

# RapiDeploy Intelligent Architecture™

---

July 2005



**Provision, Manage, Profit: Be Empowered.**

© 2005 Fine Point Technologies, Inc. All rights reserved.

The information contained in this document represents the current view of Fine Point Technologies, Inc. on the issues discussed as of the date of publication. Because Fine Point Technologies must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Fine Point Technologies, and Fine Point cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. Fine Point Technologies Makes No Warranties, Express or Implied, In This Document.

Microsoft, Outlook, Windows, and Windows NT are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Fine Point Technologies, Inc. • 139 Centre Street • New York, NY 10013 • USA

## PREFACE

---

One of my favorite all-time phrases is, “That change only requires a change to two lines of source code. It would take me no more than 5 minutes to do.” I have heard this many, many times from developers as well as customers. What are not taken into consideration are the diverse, critical implications of modifying source code.

I have prepared this preface as a personal note to everyone reading this. In the past seven years of developing software solutions for the service provider market, I have seen many companies come to market and make the same mistakes time and time again. Those mistakes are the key differences that differentiate Fine Point from any other company on the market. These differentiators are both technical as well as a philosophical, on the concept of software design called Direct Plug-in Technology™ and RapiDeploy™ Intelligent Architecture; two concepts which are the underlying principles upon which all Fine Point Technologies’ solutions are based. It is these two principals that have enabled our company to prosper even during the times of the “dot.com collapse”, “Telco Meltdown” and “9/11” both of which occurred over the course of two very difficult years from 2000-2002. It is also these concepts that enable our successful company to continue to grow at a steady pace and win over our competition.

Please note that this document contains confidential information about our Direct Plug-in Technology as well RapiDeploy™ Intelligent Architecture, and our general philosophy of software design, therefore cannot be shared without explicit permission from Fine Point Technologies.

## INTRODUCTION

---

In the internet service provider market, no two service providers are alike, nor are the feature sets that they demand ever the same. In addition, brand recognition is key for the development of their businesses. For this reason, software solutions for this market demand a high level of flexibility and customizability and must be designed in a manner to avoid the many pitfalls inherent to compiled software. The term RapiDeploy™ Intelligent Architecture defines not a technology, but rather a philosophy for highly customizable software. This philosophy delivers ease of brandability and flexibility, which is critical to the viability of any solution in the market.

### Issues with Compiled Source Code

Any traditional software vendor will take the approach of developing, managing and modifying their software product from the source code. This is traditional, expected, and the industry standard. Therefore, when changes need to be made, the industry norm is to assign developers to make the required modifications. However, this approach does not work well for customized solutions. Whenever a product source code is modified there are a number of implications.

### Source Code Flux

Usually, a product source code is in constant evolution; it is always being worked on by the development team. In most cases a software company will be working on the feature set of the next release while the existing release is being finalized in Quality Assurance (QA). While the source is being worked on, it is in a “state of flux” and can not be compiled due to the many changes that have been made to the code. What happens then is that any urgent features or changes that may come up will have to wait until the next version when all of the changes are complete and tested. Even if proper source code management (SCM) is utilized, which enables roll-backs to previous versions, the development environments often need to be changed to support the older version of the source code. This can cause a tremendous barrier when market-time sensitive changes are needed on an urgent basis. If a change needs to be made to the product in response to an unexpected market condition, it can be difficult to meet the required time frame because the product code can not be recompiled.

