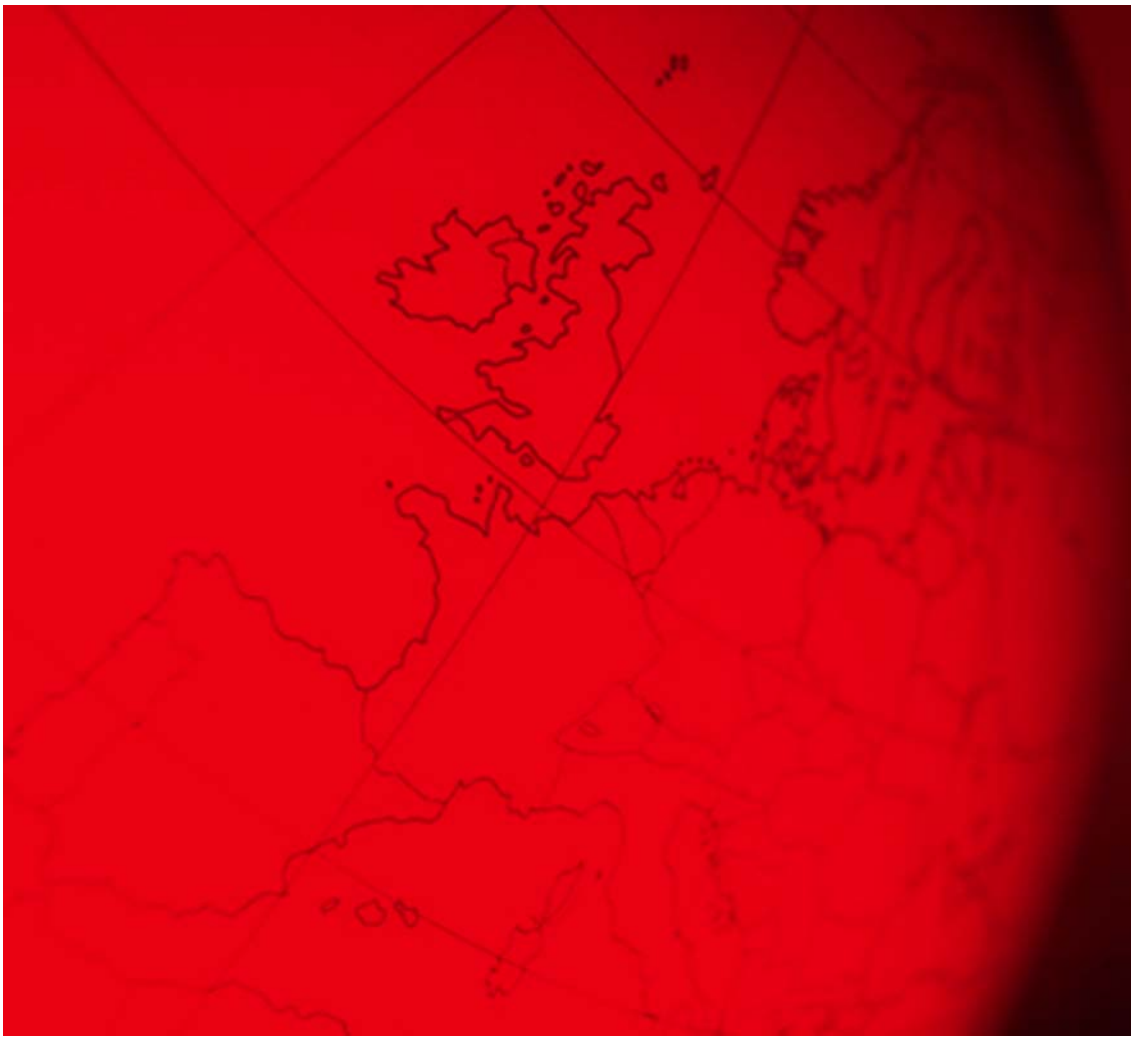
NUCLEUS RESEARCH

August 2009

Document **J61**

REPORT
MICROSOFT PATTERNS AND PRACTICES



Corporate Headquarters
Nucleus Research Inc.
100 State Street
Boston, MA 02109
Phone: +1 617.720.2000

Nucleus Research Inc.
NucleusResearch.com

TOPICS

Application Development & Integration

Enterprise Applications

THE BOTTOM LINE

Microsoft patterns and practices provides .NET developers with guidance, tools, libraries, and frameworks to help them build better applications. Development teams adopting patterns and practices achieved greater developer productivity and improved ongoing application management and application performance.

