



Considerations for Cost-Effective RIA and SaaS Development and Deployment

September 2008

White Paper

By Avigdor Luttinger, VP Corporate Strategy

Magic Software is a trademark of Magic Software Enterprises Ltd. All other product and company names mentioned herein are for identification purposes only and are the property of, and may be trademarks of, their respective owners.

Magic Software Enterprises has made every effort to ensure that the information contained in this document is accurate; however, there are no representations or warranties regarding this information, including warranties of merchantability or fitness for a particular purpose. Magic Software Enterprises assumes no responsibility for errors or omissions that may occur in this document. The information in this document is subject to change without prior notice and does not represent a commitment by Magic Software Enterprises or its representatives.

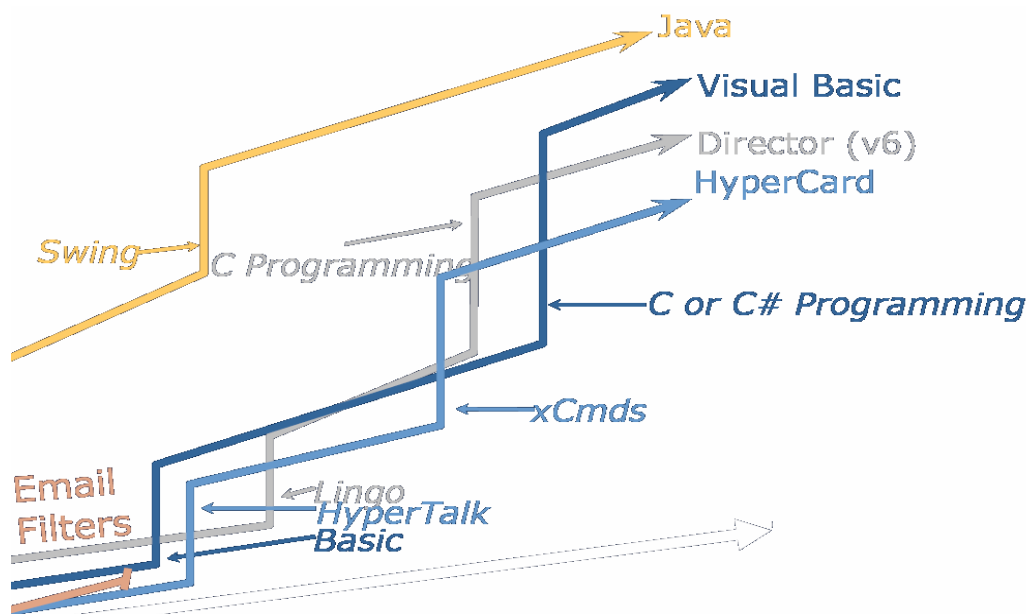
Contents

Background.....	3
Software Applications Today	3
RIA and SaaS – Explaining the Need.....	4
From Dumb to Fat to Thin to Fit – The Evolutionary Spiral.....	4
Enter Rich Internet Applications (RIA).....	4
Advantages of Rich Internet Applications – Summary	5
Challenges to Cost-Effective RIA and SaaS Adoption.....	5
On-demand vs. On-premise ‘Operation’.....	5
The Application Portfolio Mix.....	6
Usability Issues	7
Development Hurdles	8
Time to Choose.....	8
Cost-Effective RIA and SaaS Development and Deployment	9
Magic Software’s uniPaaS Features a Unitary Development Paradigm	9
Saving Money and Resources in SaaS Deployment.....	10
Conclusion	10
About Magic Software Enterprises.....	11

Background

Software Applications Today

Leading analysts predict that 60% of all software development projects will involve Rich Internet Applications (RIA) within three years. Business leaders should pay attention. Clearly it's no surprise that we're using software applications more than ever before. As our demand for new capabilities and functions grows, our software becomes more sophisticated as developers strive to provide an adequate response to our business needs. It's only natural then that the rate of application evolution will place an ever-larger burden on the shoulders of software producers, vendors and even consumers.



Difficulty of Use increases in line with program complexity and sophistication

Source: Brad Myers, Ph.D., School of Computer Science, Carnegie Mellon University

So how do we survive and perhaps even keep one step ahead in an environment of the ever-complex human – machine interface? This white paper shows how business application platforms have evolved and in particular the reasoning behind the rise of Rich Internet Applications (RIA) as part of the Software-as-a-Service (SaaS) phenomenon. The paper highlights the challenges of this new paradigm, and offers a strategy for achieving cost-effective and efficient RIA and SaaS development and deployment based upon a SaaS Enabled Application Platform (SEAP) approach.