### Baseline Quality Assurance

Whenever the source code to a commercial-grade software product is modified, it is imperative that the entire product be tested to ensure that no other aspects of the code were affected by the code modifications that were made. Even the smallest changes in source code can affect the product. As a matter of fact, slight variations in the development environment can result in problems in the product, even when source code changes aren't made! This means every feature and every capability of the product must be re-tested, which is not only time consuming, but expensive and creates a substantial “engineering hurdle” for making modifications to adapt to a rapidly changing market. What must be kept in mind is that even a small change to two lines of source code requires a full Q/A cycle which includes, but is not limited to:

- Testing on every version of the Microsoft OS (95a, 95b, 95c, 98, 98a, ME, 2000, XP, Server 2003, in addition to all service pack variations)
- Testing every feature
- Review of all text

## Feature Changes

Change and evolution of any product is essential to the success of that product. As new features are added or as new technologies need to be supported, the traditional approach requires changes to the source code. Feature changes may even include the removal of features from a product in order to customize it for a particular purpose. Once this occurs, all the issues of modifying source code once again become relevant. The most significant issue with feature changes is, how does this change affect the balance of the source code? What other features may be affected, even by the simplest change? Only a complete Quality Assurance cycle of the entire product will bring issues to the surface and ensure its' quality.

## *Feature Selection*

The other issue with traditional development is that features become a permanent part of the product. For example, in order to remove a particular feature, the product may need to be recompiled. In businesses such as Internet service, a product's features may produce either positive or negative results. In some cases it may be necessary to quickly remove a feature from a product in order to reduce support calls. When these features are permanent pieces of the source code, it becomes more difficult to quickly make such changes.

## Localization

Localization, or the process of translating both the words and the "intent" of the offering, is perhaps one of the most fundamental challenges facing any traditional software company. While Western languages (such as English to Spanish) translations typically are manageable, Eastern languages can be a major hurdle. Entering and managing such text, (referred to as "double-byte characters") requires special equipment and operators that know how to utilize that equipment. Moreover, in a customized solution, true "localization" may require more than the translating of text, but rather the modification of content to best suit the purpose in which it is to be used.

In order for products to be localized or translated, even when properly planned though traditional software practices, recompiling the source code is always required. This means that both translators and developers must be involved in the process. In some cases, specially trained (and more expensive) translators are needed to translate text that is present within the source code without damaging the source code.

## Branding

One of the most important features in a customized solution is branding, which extends beyond logos to include the entire look, feel and user experience. A user interface (UI) can be one of the most difficult iterative challenges. In situations where features are added or removed from the product, the user interface needs to be modified to accommodate those changes. Even if a features set is not customized, the user interface may need to be modified to meet specific branding requirements. Simple changes to a user interface can often lead to many unforeseen issues. Such as:

- How the new UI will fit with different screen sizes (the notorious 640 x 480)?
- What happens in the new UI when the user selects "Large Fonts"?
- How the new UI interacts with changes in the OS's supported color palette?

In most cases, under traditional software development practices, these issues aren't even given consideration. An illustrative result is the exercise of a large-font OS causing text to flow off the window of the application and becoming unreadable.

## **Dependence on a Third-Party Vendor**

Companies that utilize software products from traditional third-party vendors are dependant on that vendor for customizations. Since the software product vendor owns the source code, only they can make modification to the source code.. If a situation arises where the vendor (for whatever reason) can not comply with the request for changes, the customer's business can be greatly affected. Even in situations where the source code is held in escrow, it would take the customer a considerable amount of time to procure a development team, and for that development team to become familiar enough with the source code to make the required changes.

## **Duplication of Source Code**

As the source code of a product needs to be customized for a particular customer or modified for a specific purpose, a duplicate must be made. The duplicate code is retained as the "unmodified" or "shell" version of the product, and the new source become the customized product. This is done quite often in the internet service provider industry. Source code is duplicated and then customized time and time again. When new a feature later needs to be added it is either:

- Added many times to each of the source code set, effectively doubling the engineering requirements for each time the code is changed, or
- The feature is only included in the baseline product, therefore full re-customization must be done, typically at the cost to the service provider.

## **Engineering Resource**

As source code is modified, it takes a substantial amount of engineering resources to complete the changes. Even if source code changes are made by one single developer, it still requires the entire Quality Assurance team to test the product and ensure that quality is maintained. This means that software source code changes, even the simplest, are expensive.

