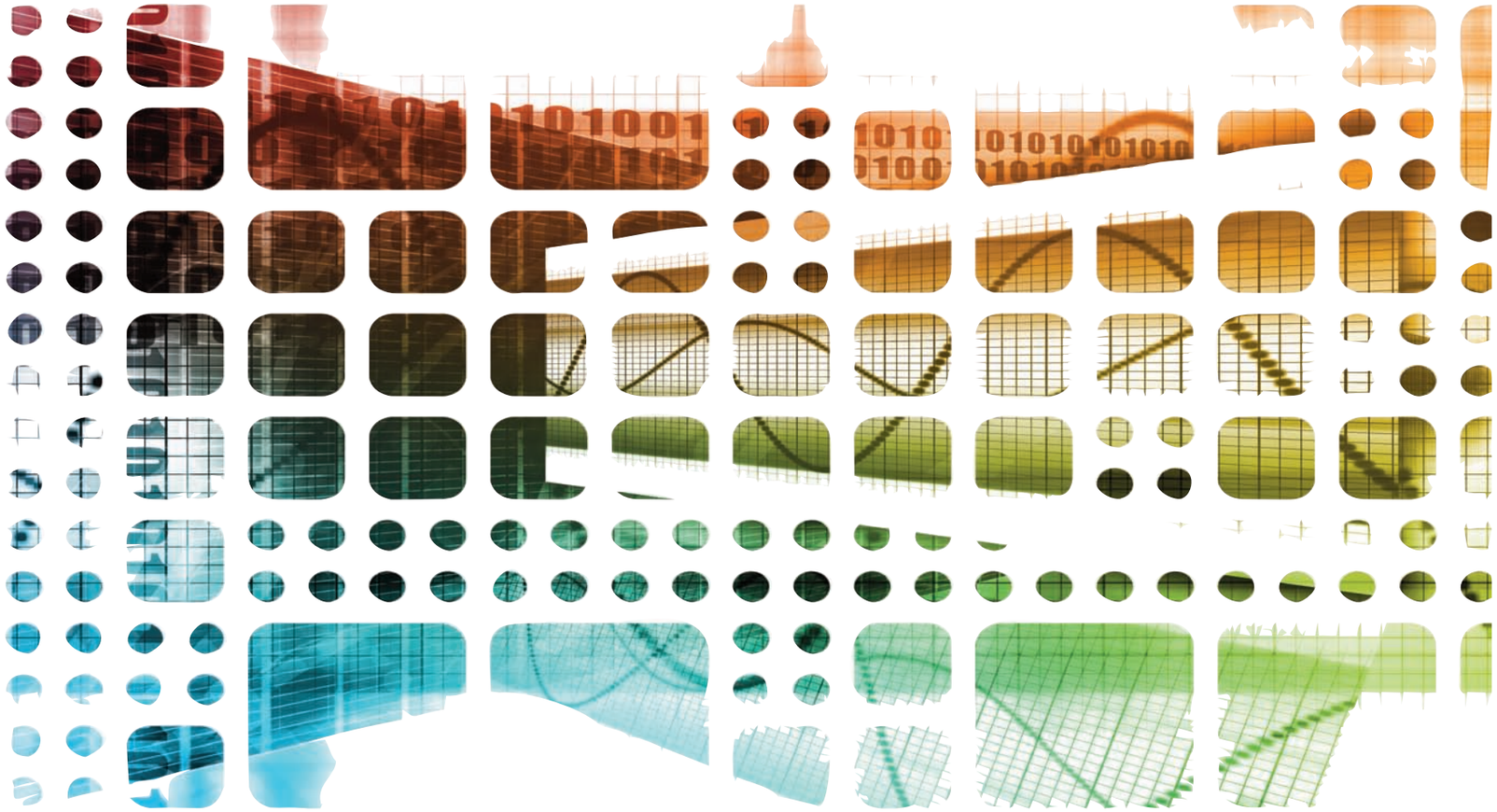
Dr. Dobb's Journal

FEBRUARY 2011



Windows Third-Party Ecosystem

Exploring new niches in Microsoft's new platforms

Next

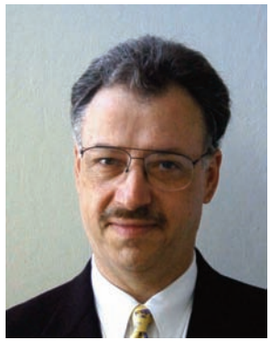
ALSO INSIDE

UBM TechWeb

www.drdobbs.com

Helping Out

My mission at Dr. Dobb's is to continue providing new, insightful information for serious developers



If you're a longtime reader of Dr. Dobb's, you're probably expecting to see the smiling face and sharp pencil of Jon Erickson in this space. Unfortunately, Jon has been on an extended health leave, which is likely to continue for a while still. During this period, Dr. Dobb's parent company, UBM Techweb, has hired me to work on many Dr. Dobb's editorial projects. This is a great honor for me personally, as I have been a reader and admirer since the mid 1980's. Like many of you, my developer skills were either born or greatly refined by reading the many excellent articles that have appeared in Dr. Dobb's during Jon's years at the helm.

Jon and I were peers at Miller Freeman Inc., where I headed up UNIX Review, the tech magazine for the UNIX industry. Since those days, I've been a technology analyst writing a regular column for *SD Times,* and I've reviewed numerous programming tools for *InfoWorld*.

My mission at Dr. Dobb's is to continue providing new, insightful information for serious developers — the kind of hands-on, practical, immediately usable stuff Dr. Dobb's has long been known for. I invite thoughtful comments and suggestions and can be reached directly at alb@drdobbs.com.