RIA and SaaS – Explaining the Need

From Dumb to Fat to Thin to Fit – The Evolutionary Spiral

When it comes to the evolution of IT, it's popular to talk about circles of evolution within the industry, which periodically moves back and forth between models as times and demands change. But it would be more accurate to refer to this phenomenon as a spiral.

The spiral starts with the original mainframes and terminals of the 1960s and '70s era, which consisted of 'Dumb' Clients designed simply to display and input information.

The PC revolution changed all that with the introduction of the personal CPU. This created, for want of a better term, a 'democratic' information technology system, which moved the center of operations and processing onto every user desktop.

This evolved into networking systems – still with very little functionality and with most of the logic sitting on the Client side. The advent of desktop processing created a significant euphoria as software suddenly became much more intelligent and sophisticated. Even the simple spell-checker that we now take for granted would have once required much more computing capacity than was ever available, until the desktop Client came along.

However, once the euphoria began to wear off, people realized that beyond the comfort, there was a huge price to pay for maintaining the so-called 'Fat' Client. A typical organization of 1000 desktops would require each machine to be installed and maintained, with regular update deployments, and a substantial capital expenditure.

With the internet revolution, things changed again. Centralized computing and the advent of portals mean that the industry has almost returned to the principle of the 'Dumb' Client – where all major computation occurs at the Server – and where the 'Thin' Client's name comes from the fact that it acts basically as a window. Having travelled full circle, today we are back at a similar topology that we started with over forty years ago, but with greater performance and functionality.

However, while the cost of operation for the 'Thin' Client remains conveniently low, the reality is that it's limited in its operational scope, lacking the richness users have come to expect from 'Fat' Client applications.

Enter Rich Internet Applications (RIA)

The inherent limitations of the 'Thin' Client is today leading the industry into a new channel and mode of deployment – Rich Internet Applications and the advent of the 'Fit Client' – that bring the best of both worlds – both the desktop richness of the 'Fat' Client with the low operating cost of the 'Thin' Client.

Rich Internet Applications are fully interactive (desktop-style) business applications that are installed at a single location (the Server) and are accessible via the internet (the Client).

Advantages of Rich Internet Applications - Summary

There are considerable advantages to employing RIA applications:

- RIAs can be accessed from mobile and remote devices, significantly increasing their usage potential
- RIAs feature desktop grade interactive richness and functionality that give a much improved user experience compared to Web browser applications
- RIAs are based upon a centralized architecture that eliminates the need to maintain every new user – thus significantly lowering the cost of ownership and operation
- RIAs are SaaS enablers. Because of their utility approach, SaaS applications are quickly and easily assimilated by customers
- RIAs feature multi-tier architecture, allowing businesses to better secure the sensitive aspects of their application, and assures high scalability

Challenges to Cost-Effective RIA and SaaS Adoption

On-demand vs. On-premise 'Operation'

With the increasing complexity and richness of enterprise applications, a new role has evolved – that of Software Operators. Applications today are being 'operated' even more than they are 'produced' or even 'consumed'. Applications can therefore be classified according to their style of operation:

1. **On-premise** (traditional mainframe, Client/Server, based upon dedicated resources, in-house infrastructure and perpetual license-based pricing);
2. **Outsourced** (intranet based Clients and third party operated and managed infrastructure);
3. **Application Service Providers (ASPs)** (Web-based Clients, Cloud infrastructure, subscription based payment); and,
4. **Software-as-a-Service (SaaS)** where Rich Internet Applications (RIAs) are based upon a multi-tenant, subscription-based pricing model.

Enterprises are faced with the question of choosing the most effective and economical mode of software operation, and often are looking at multiple operating modes.

Among these, SaaS is probably the “hottest” operating mode at present. But that does not mean that its advantage over on-premise applications is completely cut and dry. From the consumer point of view, on-premise means a substantial capital expense where as on-demand means an operating expense. This makes the decision process in acquiring and paying for the system substantially different.

