Product layer: Functionalities that are provided to record transaction obtained from transaction acquisition layers with respect to product rules. Products can be classified as liability products, asset products and treasury products. Accounting layer: Functionalities that transform transactions recorded in product layer to the corresponding ledger records. Reporting and datawarehouse: Functionalities that extract information as a whole from all applications and provides various reporting for both internal and external entities.

II. Project Overview

The development of SCBS is based on existing core banking solution that already works in Bank XYZ and is currently serving more than 3 million saving accounts. Fig 2. Shows the scope of the current system. The system currently running has the following application modules:

1. Corporate and user management module
   Handles organization data, user management, user roles, access rights, and application management .
2. General ledger module
   The general ledger module comprises general ledger transaction entry, cross-module transaction entry and interface to other module for accounting transaction posting
3. Accounting support modules These modules provide additional details to certain accounts in the general ledger module: inventory, fixed asset, amortization module, tax management, budgeting,
4. Reporting system
   Includes daily transaction reports, MIS reporting, and central bank reporting
5. Liability module
   The liability module handles teller transactions, customer base data, clearing system, bulk transactions, batch and end-of-day process, and one of the most important, interface to connect with delivery channel, accounting, payment, and card management module.
6. Debit card management module,
   The debit card management module comprises support for card lifecycle activities from production, inventory, registration, selling (for instant cards), delivery channel interface and call center support (can be integrated with the CTI system).
7. Additional applications like database gateway interface (for direct, controlled access to the database for system administrators and system maintainer), a simple datawarehouse for transaction analysis reporting.

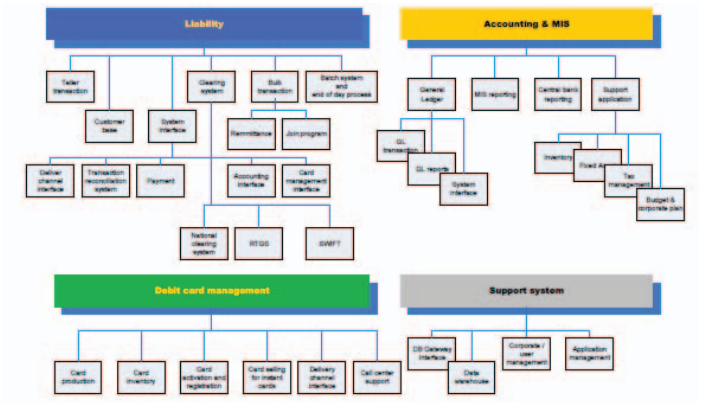

Fig 2. The Scope of The Current System

The aim of this development activity is to improve the currently implemented CBS into a more complete and integrated solution, along with better and more reliable application technology foundation. The technology developed to establish the CBS consists of two parts: First The backend technology, containing application server, application delivery model, security infrastructure and data reliability tool. And second The frontend technology containing application data model / design, functionality and architecture.

Table 1 shows the gap analysis of the currently implemented SCBS and the planned version is given in the table below. Gaps are identified for each aspect, either on the back-end technology or on the application level.

Table 1. The gap analysis of the currently implemented SCBS and the planned version

Aspect	Currently implemented version	Planned, improved version
Back-End Technology		
Development language of application server platform	The data access and business layer API is developed using Pascal / Delphi language	The new platform will be fully in developed in C++ and Java for maximum portabilitiy
Server architecture	Monolithic, integrated-in-1-place I/O handling and logic processing	Dual-system architecture, separated I/O and handling and business logic processing. Enabling higher transaction processing rate and higher degree of distributed system
Business application development	Plain (no encryption on network message)	Encryption, client authentication with secure socket connection
Client cache security (for client-side content)	Plain, no content signing	Content signing with server digital certificate to ensure no illegal content modification
Database security	Allowing easy access to the production database	More secure and regulated access to production database
Data mirroring	No application-level data mirroring	The data mirroring would also provided in application level to allow
Application Functionality And Coverage		
Application modules	Accounting Liability / customer base Debit card management Simple interface to external system User / corporate management	Add the following modules: • Treasury module • Financing module (murabahah, mudharabah, musyarakah, ijaroh, bai-al-inah) • Web servic e for integration with external system
Support applications	Simple, scattered support applications	More integrated, some important support applications like interbank clearing, reporting and remittance will be integrated with the accounting system Better integration of application support to the accounting and liability module
Accounting module	Accounting transaction entry extension is limited only to liability module Loose control on suspend / intermediate accounts	Accounting entry extensions can be used by other modules as well, like financing, and support applications Strict control on suspend / intermediate account usage, to prevent unsettled transactions
Card management	Only support debit magnetic stripe based cards with simple security control	The new card management will support higher security control, HSM and compliance to card issuing standards, it will be integrated with current proven Challenging Profile's card management
Reporting and datawarehouse	Limited to current user requirement using dedicated code	Flexible reporting and data mining, and can be self-defined by user
Business Model		
Business model	The application is now designed for single infrastructure implementation per customer.	The infrastructure requirement will be designed to enableinfrastructure sharing, and hence one server infrastructure can be shared among several banks, to create service-based business model. This is the more preferred business model in the future rather than license-based business