Microsoft patterns and practices helps software architects, developers, and their teams leverage the Microsoft platform to build better applications. Patterns and practices guidance, tools, libraries, and frameworks help to simplify the platform, provide guidance to common problems, and help development teams build loosely-coupled applications. Key components of patterns and practices include:

- The Enterprise Library is a collection of reusable software components that help developers with common development needs such as logging, validation, data access, and exception handling. They can be used "as is" or extended or modified by developers as needed.
- SharePoint Guidance helps architects and developers design and build SharePoint intranet applications.
- Composite Application Guidance for WPF (formerly code-named Prism) helps developers build and evolve modular Windows Presentation Foundation and Silverlight client applications. These applications typically include multiple screens and role-determined behavior and are designed to change over time based on business requirements.
- Web Service Software Factory are patterns, models, and written guidance integrated with Visual Studio 2008 that are designed to help developers build Web services.
- Microsoft ESB Guidance for BizTalk Server 2006 R2 is designed to help developers leverage the Microsoft BizTalk Server platform to build an enterprise service bus (ESB).

Each component includes written documentation, reusable source code, and in some cases, reference implementations. All are developed and released based on the input of internal Microsoft development teams and Microsoft partners and are provided free of charge for developers.

In 2009, new patterns and practices releases included:

- Enterprise Library 4.1 for .NET Framework 3.5 and Visual Studio 2008
- SharePoint Guidance
- Application Architecture Guide 2.0 community technology preview with design-level guidance for common types of applications and their key design characteristics
- Acceptance Testing Guide to help developers evaluate if applications meet users' needs and will be effectively adopted
- Composite Application Guidance for SPF and Silverlight with more guidance for building modular loosely coupled applications built on WFP and Silverlight
- Web security scenarios and implementation guidance.

Nucleus has followed the evolution of patterns and practices at Microsoft for some time. This report evaluates the key benefits architects and developers have experienced from adopting patterns and practices tools, guidance, and models to improve their application development practices and is based on Nucleus's analysis of the experience at many Microsoft partners and customers including Idea Blade, HIS, RDA, URS, Volvo, and AIC.

KEY BENEFIT AREAS

Nucleus found organizations adopting patterns and practices achieved benefits in a number of areas, including increased developer productivity, improved application management, and improved application performance.

Increased developer productivity

Patterns and practices guidance, tools, libraries, and frameworks help increase developer productivity throughout the development process, from application planning to coding to testing and debugging.

Accelerated application planning

Developers can use patterns and practices guidance and reference architectures to quickly determine the best structure for an application, reducing the time needed to conceptualize and actually begin to design the application:

- "*It saved us a lot of decision making time trying to find solutions for things. When a problem or technical issue came up, we could go back to the guidance in sample applications and see how they solved the problem. We were able to leverage that to know how to best solve the problem. SharePoint has a huge API and a lot of developers have to hook into information within lists to get their data. Patterns and practices gave us a repository that abstracted out a single point of contact to get everything from the list. We could send a repository the list name — and the repository does all the handling — so you don't have to write 100 different lines of code to get to the same list.*"
- "*When we ran into problems, we could look to see how they had implemented to make sure we had implemented right. We had never deployed custom ASPX forms, so we used the guidance to learn about linked files. That would have taken us a long time to figure out. Because it was there we could follow their example. We ended up with more than 120 ASPX forms that were deployed.*"
- "*It saves time on planning — if someone comes to me and says we want to migrate an app, the first thing I do is take a look at what they have. Often you wish someone would have suggested architectural details that you wouldn't have thought of.*"

Accelerated application development

Reusable code components for common development challenges such as logging, validation, data access, and exception handling enable developers to shorten the time needed to develop tactical components of the applications and instead focus on tailoring the applications to specific business needs:

- "*There was some ramp up time in reading the patterns and practices guidance but after that development could be up to 25 percent faster.*"
- "*We reused some of the code and tailored it to [our client's] environment — from the code sample standpoint that helped us be 25 to 30 percent faster.*"

- "*It's all about reducing plumbing — I don't have to figure it out. When you're building software the last thing you want is to reinvent the wheel. Patterns and practices gives you a set of well-thought out and tested ways of doing the plumbing and lots of documentation on how to do it.*"
- "*I will turn to patterns and practices and say that, out of the box, I have 80 percent of what I need and this is a billion times better than starting from scratch.*"
- "*Everyone's on the same page about how the parts and pieces should be and how they should interact. I can walk into a client site and if they say they have an app I can see how it will be laid out. I know how things will talk to each other and I can be productive immediately because I know how they're doing things. I have a roadmap in my head and there's a strong likelihood they're going to be on it.*"
- "*The amount of time that patterns and practices dedicates to a problem is far more than we could dedicate. I can't have someone spend two months finding out the best way to log, for example. If the action comes down to building it yourself or using patterns and practices, patterns and practices will be better built, better thought out, and better documented than what you can do. Invariably developers spend 90 percent of their time on projects writing things that someone has already written before. We've tried to dramatically cut down on that, and patterns and practices gives us a common base. It removes 30 to 40 percent of the work and lets us focus on very domain-specific work that patterns and practices can't help me with."*

Developers adopting patterns and practices can expect an average productivity increase of 25 to 40 percent, depending on their skill level and the complexity of the application.