From an operational point of view, businesses must distinguish between levels of control, where on-premise offers high control and SaaS provides only abstracted control, where the application is simply ‘consumed’. The SaaS model therefore depends on a service level agreement to ensure that the software is made available according to the demands of the users.

In terms of adaptability, on-premise allows for customization – but at a price. However, for SaaS, where the software is not owned outright, significant customization becomes much more difficult.

The Application Portfolio Mix

Because of these trade-offs, it’s important to remember that the industry is not simply moving through revolutions. Consumers and businesses continue to use all modes of deployment including mainframe/terminal, Client/Server, and Web, and will continue to do so for years to come. The challenge now is how to make these heterogeneous application portfolios less complex for independent software vendors to manage or for any other form of software ownership.

Because of the inherent trade-off between on-demand and on-site, most organizations will operate a mix. With such a heterogeneous application portfolio, businesses need to service different (though sometimes overlapping) constituencies and will require varying skills and competencies to operate them all. For example, a typical business may need to keep up legacy applications, such as COBOL, along with a host of in-house and possibly packaged applications because they contain critical features for so-called ‘power-users’. Along with these may be the more modern Web applications that require Web developers. So what happens when an organization now wants to introduce Rich Internet Applications?

Application Type	Usage	Skills
Legacy Applications	Critical processes, power users	Retired?
In-house Client/Server Applications	Important processes, sometimes deployed Remotely (Citrix or similar)	Programmers (C, Java, VB,...) Business Analysts
Packaged Applications	Critical processes, power users	Application Specialists
Web Applications	Occasional usage	Web Developers
Rich Internet Applications	SaaS (CRM,...) and experimentation	New language (Client Side), Server Side, Business Side, ...

How many organizations can afford the multitude of diverse skills and persons needed to assure the optimal operation of all their applications?

It must be taken into account that RIA by itself constitutes a highly complex form of application. So how can an independent software vendor or system integrator maintain and develop all of these applications types – along with the extremely varied skill-sets that must sit within the organization to keep all of these systems running?

What organizations really need is the ability to smoothly switch between deployment modes and technologies, based on situational requirements.

It is not yet clear which deployment modes (SaaS or on-premise) are more cost effective, considering the full lifetime of an enterprise application (7-10 years). What is clear is that it's easier to experiment with a new enterprise application that is also offered in SaaS mode. Recent Forrester research shows that 58% of organizations that are not interested in SaaS listed “total cost concerns” as a reason for their lack of interest, second only to integration issues (65%).
Source: Forrester's Enterprise And SMB Software Survey, North America And Europe, Q3 2007.

Usability Issues

Essentially, there are four main alternatives for RIA development today. Terminal-based Clients such as Citrix; browser-based RIA such as Ajax; browser-free RIA such as Microsoft Silverlight and Adobe Flex; and Metadata-driven Application Platforms such as Magic Software's uniPaaS.

Terminal Clients such as Citrix deliver a full desktop application remotely. Due to poorer scalability, it's relatively costly as far as infrastructure is concerned, and it is not aware of the Client device on which it executes. As a result, additional challenges arise with the usability of rich application screens and local device features.

Taking it a step further, RIAs of the dynamic HTML and Ajax types are attempts to insert dynamic data elements into the Web page – including partial page refreshes and more field-specific interactions. However, it must be remembered that the browser is not specifically designed for this level of usability, and is not the ideal environment for 'Rich' operations. Ajax-type applications are particularly problematic for the 'power-users' such as those operating SAP or Salesforce.com applications with screens displaying dozens of different data fields at a time. Ajax implementation makes such screens extremely heavy and slow and makes development difficult. As usage requirements increase it makes more sense to move away from Browser Clients and more towards the so-called '**Fit**' Client.

The invention of the 'Fit' Client therefore attempts to circumvent the limitations of the browser environment through the implementation of an internet Client outside the framework of the browser. Browser-less Clients such as Microsoft Silverlight and Adobe Air follow the 'Fit'

Client principle. They are not page oriented and in this respect they behave very similar to a desktop Client. They are still lightweight, do not work independently and are more of an extension of the Server's capability – providing much better partitioning between the Client and Server sides. The result is that just the right amount of processing capabilities is discharged from the Server to the Client in order to enable complex interactive features, while maintaining all back-end operations (that are not interactive) at the Server.

