A Proposed Framework for an Effective Integration of Supporting Environments for Smart Client Application Development

Santhosh John
Asia Pacific University College of Technology and Innovation
Lot 6, Technology Park Malaysia
Kuala Lumpur, Malaysia.

Maisie Han Wi-Mei
Asia Pacific University College of Technology and Innovation
Lot 6, Technology Park Malaysia
Kuala Lumpur, Malaysia.

ABSTRACT

In the past, traditional clients were built as stand-alone applications that resembled islands of functionality. As a result of this tight coupling between clients and their related applications, software developers/users have become frustrated by the increased complexity involved in the development/usage of client applications. In the late 20th century, the birth of the Internet gave way to the development of thin clients which are easily developed and deployed. However, the trade-off was that its user experience was not as rich as its predecessor and it always had to be connected via network. By evaluating the benefits of both clients, a new type called smart client emerged. The newly-evolved client would be rich in functionality like thick clients and easier for deployment like thin ones. As with any software initiative, a framework is required to provide the necessary guidelines that will lead to the development effort towards the correct goal. The framework would have to be founded on a methodology that becomes the underlying philosophy for all activities. Besides that, the implementation of the framework would have to be based on the effective usage of CASE tools that provide an environment to capture artifacts. These artifacts will eventually be used as the blueprint for development work as well as to provide traceability back to the original request.

Keywords: Framework, Information Systems, Software Development, Methodologies, Smart Clients, CASE tools.

I. INTRODUCTION

The smart client is a sure-bet as the next generation application client. So, to accommodate efficient development of such a technology, we must adopt the proper framework that will help the team achieve maximum productivity without rigid rules that will hold the team back. As suggested, RAD and Agile methodologies are two highly recommended methodologies that we can follow. Also, there is a need to effectively use CASE tools not only in a selection section of a life cycle but in every phase of the smart client development process. This research paper aims to suggest a framework based on suitable software development methodology and smart usage of CASE tools at each phase of a smart client application development.

II. EVOLUTION OF CLIENT APPLICATIONS

During the mid 1990s, there was a huge surge of rich client applications developed on platforms such as the popular Microsoft® Windows Operating system. The development of these rich clients was based on taking advantage of the local hardware resources and the features of the client operating system. Despite their impressive functionalities, limitations were not lacking. Most of these applications operate in a stand-alone mode with little or no regard to the environment on which they are running. This environment includes other computers or services running on the network, as well as any other applications running on the host computer.

A short while later, two-tier application entered the client applications scenario to open up services for multiple users, in order to access common data residing on the network. Arrival of distributed technologies like RMI, CORBA and DCOM allowed clients/applications to become more distributed. However, the distributed nature of these applications came with a price as their size and its underlying complexity grew. The tight coupling between the client application and the services they consumed also lead to increased complexity, incremental development and maintenance tasks.

Although the internet had a little impact on the rich client applications, it did however provide an alternative to the traditional rich client application, one which promised to solve all the problems associated with application deployment and maintenance. The “Thin client” – a browser based application allowed the applications to be deployed and updated centrally, thereby reducing the problems associated with these two stages of an application. This model also allowed companies to disperse their applications/services effectively and efficiently to a large global and diverse audience.

In addition to the advantages listed above, the thin client applications have their own share of glaring disadvantages too. Constant connectivity to the internet is required for this kind of client applications and this has suppressed the mo-
bile consumer’s access to the application at anytime and anywhere. This has brought about a situation whereby the so-called “mobile consumer” would be required to re-enter data which was captured offline once he returns to his workstation. Excellent features of traditional thick clients such as drag-and-drop, undo-redo are also no longer available consequently reducing the user friendliness of the application. Regardless of these drawbacks, the deployment and management problem with rich client were so big that the trade-off seems worth it. As a result, the thin-client application model has dominated its presence in the recent years.

However in recent years, factors like the wide adoption of Wi-Fi, hardware cost reduction and broadband availability are setting in motion the era of mobility. With regards to this aspect, the thin client’s features are sorely lacking. This is driving the emergence of a new type of client application called the smart client which was designed to combine the beneficial features of rich client applications and the manageability benefit of thin client applications with its rich support for today’s mobile lifestyle.

![Smart Clients](image)

**Fig. 1 : Smart Clients – An intersection between rich and thin clients [7]**

### III. SMART CLIENT: A DEFINITION

The term smart client was coined to distinguish yesterday’s “rich” and “thin client” from the next generation of client applications. These clients often have very diverse requirements but they do share some similar characteristics as listed below: [7]

- **Utilizes local resources** – A smart client application always has the ability to fully utilize the local resources as in either hardware or software on a client PC.
- **Network-aware** – Smart client applications are never standalone. They can consume and use different services and data over the network. In addition, they have the distinct advantage of being able to operate in a non-networked environment.
- **Supports occasionally connected users** - Smart clients take advantage of local caching and processing to enable the operation during periods of no network connectivity or internet connectivity. Smart Clients are network aware where it not only knows how to get “new” data when a connection is detected, but also how to store this data so that it is available when the user is offline.
- **Easy to deploy and update** - Deployment can be done from a centralized server and updates can be done automatically and intelligently utilizing the network aware characteristics.
- **Device independence** - Smart clients are not limited to the desktop. It can also be deployed on portable devices like laptop, tablet PCs, Pads and even smart phones.