Accelerated application testing and debugging

Because the reusable code components and frameworks provided by patterns and practices have already been tested and debugged, the overall amount of code that needs to be tested and fixed is significantly reduced:

- "*They've already tested their framework and where it's not reliable there's a bug database so people can learn about workarounds. If I'm on a 6-month project, I could see three to four months consumed by testing work and fixing bugs. Now I can focus on testing my application.*"
- "*The whole series of test classes means they're testing as they go. There's no getting around testing and bug fixing but I don't have to worry about testing the plumbing — I still have to test how I use it but I don't have to test their logging tool, for example.*"
- "*I didn't have to spend a month and decide how we were going to do something — they've gone through and tested it and you have several thousand other people using it and testing it. You're getting a lot of QA for free.*"

Using patterns and practices dramatically reduces the testing and debugging time needed for an application, because all of the patterns and practices components have already been tested.

Improved application management

Loosely-coupled applications are easier to manage and evolve over time. Patterns and practices makes it easier for developers to build loosely-coupled applications, resulting in lower application maintenance costs and lower costs to update and enhance applications.

Lower cost of maintenance

As one customer said, "*Applications are easier to build and easier to maintain, and it's easier to work on one part without breaking the other parts. It's easier to be modular in the sense that they can decide at runtime whether a specific piece of functionality should be available or not*."

Lower cost of updating

Companies building custom applications often find staff turnover, new business requirements, and application complexity can make updating or changing custom applications extremely costly. This can present a particular challenge if a developer or consultant builds an application and then leaves without sharing, within code notes or another means, insights into how certain functions or processes were coded.

However, applications built using patterns and practices can be updated more quickly and cost-effectively because they rely on standard components and frameworks:

- "*If you didn't have Composite Application Guidance you would spend a week figuring out what it would have told you to do, then lose a few months trying to write the things Composite Application Guidance already writes — or you'd shortcut it. But where you would really get stuck is after v1. But you're in a much better position with Composite Application Guidance, because it's designed for change so every change is not going to cost you months*."
- "*Another huge benefit that is not always appreciated is that there are many many ways to do this. When you bring new people into your project you have to show them how you did it. If I can say, I have a composite application then I know how you're solving certain types of problems, where things will be*."
- "*It helps you build smaller decoupled parts. There's a maintenance argument for loose coupling. There are lots of ways to do it but Composite Application Guidance really helps you along. It helps you proceed in the best ways with the scaffolding and infrastructure to do it*."
- "*Composite Application Guidance saves you tons of code and when you go back later and try to change the application it's very loosely coupled so you can change it. If you just write what was initially asked for you can paint yourself into a corner. With Composite Application Guidance it's very loosely coupled and you can just extend it. If you have any composite application and you're not sure how it may evolve, you should invest a few days in learning Composite Application Guidance. If you're going to have an application that lasts any period of time, it's definitely worth it*."
- "*We just released a WPF application that is 100 percent Composite Application Guidance and Enterprise Library. We were under budget, under time, and the flexibility and maintainability of the application is outstanding. If we hadn't*

used Composite Application Guidance it probably would have doubled the time of development and we increased maintainability by a factor of three."

Improved application performance

Companies running applications developed using patterns and practices also found that the frameworks and code components were powerful building blocks to help them build more stable and reliable business applications. As one user said, "*Patterns and practices gives us some stability that we didn't have before. We build line of business applications for our corporation. Business users want business applications, not frameworks. Being able to download building blocks to develop powerful line of business application means we don't have to waste the time creating, for example, our own logging application*."

CONCLUSION

Writing business applications is much easier than it used to be. Improvements in tools and frameworks help boost developer productivity, and better languages make it easier to write better code. However, the real boost in both developer productivity and complex application performance and maintainability comes from reusable patterns, components, and frameworks that can be linked together to support business processes while being able to be independently changed and adjusted as business needs evolve.

Microsoft patterns and practices provides .NET developers with the building blocks and guidance to rapidly build complex, loosely-coupled applications to meet their current business needs and a structured application environment that can be cost-effectively adapted as needs change. Companies adopting patterns and practices can expect greater developer productivity but also improved ongoing application management and performance.