III. DESIGNING BASE / BACK-END TECHNOLOGY

The technical description of this development project is divided into three sections: Base / back-end technology: This section describes networking and application server technology layer that serves as the foundation / platform of all applications to run. Application: Due to the SOA application model, the system is divided into some independent yet interconnected applications. This section describes about DAF (database application framework), a product of Ihsan Solusi that underlies the application development of the SCBS [2]. DAF is a set of development library, development tool and runtime environment that is used to develop the SCBS and is required by the SCBS to run. The description of DAF will include its technology aspects, current state of implementation and the targeted state of implementation in the new version. Aspects of back-end technology are:

1. SOA application model
2. Application delivery
3. Messaging
4. Security
5. Development tool
6. Application server strategy and platform availability

DAF is intended for multi-tier application development and deployment. Application in DAF is divided into two portions: client-side part and server-side part. The client-side part comprises of user-interface definition and scripts, while the server-side part comprises of business logic and client request processing. All DAF applications on the server-side functions are automatically by itself remote-invokable by other DAF applciations. Even the user-interface in a DAF application can be remotely attached to other DAF application, transparently, without additional programming effort. This natural support for SOA helps very much in a logically complex application like the core banking system.

Using SOA, the development can be splitted into smaller parts and modules without sacrifying the whole logic and data integrity. This nature also helps DAF applications to be quickly integrated to other legacy applications and external systems, including transaction switching for electronic banking / delivery channels. The improvement of this part is intended to create more robust and higher processing capability (usually measured in transaction per second / TPS). Another improvement area are service delivery management and control, and more extensive transaction logging and log analysis for audit requirements.

DAF uses a specific client application called DAF client application. The client application acts as client entry point to access application on the server. The application client will require user login information (user name, login, password, application name and configuration name) and then authenticate the login information to the server. If the login information is valid then user is given further access to application on the server. The application definition on the server consists of two kinds: client resources (like user-interface definition and user action scripts) and server resources (basically server-side procedure). Client resources are downloaded to client as the resources are required by the current application in client. Server resources (server-side procedure) can be invoked by application script in client through a defined Application Programming Interface (API).

The server resources are invoked when client requires certain data to display to user or when client will post transaction data to be processed in server. In the current version, application client can only interpret only limited amount of content types. The currently supported client content are menu definition, form definition and client script developed with DAF development tool. In the improved, planned version, the application client will be able to interpret various content type (which can be developed using various development tools), using dynamically configured plug-ins.

Kim, Tao, Shin, & Kim [4] classify security perceptions of customers during e-payment systems into 3 categories. security statements; transaction procedures; and technical protections. They claim that Security statements are important to enable trust in the customer with respect to e-payment systems. They found positive association between users' perceived security and their use of e-payment systems. Yoon [5] found that security in online banking strongly influences customer satisfaction. The importance of security and privacy for the acceptance of online banking has been noted in many banking studies [6].

In the current version, no encryption-technology-based security is applied to DAF. The only available security authentication is based on user ID / password (which later composed into a session ID). This user ID / password can be checked against custom database or external service like LDAP. In the next version, security model will be applied to the following areas:

1. User/terminal authentication: Aside user ID and password, server also requires terminal / user ID to be authenticated. This can be achieved using client certificate and SSL handshake or using a pre-shared key for each user. To be more advanced, the client certificate / pre-shared key can be stored in smart card.
2. Network data encryption: After handshaking is established, all client-server communication also will be encrypted using dynamically assigned symmetric key.
3. Message authentication (for transaction): For transaction, each transaction will be assigned a unique message authentication code (MAC). This is to ensure that every transaction originates from valid user / terminal rather from a bogus node on the network.
4. Client content signing: Transportable code is one of the most dangerous source of security threat. Hence all client-side code downloaded from server (including the cache contents) should be signed to ensure integrity of the code.

