Web Centric Evolution Of Legacy System

Dr. Shahanawaj Ahamad, MIAENG
Department of Computer Science College of Arts & Science in Wadi Al-Dawasir, King Saud University
Wadi Al-Dawasir-11991 Kingdom of Saudi Arabia.
sa_sum@yahoo.com

Abstract— Today’s I.T. Organization has made huge investments in mainframe-based applications. As a result, these legacy systems contain tremendous hub of knowledge that drive daily business operations. Most of the day-to-day transactions that occur in large organizations are woven with complicity around these systems, which serve the current demands. With the increasing popularity of Internet, business boundaries have diminished. Every business organization wants to reach the maximum number of customers through the web presence and it is also observed that those who has done this now entertaining whole world. So it is more demanding now to evolve a legacy system over web. This paper produces solutions to various technological challenges including reusing existing logic to support current business process, reusing existing process to reduce costs, reusing existing system design to ensure dependable and reliable solution, reusing existing system’ elements for faster deliver and challenges need to be dealt in evolution of a legacy system over web.

Index Term-- Evolution, Legacy, Middleware, Reuse.

I. INTRODUCTION OF EVOLUTION ISSUES
Extending the capabilities of legacy over web and evolving it on this architecture faces some major issues which are identified and described below.

A. Monolithic Architectures
 Technologically, the constraints faced in web centric evolution of legacy system are due to its architectures, programs that have tightly integration with user’s interfaces, business processes and data processing logic. When transitioning to a web based system, the traditional user interfaces must be replaced with suitable web based user interfaces. Separating the legacy system’s interface from the data processing logic and business processing is not an easy and small task.

B. Developer’s Skill
 The lack of experienced developers who understand both legacy systems as well as trends in conducting business over the Internet are one of the largest challenges organizations are facing. Employees working on legacy systems have a tremendous understanding of organization’s business processes. Conversely, Web technologies are so diversified and evolving that attaining these skills so early is impossible. The answers to these challenges lie in solutions that re architect monolithic systems, establish communications between legacy systems and Web-enabled platforms, and provide legacy data access to other systems to support their business processes. These same concepts can be extended to Web-enabling non-mainframe-based legacy applications as well.

C. Database
 Legacy database contain terabytes of information that feed business processes. Obsolete access mechanisms used by these databases do not allow data to be shifted outside these legacy systems to web. This leads to the need for redundant data storage in other systems to support the business processes implemented in them. Developing mechanisms to access this legacy data is crucial.

D. Extended User’s Profiles
 Most Web-based systems are designed to reach the maximum number of users. In some cases, the goal is to reach the end customer directly. This creates a need to address a broad range of user abilities. Operators may not be present at all, leaving customers/end users alone to interact directly with the system. These types of situations often demand a restructuring of workflows and simplified presentation layers.

II. RE-STRUCTURING
As mainframe legacy systems can be as batch or online systems. Online systems typically offer more benefits once be Web-based due to their higher interaction with end users. These online systems are prime candidates for evolving over Web, it is important to integrate the background batch systems with open systems to carry out the necessary business-to-business integration. Batch systems also produce vast numbers of reports, which are well suited for Web-based evolution.
Re-structuring an existing system and reusing it has several advantages. It ensures a dependable, quick and cost-effective solution compared to new development. Additionally, through re- structuring, business processes become more modular and exposing sub-modules that can be reused. Re- structuring also provides additional advantages:
• Existing business process automation can be determined.
• Redundant functionality can be identified and eliminated.
• A new automation baseline for supporting business processes can be established.
• The system becomes more manageable because of clean code.
The tightly coupled user interface, business logic and data manipulation logic of the typical legacy system can be re-structuring to separate the user interface from existing programs. Once separated, the business logic can be
appropriately wrapped and invoked from middleware. Newly developed middleware provides an integration facility for Web-based GUIs and wrapped business processes. The principle reason to re-structuring an existing system is to make efficient use of its “business rules.” Extracting and transiting these business rules for use on the Web can be accomplished through the re-structuring process.

III. RE-STRUCTURING PROCESS
Process to re-structure an existing legacy system for evolving over Web has been phased as given in this section.

A. Assessment
Most system re-structuring efforts first need to identify all of the individual components, e.g., individual programs, copybooks, jcls, procs, delgens, etc. contained in the existing system. It is essential that boundaries are drawn for each system involved in the re-structuring process to ensure that appropriate bridges, which are required at various interface points can be identified so they can interact with the other systems. Identifying the latest copies of source code is also very important since the existing system is being restructured for reuse in the new architecture. An analysis of system components must be conducted to ensure the system is complete in terms of all of its components.