Based on the characteristics highlighted above, it is noticed that developing a smart client can be quite a challenging task as its environment is highly dynamic. This results from our observation whereby smart clients often have to perform in a mobile environment that has limited capabilities, subject to various hardware and network protocols and needs to accommodate the rapidly changing nature of devices that host them [1].

### IV. OVERVIEW OF A FRAMEWORK

Frameworks are a widely used term within the context of software development that has various definitions. However these mainly include software components that can be reused or reassembled to create new applications, tools that can be used to customize the applications according to specific requirements, and also rules or protocols for inter-application communications [11].

Frameworks based on the Object-Oriented (OO) concept provide modularity, reusability, extensibility, and inversion of control to developers. One such framework, Microsoft Foundation Class (MFC), has gained prominence over the years from the ubiquitous usage on the Microsoft Windows operating system [4]. Reusable software components of a framework, such as communication protocols and graphical user interface are usually well-tested, which ensures functional reliability. These components are usually integrated to create specific applications using programming or scripting tools supported by the framework, such as C, C++, Java, etc. For open-source frameworks, the source codes provided enable developers to customize the same for their own features and functions.

The emergence of smart clients together with the service-oriented paradigm has redefined the concept of flexibility, upgradeability, and portability. Applications will no longer provide value on their own, but as agents of service delivery. The smart client concept enables software developers to change features and functions as frequent as possible, depending on the type of service that it delivers. All these are transparent to the users, who will only realize the differences during his/her next login.
As a start, the basic framework must tackle the areas listed below in order to deliver maximum efficiency, promote reuse and get the best return on investment [11].

- Simple to understand and implement
- Is well documented with practical examples on how to use the framework
- Be extensible, providing room for improved add-ons to the framework
- Should be cross-platform and reusable in most environments.

In addition to the areas listed above, the web-based and network-dependent nature of a smart client requires that a framework for developing smart client applications must possess at least the following components:

- Reliable network services
- Transaction security services
- Web-based graphical user interface
- Support for any type of browsers by providing plug-ins
- Support for any type of operating systems – conventional and mobile
- Support for popular scripting languages such as Java, Perl, Python, etc.
- Independent storage facility – local and network-based

However, out of the components mentioned above, areas which the smart client development framework must emphasize on are networking reliability and security. Having these important aspects in place will further enable smart clients to be deployed for any type of applications and eventually becoming dominant on desktops, laptops and mobile devices.

V. APPROACH FOR DEVELOPING A FRAMEWORK FOR SMART CLIENT DEVELOPMENT

Developing a framework is the product of a long and evolutionary process. It doesn’t materialize over night as a framework is developed by observing the best of past software development practices and integrating those best practices into a singular framework. Since a framework needs to be a generic in nature, it is supposed to generalize good practices from existing systems as well as leveraging knowledge and research done on existing frameworks [4].

To develop a framework, an underlying methodology needs to be adopted first. This will act as the guiding principle by which the entire project should be executed in terms of project management to the actual implementation of the application. Once that is identified, the next step would be to identify existing CASE tools which are suitable for the environment to execute a smart client development project effectively and efficiently.

VI. IDENTIFYING A METHODOLOGY

The term methodology is often vague and is seldom fully understood. However, in loose terms a methodology is defined as a codified set of recommended practices, sometimes accompanied by training materials, formal educational programs, worksheets, and diagramming tools. This is very different from a method, which is just a series of steps used to build software [3].

Methodology addresses the following areas:

- What are a project’s stages and the tasks to be executed at each stage?
- What are the inputs, outputs and constraints that impact each stage?
- How should the project be monitored and controlled?
- What is the philosophy that drives the flow of the project?

Rather than haphazardly developing software, a good methodology will bring about many benefits. Among some of these benefits include: [9]

- Well defined projects with clear goals have a better chance of succeeding thus protecting the management’s investment and increasing staff productivity because of less rework efforts and healthy management of resources.
- All parties involved in the project are clear on its expectations.
- Quality is emphasized with the implementation of quality assurance reviews and audits which give management an idea of the quality of work being done.
- Surprises such as cost overruns, scope changes, late implementation, and other risks can be minimized with the availability of project status reports.

To select a methodology for smart client development the following need to be considered:

- Must accommodate dynamic business requirements.
- The time it takes from getting a requirement to actually delivering a working prototype or product must be short.
- All phases of the development process must closely involve customers to solicit prompt feedback and enable action to be taken.