## **Time to Market**

It is clear that software source code changes take time. This means that time-to-market for new features of a software product also take time. A typical software company can reliably update their source code and product from that source code no more than two times per year, while the generally accepted practice is one major update annually.

## THE FINE POINT TECHNOLOGIES DIFFERENCE

---

Through patented technologies, all of Fine Point Technologies' software is developed to overcome the implications of traditional software development. To summarize the key differentiators:

- **Direct Plug-in Technology** – A software design methodology through which product features can be added or removed from the product without recompiling the source code.
- **RapiDeploy™ Intelligent Architecture** – A software design philosophy through which software can be easily localized, branded, & customized without recompiling source code.

These technologies enable the following benefits for the service provider, and ultimately, their subscribers.

### Minimized Source Code Flux

With Fine Point Technologies' approach, since changes and evolutions can be dynamically added to the core product without requiring significant changes to the source code, the source code is not subjected to the "state of flux" issues associated with traditional software design. Therefore, products that leverage Direct Plug-in Technology and RapiDeploy Intelligent Architecture will be better positioned to quickly adapt to market changes and demands.

### Minimized Quality Assurance Requirements

Products that leverage Direct Plug-in Technology and RapiDeploy Intelligent Architecture can be tested in QA and a baseline can be established for every release version, as is traditionally done in the first release of development. Using our approach, various changes that are implemented via these technologies do *not* require that the entire product be retested. Therefore the changes can simply be tested against the baseline and only the difference (the changes) need to be tested. This drastically reduces the amount of engineering required to perform customization, branding and localization.

### Easily Add or Remove Features

Technology is ever-evolving. Trying to forecast or predict which technologies will be relevant in two or three years, while developing a product, is a difficult task. In most cases, new technologies may come to market that require integration. There is no simple way to predict this. For this reason, with Direct Plug-in Technology, a core product can be well positioned to incorporate third-party technologies in a short amount of time. This enables the core product to easily include these features without the need to recompile and QA the core product's source code.

### Simple Feature Inclusion/Removal

Through Direct Plug-in Technology, features and functionality of the core product can be easily removed without recompiling the source code.

### Fast Localization

Using the RapiDeploy Intelligent Architecture philosophy, localization and text modifications are easy. All text can be easily modified via an external text file. Translating text to a new language takes no more than electronically sending the external text file to any commercial translation service and replacing the

text file on the final product CD-ROM. All text will be dynamically loaded at runtime. This approach also makes for easy changes to the text should the wording not meet the needs of the specific situation.

## **Improved Branding**

RapiDeploy Intelligent Architecture enables applications to be easily and quickly branded. All GUI elements are stored in external bitmap files, which can be easily modified by anyone with graphics software such as Photoshop.

## **Single Source Code Base Focused Evolution**

Software products that are developed with the RapiDeploy Intelligent Architecture utilize a single and common source code base. There are a number of advantages of this approach:

- ***Focused Evolution*** – The customized core product will include the common feature set and evolution contributions of all of the target audience. This means that as features are added as a result of one of our customer's request, all other customers can experience the benefit of the new feature as well.
- ***More Mature Source Code*** – Since the source code is common to all customers, the maturity of the source code is greater because the same code is in use in a much wider audience than just the one specific customer.

## **Independence of Third-Party Vendor**

Direct Plug-in Technology and RapiDeploy Intelligent Architecture enable the customer to be independent of a third-party vendor. Since modifications and customizations can be made without the source code, the customer can make changes with average technical knowledge without the need for source code. It is also common practice of Fine Point Technologies to grant the customer the licensed the right to modify the product via Direct Plug-in Technology and RapiDeploy Intelligent Architecture themselves.

## **Faster Time to Market**

Since no compiling is needed to customize a product developed with the RapiDeploy Intelligent Architecture philosophy, time to market is drastically reduced, directly benefiting the service provider.