B. Understanding
To re-structure and simplify existing processes, it is essential to understand the existing system’s structure. This needs to be developed at various levels of abstraction, e.g., system-to-system interaction to identify bridge points as well as the lowest level of program-to-program interaction. Typically, the system documentation is the best place to gain this understanding. Also, the business matter experts should be a reliable resource. Unfortunately, with most legacy systems, code modifications are not consistently documented. In addition, the business matter experts often cannot remember all of the details contained within the business rules of a periodically maintained system. This leaves the living, breathing legacy source code as the only reliable source of information capable of offering the clues necessary to understand the system. Fortunately, advanced language-processing algorithms can be used to help extract needed information from the legacy source code. Automatically generated Call Graphs explain program-to-program interaction. This system-level understanding assists in the process of application partitioning, which is where decisions are made for grouping elements of the systems at highest level of coupling. System-level views also assist in understanding the overall workflow, which needs to be simplified as a part of the evolution efforts. Additionally, program structures - how programs modify data - need to be understood. Control Flow Graphs, Intelligent Code Navigation facilities and Structure Charts are very helpful in carrying out this activity.

C. Component Identification
Advanced data-centric and event-centric algorithms are the keys to extracting business rules (which are the segments of a program performing an intended business function) from monolithic legacy systems. The event-centric approach drives rule extraction based on events triggered by either the user or the system. The data-centric approach focuses on rule extraction, making the data the focal point for identifying processes working on this data as business rules. A repository facilitates registrations of such identified business rules and the association of relevant system components with it. Documentation can be attached to these registered business rules to make them self-explanatory.

D. Component Extraction
Business rules can then be extracted in the form of callable services. These extracted rules are independent functions that, if all necessary input parameters are provided, will execute business rules and generate the appropriate output. Extracted business rules can be grouped together based on their contribution to achieve the intended business functionality. This group of rules, which is processing some common set of data to achieve intended business functions, is called legacy components. The number of rules that need to be grouped together is a matter of choice, depending on the desired granularity of the component. A callable interface needs to be provided for these components. The encapsulation of business rules in the legacy component is a major step in re-architecting. These easily maintainable legacy components can be reused in new development efforts on legacy platforms as well as shifted over the Web through an interface.
E. Wrapping Components

While the componentization process transforms monolithic legacy system architectures into reusable, maintainable, component-oriented architectures, it may not be necessary to shift all of these components directly over the Web. It’s wise to identify the individual components that are necessary to deliver the functionality that’s required over Web. These components, in turn, can continue to interact with other supporting components to drive all business processes.

Legacy platform vendors provide various clients and APIs to facilitate call-level access to legacy processes on these platforms from the external, open world. The automatic generation of wrapper code for these legacy components should adhere to the standards of more generic component architectures based on COM/CORBA/EJBs. Calls to legacy components are automatically generated in wrapper code. These generic components are more accessible and can be integrated with Web technologies. Some of the typical options available to connect to the legacy component from the wrapper code depend on the locations of middleware with respect to where one wants to access these legacy services.

F. Developing Integration of Components

Middleware application servers facilitate the development of code to integrate wrapped legacy components and separately transformed Web favorable GUIs. The integration phase also provides a chance to improve and simplify workflows that are required for deployment of an application over the Web.

G. Connecting Middleware

There are three aspects for middleware as

- Middleware on the mainframe

Various application servers available on mainframes facilitate Java-based development. They provide connectors to legacy runtime environments based on Advance Program-to-Program Control (APPC), External communication Interface (EXCI) APIs. APIs can be used to invoke the callable services or transactions deployed in these legacies runtime environments.

- Middleware outside the mainframe

Application servers on these platforms allow development of middleware services based on COM/DECOM, CORBA, EJBs etc. The main difference compared to middleware on mainframe is the access protocols supported from such external platforms. Most legacy vendors provide connectors/clients to access their legacy environment from outside the mainframe. For example, CICS services can be accessed via an External Call Interface (ECI) provided by a CICS client.

- Message-oriented middleware

Often legacy system architectures are more asynchronous in nature than synchronous (immediate response). These are the applications where operators enter data and once basic validation is done, that data is queued for further processing. These systems are good candidates for message-based system re-architecting.

The backbone of these re-architected systems is messaging software. Programmers using this software don’t need to be aware of where the target application resides physically. Using underlying transportation mechanisms, this software ensures guaranteed delivery of messages to specified destinations. Architecturally, user interface programs format data appropriately in packets and attach the address of the target applications. Target applications are generally listeners/dispatchers located on the mainframe, which, depending on the message type, route the message to the appropriate target business process. XML is gaining popularity in areas of formatting data in packets. XML makes the data content of the packet self-explanatory and reduces structural interdependence between individual applications.

Legacy components, which are self-sufficient and do not need to interact with other components, can be then automatically transformed to modem platforms like Java or C++.

H. Transformation to a Web GUI

Automatic transformers can convert legacy user interfaces, like CICS maps, to Web GUIs, like HTML or Java applets. With slight cosmetic modifications, automatic transformation gives one-to-one mapping between screens and translated GUIs.