As a result, this research paper has identified two existing software development methodologies that provide adequate support to efficiently carry out the development of smart clients.

VII. RAPID APPLICATION DEVELOPMENT (RAD)

Rapid application development (RAD) takes into account the changing business process requirements during the development lifecycle. It is also well suited to fast-paced development environment that is very customer-focused and increasingly global in nature [3].
This methodology incorporates the usage of:

- 4th Generation Tools to enable quick delivery
- An iterative model of systems development which allows backtracking in the light of changing requirements.
- Evolutionary prototypes that advocate that a working model is worth a thousand pictures.
- A very high level of user involvement in the development process to aid in communications and to encourage feelings of commitment and ownership.
- The empowerment of highly skilled, multi-disciplinary teams consisting of users, analysts and technical specialists.

VIII. AGILE METHODOLOGY/MODELING

Another suitable methodology for smart client development would be the agile methodology that emerged out of a set of light-weight software development methodologies. The core of this methodology proposes that processes be focused on:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan.

As such, we can quickly observe that agile development is highly suitable for small teams when facing unpredictable or unchanging requirements. It is also time-bound in nature and provides the framework to release software incrementally. Its characteristics allow for methodologies to be adapted to the environment rather than follow rigid static rules thus making it very appealing as not all software development environments are similar [10].

IX. CASE TOOLS

CASE stands for Computer Aided Software Engineering and it is defined as tools that provide leverage at any point in the software development cycle. The principle behind CASE is that when implementing software, its best when design is separated from implementation.

Below is a list of advantages that CASE tools bring to development team [5]

- Helps to keep an accurate description of a project’s specifications
- Helps to associate design specifications to actual code implementation so that possible impact to the design can be reflected quickly in the code implementation. This indirectly saves time and effort.
- The graphical nature of most CASE tools make it easier for developers and non-developers alike to produce solid and clear design diagrams that reflect requirements to be translated into code.
- Reduces future surprises by enabling thorough analysis to be done up front.
CASE tools began as pencil and paper driven design but since then have evolved into full fledged tools that are extremely user-friendly, based on solid software development methodologies and are highly integrated with one another.

Below is list of tools that target different phases of software development.

- **Data Modelling Tools** – used to help database designers identify the best architecture to enable data sharing between applications by modelling relationships between entities.
- **Analysis and Design Specification Tools** – used to model an application in a structured manner to capture data flows.
- **User Interface Prototyping Tools** – facilitates prototyping of user interfaces by evaluating user feedback.
- **Code Generation Tools** – enables design specifications to be translated directly into executable code.

To select the best fit of CASE tools for smart client development, we can take the approach of evaluating two environments.

- **Open Source Environment** - This environment has no one dominant vendor that determines the CASE environment. Instead a varied list of companies both big and small provide CASE tools that can be integrated together for software development. An example of this is the ECLIPSE framework that is provided free by IBM.
- **Single Vendor Environment** - This environment is dominated by a single vendor or a few big vendors that collaborate together via a technological alliance. An example of this is the Microsoft® Team System 2005.

In September 2005, Microsoft launched a new development suite called Microsoft® Visual Studio Team System. The release of this tool was quite historic as this was Microsoft’s first attempt at providing an integrated environment with CASE tools to accommodate the daily tasks all members in a development team from a project manager to system analysts, developers and testers.

From the start of the design phase, the tool provides modeling tools so that Software architects can not only draw pretty diagrams but actually capture design that can later be translated into production code. This also allows designers to work with one another and ensure that all aspects of the design are compatible with one another. The design capabilities are not only limited to designers but is extended to the deployment team who can logically test the environment even before a single piece of code is built.

From a software developer’s perspective, the .NET framework also has features like light-weight mobile controls for smart client development. Its compact framework also provides adequate support for development of mobile applications. In addition to that Microsoft is also touting their ubiquitous Microsoft Office as the presentation layer for smart clients. All the development work can also be done easily via one integrated tool known as Visual Studio .NET.

To facilitate testing and code review, the tool also incorporates the use of FxCOP a time-tested tool to quickly analyze code and provide suggestions to improve on it. Unit tests are also a new feature enabling test-driven coding to be accomplished.

At the base of this integrated environment is the Team Foundation Server that becomes the repository for all project related work.

**XI. CONCLUSION**

Organizations with an increasingly mobile workforce should seriously consider developing smart clients as a potential
alternative to maximize the benefits of the feature-rich thick client and the easy-to-deploy thin client.

However to do so successfully, would require a robust framework that can accommodate dynamic requirements, facilitates high user involvement and enables the development team to quickly produce a working product. An essential part of this framework is recommended methodologies such as RAD or Agile methodologies that are suitable for fast-paced environments. CASE tools are the critical other half of this framework that will enable efficient execution of smart client development. Pursuing smart client development without adopting a framework to support the environment would be too risky to enjoy the benefits of a smart client.

REFERENCES