DAF uses a self-owned application development environment (IDE) called DAF AppStudio. DAF AppStudio is used to develop user menu, user forms, scripts, metadata, and other application objects. The scope of the IDE is complete, however to enable better productivity, it would be much acceptable if developer can use open-system IDEs like Eclipse

or NetBeans to develop application. The strategy is also related to acceptable application content on the client-side.

Because currently only propietary content is enabled, only propietary IDE can manage the development. As the content on the client-side will be shifted to open content, open-system IDEs will also take larger portion of the development.

DAF application server uses "spawn" process server model. There is a single process accepting connection request from client, and as a connection arrives, a new process will be spawned (called worker process) and handle entirely the communication of the connection. Each worker process will load the script required by user action, connect to the database and process user request as defined by the script. Hence each worker process has two tasks: handle network messages from specific client connection and execute server-side code according to user request (using script engine).

The weaknesses of this approach are:

1. As the client terminate connection upon request completion (this strategy is to save network and memory resources), the worker process will also terminate (it cannot be reused , because no relationship is maintained with master process). It means that if a new request comes, the whole executable and libraries of worker process will need to be reload, including reloading and reinitializing the script engine and connection to the database. The overhead that occurs in this initialization will highly burden the application server's performance (even though it is already minimized to an optimum level)
2. Server load capacity is uncontrolled. The master process will continue to spawn worker processes regardless of how many worker processes already working in the server. This may cause performance penalty to the server when requests come in unanticipated frequency rate.
3. Worker process and master process have to reside in the same machine. This means that all libraries and execution capability of server code (which is part of the worker process) is limited to the platform where the master process is available, hence limiting platform availability. And moreover, The new approach suggested to overcome the problem of current application server strategy is to split the application server into two parts: the network message handler part and the user-code-executor part. Another change is to make the master process more intelligent by managing available worker processes (instead of just spawning and let the worker processes go). The scheme of new approach of the application server is shown in the following figure:

The mechanism of the application server is as follows:

1. The network handling is carried out in a process called switch process. This switch process opens two ports, one for client connections and one for worker process connections.
2. Worker processes are created independently from switch process (yet they must be available before client requests are coming). The number of worker process depends on the service level required. These worker processes are launched by service manager and the service manager will keep the number of worker process constant, if a worker process is unexpectedly terminated, the service manager will immediately starts another instance of worker process.
3. Each worker process is connected as network client to the switch process. A worker process will continously wait message to be delivered from the switch process (using simple blocked I/O) , process the message using pre-loaded script engine and database connection and send the reply to switch process.
4. The switch process handles incoming client connections in client processing threads (these threads are pooled in a limited amount instead of created dynamically for each client request). Each client request will be put in request queue, while another thread in the switch process detects idle worker processes. If a worker process is available, this thread will consume a request from queue and then 'feed' it to an idle worker process.

Benefits of this new application server strategy are:

1. In each worker process, script engine and libraries are loaded and database connections are created only once during initialization. This dramatically reduces processing time of each request compared to current approach.
2. The number of worker processes is limited, and can be defined according to the desired service level. Request overflow will not create excessive number of worker processes, instead overflow requests will be rejected.
3. Because connection between worker process and switch process simply uses ordinary TCP/IP network connection, the worker process can reside in the different machine with switch process. This opens the possibility of better distributed architecture and to separate further development of worker process and switch process.

IV. Designing The Application functionality model

From the functionality point of view, the SCBS design is intended to cover three dimensions of service delivery in he banking industry: product, delivery channel, and support system. This three dimension model can be described briefly in the figure 5.

The improvement will target to complete missing "blocks" in this application functionality model. The approach will also need changes. Each part of the solution must have capability to work flexibly with parts from another 2 dimensions.

Traditional core-banking system not only has high development cost but also acts like an information island and lacks a more flexible business model where its subsystems are independently established and hard to share data with each other [7]. To perform an integrated service for the bank, all modules have to be interconnected. Thanks to the SOA application model so that all of these modules can be interconnected smoothly with only small development effort (compared to development effort of each module).The central of application interconnection lies in the corporate

management and general ledger module. The corporate management module verifies all of user's integrity, and serves other information such as organization structure, authority level and user setting parameters to other modules that require them.