IV. EXTENDING USER INTERFACE

The addition of a Graphical User Interface (GUI) is a quick solution because the external presentation layer is simply added on top of the existing user interface and the original legacy system remains unchanged. Called GUI Extension, these solutions facilitate the generation of a single, Web-based user interface that interacts with the data from multiple screens contained within the legacy application. However, changing the user interface of a legacy system has a rippling effect on the design and implementation of a new Web-based user interface. Keeping both user interfaces synchronized is
tedious and, since the existing system is not changed, modifications to workflows to suit the Web are not possible. The quickest way to legacy over the Web is through the use or GUI extension. They are often referred to as “screen scraping.” Quick and cost-effective, GUI extension does not require modification of the legacy application but, instead, are based on the processing of the data streams that the mainframe uses to communicate with terminals. The concept behind GUI extension is to facilitate translation of the screens generated by the mainframes to a Web-suitable format.

A. Terminal Servers
Terminal servers are the simplest form of extending legacy screens over the Web. Terminal servers like Microsoft Terminal Server or Citrix WinFrame terminal servers, host terminal emulation software that interacts with the mainframe. For each Web-based user, a session is created on the server and a copy of the terminal emulator screen is displayed over the Web on the user’s screen. Though simple, this solution demands powerful and costly servers. Also being the central point to initiate and maintain communication, it can become a single point of failure, e.g., during a server crash, no one can get access to legacy applications.

B. Java Applet or ActiveX Components
In this type of the solution, an HTML page contains an embedded Java applet or ActiveX components. These pieces are then downloaded based on user demand. They provide the basic functionality of terminal emulators. Once downloaded, they establish a direct connection to the mainframes, bypassing the Web server. Applets interpret the contents of 3270 data streams and display them as a green screen within their HTML page. Control keys and function keys are mapped appropriately to buttons or mouse events. When it comes to selection between applet or ActiveX-based terminal emulators, various points need to be considered. By embedding applets in a browser page, any browser supporting Java can be accessed, even when using diversified platforms like Mac or Unix. On the other side, ActiveX components work only on Windows, which, of course, limits users to Windows as browser platform.

The advantage (or in some cases disadvantage, when security is a consideration) of ActiveX emulators is their ability to access local services like print services or file services.

C. Host Integration Servers
Host Integration Server is a breed of software that works along with Web Server. The main concept behind this solution is to intercept 3270 data streams generated by legacy applications and convert them on the fly to HTML that can, in turn, be displayed in browser. The advantage of these solutions is that they do not have deployment costs on the client since the HTML pages are served up on demand. Also, the fact that they’re based on HTML means that the user can be on any platform supporting browsers.

Recent advances in host integration services facilitate the maintenance of the state of each individual user session, which helps to re-sync sessions in case the user performs some asynchronous operation on the browser. These servers also provide API, which operates on the 3270 data stream. These APIs facilitate access or manipulations of data fields on 3270 screens non-visually. APIs can be used to develop applications in conjunction with Web- favorable UI platforms like Java. For example, Java applications can extract data from 3270 screens to present it in a dynamically generated html form or applet field. This approach provides better control over screen rejuvenation by facilitating manipulation of multiple 3270 applications to form a single, Web-based graphical user interface. Conversely, redistribution of data entered by the user to the appropriate 3270 application is handled programmatically by the host integration servers.

V. CONCLUSION
Majority of business data processing is carried out on mainframe-based legacy systems. Approximately 80% of the data resides on mainframes and is processed within these legacy systems. It is much more cost effective, faster and more reliable to evolve these existing systems for web rather than to develop new one, capable of reaching new Internet-based markets.

The best way to Web centric evolution of legacy system is the use of methods that facilitate the decomposition of COBOL-based systems into individual legacy components that contain just the business rules without the green screen user interface. These legacy components can be invoked synchronously by making use of client APIs, or they can be invoked asynchronously by making use of message-oriented middleware. The components can then be wrapped by middleware technologies like COM, CORBA or EJBs to enable their reuse on a variety of open architectures. Legacy screens can be automatically translated to Web-favored user interfaces. In situations where quick and cost-effective solutions are necessary and the attractiveness of the interface is not critical, extension of GUI can be the right tracked evolution. Terminal servers and terminal emulators based on Java applets or ActiveX components are more primitive types.
of GUI extension services. It is very useful in web integration of legacy based businesses services.

REFERENCES


ABOUT THE AUTHOR

DR. SHAHANAWAJ AHAMAD is an active academician and researcher in the field of Software Reverse Engineering with experience of ten years, working with King Saud University’s College of Arts and Science in Wadi Al-Dawasir, K.S.A. He is the member of various national and international academic and research groups, member of journal editorial board and reviewer. He is currently working on Legacy Systems Migration, Evolution and Reverse Engineering, published more than twenty papers in his credit in national and international journals and conference proceedings. He holds M. Tech. Degree in Information Technology followed by Ph.D. in Computer Science major Software Engineering, supervised many bachelor projects and master thesis.