However, both browser-based 'Rich' Clients and browser-less 'Fit' Clients essentially follow the same principles – and require the implicit planning and programming of separate Server and Client sides, leading to a number of challenges as we will now see.

Development Hurdles

Rich Internet Applications today represent one of the most challenging development processes yet to be seen. Firstly, Client-side development uses different technology than Server-side development. For example, the Adobe Flex language may be used for developing a typical Client. This Client would then consume Server-side services developed with C# or Java. Then there is the inter-lying communication layer that has its own particularities. So, a typical development effort requires building and maintaining a number of different teams to work on the different aspects of the application. As a result, the design, planning and management of the project become more risky, more complex, and of course more expensive. As with any system, where there are more moving parts, there is an increased chance of breakdowns.

An added complication when dealing with a RIA Client is that you have to explicitly program responses into the application – since the Client is now an independently functioning entity that must be managed on a per-field level.

Therefore, developing and deploying a RIA (or SaaS) solution requires businesses to first buy, and then integrate multiple platforms along with diverse Server and Client paradigms.

Time to Choose

The discussion above is reflected in a growing consensus among industry analysts that traditional technology platforms, such as standards-based application servers, are sufficient for simple RIA and SaaS use, but advanced and broad-based offerings will rely on specialized or extended SaaS-Enabled Application Platforms (SEAP).

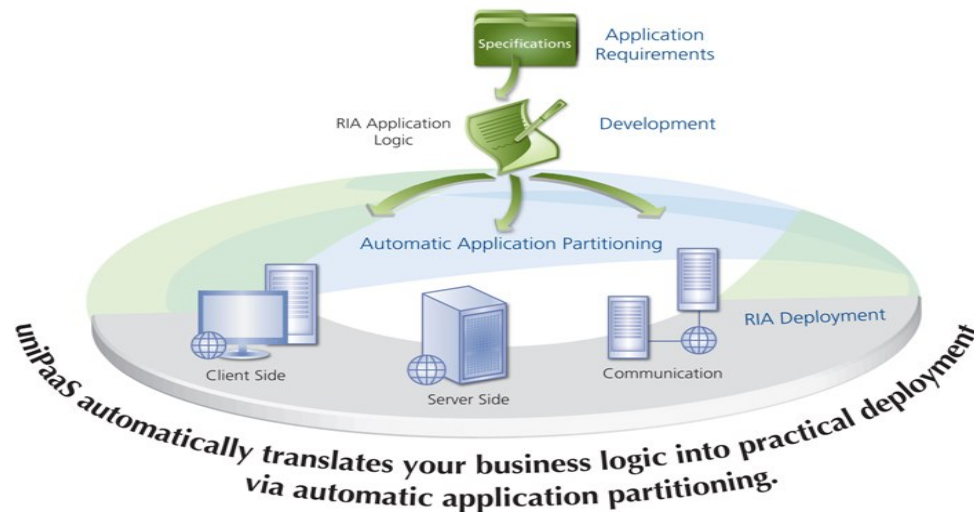
One of the first such SEAPs is Magic Software's uniPaaS. It provides much more intensive and extensive abstraction and significantly reduces the complexity of development and deployment over the entire application spectrum.

Cost-Effective RIA and SaaS Development and Deployment

Magic Software's uniPaaS Features a Unitary Development Paradigm

uniPaaS is a new breed of SaaS-Enabled Application Platforms (SEAP) capable of delivering RIA and SaaS through a unique, unified development paradigm that incorporates all aspects of the development and deployment process. uniPaaS can manage the setting and controlling of the Client-side and Server-side logic, in addition to the communication between the Client and the Server, and the consumption and manipulation of back-end services – all within a single solution and paradigm.

uniPaaS provides all the benefits of Rich Internet Applications, but unlike the browser-based approach of Ajax-type solutions, is not dependent on the browser and its various usability problems. Also, unlike the browser-free RIA solutions that still require in-built and separate Client and Server partitioning, uniPaaS handles all Client/Server partitioning automatically, thus greatly simplifying the development process and significantly reducing costs.



As an application platform, uniPaaS therefore supports the entire application delivery spectrum – desktop, Client/Server, Web, RIA and SaaS, with a common application source. To illustrate, a typical OS 400 application can now continue to be deployed on the old terminals – but they now become accessible to Fat Clients, Web Clients, Rich Clients and Mobile Clients as well.