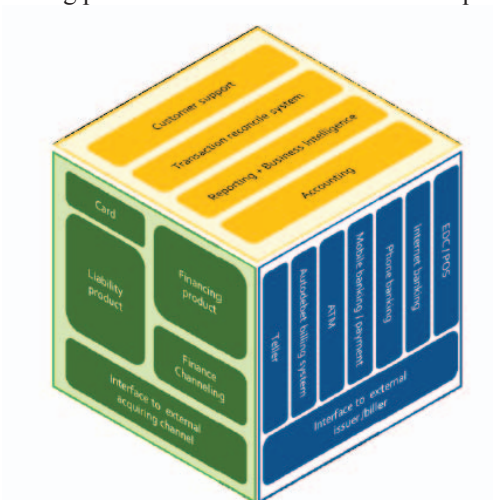

Fig 3. Dimensional of Service in Banking Industry

The general ledger module serves as the accounting settlement process of all transactions in other modules. In currently implemented CBS, the general ledger module also has one additional capability to initiate transaction and integrate all transactions from other modules. This capability is particularly required in back office transactions where complex transactions may be composed of several transactions in other modules. This capability also reduces discrepancies and inconsistency between general ledger records and detail transaction records.

In the current version of CBS, the only related business model on the software is by selling the license, implementation project and maintenance contract. By improving the functionality and the core technology it is possible to virtually separate one server infrastructure into several logical paritions used by several entities. Using this virtualization capability, the business model could be extended to rental / pay per use business model. This new business model is much more flexible then the license-based model.

Another technical issue that affects the business model is to remove the dependency from current ORACLE database to license-free databases like PostGreSQL. This migration will highly affect total cost ownership (TCO) of the system. In this CBS, ORACLE is the only external dependency factor with the most significant price. Once this cost can be eliminated, the TCO of the system will dramatically be reduced. The CBS rental can also be combined with delivery channel rental to create a total banking solution for both transaction acquiring Current development status.

V. Conclusions

The following elements are already developed as initial efforts of the SCBS improvement:

1. There is the gap analysis of the currently implemented SCBS and the planned version. Gaps are identified for each aspect, either on the back-end technology or on the application level.
2. Part of the new application server architecture using switch and worker processThe new architecture proposed for the application server is already tested in a simplified form in a switching application API. This switching application API is already tested for several purposes like ATM switch and mobile banking server. The next stage is to extend this switching API to more flexible application server messages and actions.
3. The design of murabahah financing applicationThe design that includes data design, transaction model design and user-interface design has been developed for murabahah financing application
4. Part of accounting support application integration API The initial form of integration API for integrating accounting support applications to the general ledger module has been developed and tested for simple integration cases.
5. In the current version of CBS, the only related business model on the software is by selling the license, implementation project and maintenance contract. By improving the functionality and the core technology it is possible to virtually separate one server infrastructure into several logical paritions used by several entities. Using this virtualization capability, the business model could be extended to rental / pay per use business model. This new business model is much more flexible then the license-based model.

VI. References

[1] Hamdan, Charlie, 2009. "Banking on Islam: A Technology Perspective", Islamic Finance News.
[2] Ihsan Solusi Informatika, "DAF:Database Application Framework", Unpublished.
[3] Ihsan Solusi Informatika, "i-Core Banking System", Unpublished.
[4] Kim, C., Tao, W., Shin, N., & Kim, K. S. (2010). An empirical study of customers' perceptions of security and trust in e-payment systems. Electronic Commerce Research and Applications, 9(1), 84-95.
[5] Yoon, C. (2010). "Antecedents of customer satisfaction with online banking in China: The effects of experience". Computers in Human Behavior, 26(6), 1296-1304.
[6] Hernandez, J. M. C., & Mazzon, J. A. (2007). "Adoption of internet banking: proposition and implementation of an integrated methodology approach". International Journal of Bank Marketing, 25(2), 72-88.
[7] Yung-Hsin Wang, Shih-Chih Chen, and Ping-Hsin Peng, "Applying Service-Oriented Architecture To Construct The Banking Letter Of Credit System Integration", International Journal of Mangement Research and Business Strategy, Vol. 2, No. 3, July 2013.