## ABOUT DIRECT PLUG-IN TECHNOLOGY

---

Each of Fine Point Technologies' products incorporates some form of Direct Plug-in Technology. While the actual specifications of Direct Plug-in Technology may vary from product to product, the concept and philosophy is the same. That philosophy is that each implementation of Direct Plug-in Technology must conform to:

- **Extensible Functionality** – The products functionality can be extended via an external DLL or executable (“Direct Plug-in”). While a software compiler may be used to create the actual Direct Plug-in, no compiler or special tools (except for any supplied wizard applications) are required to include or configure the Direct Plug-in for inclusion within the core product.
- **General Architecture** – The Direct Plug-in must support RapiDeploy™ Intelligent Architecture.
- **Dynamic Enumeration** – The core product must be able to dynamically enumerate all installed plug-ins at runtime and potentially support an unlimited number of plug-ins. Each Plug-in can be added or removed from the core product easily through a simple configuration file of the core product.
- **Self-Contained** – All settings, data and text specific to the Direct Plug-in Technology are to be self-contained within the plug-in so that all of the settings, data and text will be added or removed with the plug-in.
- **Dynamic User Interface** – Any user interface elements that implement features, which are included via Direct Plug-in Technology, must be created dynamically at runtime by the core product so that the interface will dynamically modify itself at runtime based on the Direct Plug-in features that are installed. For example, a when a Direct Plug-in feature is implemented through a button in the user interface, the core product must dynamically generate the button and install the functionality. The design should accommodate for an unlimited number of Direct Plug-ins to be implemented via the GUI. If a hard limitation on the number of Direct Plug-ins is necessary, the core product must trap the condition of too many plug-ins and display a message to the end-user.
- **Third-Party Documentation** – The architecture of the Direct Plug-in Technology must be fully documented so that a third-party can easily create Direct Plug-in for the core product.

## ABOUT RAPIDEPLOY™ INTELLIGENT ARCHITECTURE

---

Each of Fine Point Technologies' products follows the philosophy of RapiDeploy Intelligent Architecture. The general rules of this philosophy are as follow:

- **Externalized Text** – All text that appears in the core product is contained in an external plain text file. This file is external to both the application as well as the installer. Any installer text should also be externalized as much as possible. The file format for this external text should be in a standard Windows Initialization File Format.
- **Externalized Interface** – The interface of the solution must be externalized. The external files must support Windows standard bitmap (BMP) file format.
- **External Configuration** – All logic, features and Direct Plug-in support must be configurable via an external text file. The external files must support Windows standard bitmap (BMP) file format. In cases where functionality or flow of the core product can be modified, the configuration file must be dynamic enough so that this functionality can be effectively “programmed” at runtime via the configuration file. An example of this would be workflow.
- **Administration Guide** – Any product that supports RapiDeploy Intelligent Architecture must have an accompanying “Administrator’s Guide” that provides detailed instructions on how to customize the solution via the RapiDeploy Intelligent Architecture.
- **Optional Administration Kit** – Ideally any product that supports RapiDeploy Intelligent Architecture will have an accompanying Administration Kit that enables the complete customization of the product via an easy to use software interface.
- **Intelligent Installer** – Any product that supports RapiDeploy Intelligent Architecture must have an installer that also is designed based on RapiDeploy Intelligent Architecture and manages the installation of the core product, Direct Plug-ins as well as all of the required RapiDeploy Intelligent Architecture components.

## **REGIONAL LOCATIONS WORLDWIDE**

---

### **United States (Corporate Headquarters)**

139 Centre Street, 6<sup>th</sup> Floor  
New York, NY 10013  
USA  
+1.212.962.7410  
info@finepoint.com

### **United Kingdom**

Globix House  
1 Olivers Yard  
London EC1 Y1HQ  
UK  
+ 44.2075.264818

### **France (Europe, Middle East, Africa)**

Les Algorithmes, Bât. Aristote A  
2000, Route des Lucioles, BP 29  
06901 Sophia Antipolis  
France  
+ 33.1707.18418