Saving Money and Resources in SaaS Deployment

The ability to deliver service to remote devices is the first prerequisite for the Software-as-a-Service model; and one of the main challenges within this model is the issue of multi-tenancy. A key difference between SaaS and ASPs is elasticity – the ability to allocate resources dynamically to those who need them. This requires that each consumer, or ‘tenant’ using the software have a minimal static footprint in terms of resources dedicated to them, but with near-unlimited scalability. In general, SaaS achieves this by using a single instance of the Server platform, which is tenant-aware, while providing full-tenant independence. In most cases this requires applications to be developed in a dedicated ‘tenant-aware’ mode or developed from scratch on a tenant-aware or multi-tenant platform.

This demands more from the development team and raises costs for the software development community. If we take the example of a successful human resources application designed to run on a Java application Server, if the ISV also wants to start selling the application as SaaS, they need to redevelop it on a new platform – while at the same time continuing to service and maintain the old one. Both platforms will have to run in parallel for the foreseeable future, since not all customers will want to move to the SaaS model.

If developed with uniPaaS, the abstraction features of the platform along with the use of a single development paradigm means that the same application can be deployed back and forth in both multi-tenant and single-tenant modes. Even applications initially developed as single-tenant can now be deployed as multi-tenant without re-writing; thus saving the developer a huge amount of money in addition to development resources.

An additional challenge of the SaaS model is the question of fine-grained metrics – how to charge for the usage of the software. The overall creativity of the SaaS model is constrained by what can actually be measured in terms of usage. uniPaaS offers an unprecedented choice of measurement down to the individual user level, providing for the deployment of every type of rich business model.

Conclusion

Today we are witnessing a new change in the way the industry thinks about software applications. The increase in SaaS application development and consumption is driving up demand for Rich Internet Applications and a new set of platform technologies built specifically to develop and deliver them.

Software vendors, both large and small, are now considering how best to adapt to the new paradigms of the RIA and SaaS markets, while a large number of developers across the world are moving to RIA and SaaS application development. As with the evolution of technology

platforms in the past, we are beginning an exciting time for the industry as a host of mega-vendors and start-ups engage in the emerging platform wars.

Although still early, we can already expect development and delivery platform vendors to form multiple alliances – and to obtain integrated solutions from vendors. And with the total opportunity in SaaS platforms predicted to grow between \$5b and \$11b by 2012 (**Emerging platform wars in enterprise software, McKinsey & Company, Inc. 2008**), vendors, operators and consumers would be wise to consider the challenges of cost-effectively managing the transition. It is also important to keep in mind the pattern of spiral evolution – that there are always exceptions in the software development industry and few, if any, clear-cut transitions from one model to another.

uniPaaS is designed to respond to all of the challenges outlined above and is the first application platform to give businesses the power to both operate and manage an entire application portfolio from legacy to RIA and SaaS.

In particular, uniPaaS simplifies the development process by abstracting technical and architectural complexities and allowing the developer to focus resources on optimizing business logic. It reduces the ownership and maintenance costs by rendering the same application suitable for multiple Client modes, from stand-alone desktop to service-based multi-tenancy. And it incorporates a SOA backbone, with business integration and process management technology that enables its users to span the entire IT spectrum – **all from a single platform**.

For this reason, a proven SaaS-enabled application platform based around a single development paradigm should be vigorously considered by any business looking to step into the promising future of RIA and SaaS while desiring to maintain and leverage the value within their current software investments.

About Magic Software Enterprises

Magic Software Enterprises Ltd. (NASDAQ: MGIC) is a leading provider of **multiple-mode application platform solutions** – including Full Client, Rich Internet Applications (RIA) and Software-as-a-Service (SaaS) modes – **and business and process integration solutions**. Magic Software has offices in 10 countries and a presence in over 50, as well as a global network of ISVs, system integrators, value-added distributors and resellers, and consulting and OEM partners. The company's award-winning code-free solutions give partners and customers the power to leverage existing IT resources, enhance business agility and focus on core business priorities. Magic Software's technological approach, product roadmap and corporate strategy are recognized by leading industry analysts. Magic Software has partnerships with global IT leaders including SAP AG, salesforce.com, IBM and Oracle. For more information about Magic Software Enterprises and its products and services, visit www.magicsoftware.com. Magic Software is a subsidiary of Formula Systems in the Emblaze Group of companies.