While I am excited to be contributing to Dr. Dobb's — it's the only publication that could have talked me out of leaving my consulting practice — I regret only that Jon Erickson, who wrote so many perceptive introductions to new issues and new themes, could not have written this one, too.

— Andrew Binstock is Executive Editor for Dr. Dobb's and can be contacted at alb@drdobbs.com.

Windows Third-Party Ecosystem

For developers, new niches in Microsoft's new platforms

By Eric J. Bruno

The Windows third-party developer ecosystem, carefully cultivated by Microsoft, has long been one of the most vibrant communities of its kind, adding value and innovation to the core Windows platform. Trusted third-party components can speed the development process, cut costs, and ultimately make the difference between project success and failure.

Windows is so pervasive in the technology industry that it's likely almost every commercial developer has done some sort of Windows development at one point in time. Through the years, third-party tools, frameworks, and libraries have blossomed around the popular Windows-based technologies of the day. Today, it's blossoming again around Silverlight and the cloud. Other areas of the Windows ecosystem include application design and code generation (IBM's Rational, for example), SharePoint tools, SQL Server modeling and development tools, and Windows server management. Tools help make developers successful, and Microsoft itself, by design, is far from the only vendor for native Windows application development. Let's explore this in greater detail.

Widget Libraries

Microsoft has always been a leader in user interface design, often creating Windows widgets (called Controls) for its products, then making them available to developers through the Windows Software Development Kit. Examples are the toolbar, with its associated tooltips and, more recently, the Ribbon interface. However, with special support for native custom Windows control and ActiveX control development, Microsoft also has fostered a third-party Windows widget ecosystem.

Early vendors such as Sheridan (acquired by Infragistics) offered specialized custom Windows controls. These widget libraries helped fill niches, target specific industries, and plug gaps in Windows' own toolkit. This meant that Microsoft wouldn't need to bear the burden of providing all of these widgets and controls, and third-party application developers were offered options through the very ecosystem that they were a part of.

This ecosystem has grown beyond single UI-based solutions and now extends to full libraries. The Boost C++ Libraries, which support the C++ Standard Template Library and other up-to-date C++ standards on Windows, is an example. In addition, communities such as Codeguru.com have grown to provide an avenue for developers to offer their libraries for all facets of Windows development to other developers within the ecosystem. This includes enhancements to the

.NET framework, Windows Presentation Foundation (WPF), the Windows Phone 7 Software Development Kit, and Silverlight.

The Open Source Question

Microsoft has resisted the open source movement in terms of its crown jewels: Windows and the Office suite. Even Visual Studio has remained closed with commercial licensing, though the relatively limited Express edition is available free of charge.

This doesn't mean that open source tools, compilers, and IDEs don't exist for Windows. IBM's Lee Nackman claims that the open source Eclipse IDE, along with the choice of name, was created to target Microsoft and, specifically, Visual Studio. Eclipse is known as a Java development environment, but combined with other open source frameworks, it supports native Windows application development. For example, the Minimalist GNU for Windows (MinGW) project provides a complete framework and set of open source tools for native Windows development. It contains no commercial libraries, but fully supports and depends on Microsoft's own libraries, such as the MSVCRT.DLL C Runtime for Windows.

Microsoft's decision to make the runtime easily accessible in this form was intentional, and it has allowed for the further growth of the developer ecosystem. As a result, Microsoft can boast support for open source Windows application development without having to release and support its own software with open source licensing.

Regardless of how you feel about Microsoft's strategy, it does result in a win-win scenario for both Microsoft and developers who want open source tools. For developers, the resulting open source market for tool and IDEs has lowered the entry costs for Windows development, increased mindshare among students, and cultivated startups around Microsoft technologies. The result is quicker and cheaper market solutions for third-party developers and an increase in Windows-based technologies being deployed in data centers.

Beyond The Desktop

Let's look beyond the classic desktop-server world of Windows and consider Microsoft's efforts to rally developers around its mobile technologies, rich Internet applications, and cloud computing platform. The Windows Phone Marketplace is a huge step forward in Microsoft's support for the Windows Mobile ecosystem.

This is one area where Microsoft is far behind the competition, which many consider inexcusable given how long its mobile technologies have been in the market. As for limitations, application development is limited to Microsoft's XNA or Silverlight 4 frameworks, while the development API still doesn't provide access to all of the phone's capabilities, such as Sockets.

To help drive its ecosystem in the face of stiff competition from Google, Apple, and RIM, Microsoft has made some of the development tools free, such as the integration of Expression Blend with Visual Studio 2010 Express. Developers can use the resulting toolset to design Windows Phone-friendly user interfaces, along with the code that makes them functional, that can be deployed as Windows Presentation Foundation or Silverlight applications.

Microsoft CEO Steve Ballmer announced at the Consumer Electronics Show earlier this month that there were 22,000 registered Windows Phone 7 developers, about 5,500 third-party applications, and around 100 new applications being launched each day. In terms of usage, customers are downloading a new Windows Phone 7 application each day, on average.

The door is open for third parties to create widgets, libraries, and data integration tools that make the environment even more powerful. And since Microsoft has made its portion of the toolchain free, it's certain to get it into the hands of developers more quickly; this presents greater market opportunities for tool vendors.

Increasing this opportunity is the fact that Silverlight isn't a platform-specific technology. In fact, it's not even Windows specific. The Silverlight runtime, along with browser plug-ins, exist for Apple Mac OS X, Linux, and the Symbian mobile OS, in addition to Microsoft's Windows and Windows Phone platforms. And since browser support includes Google's Chrome OS, Silverlight should run on Google's Linux-and-Chrome-based Chrome OS.

As a result, a heterogeneous third-party tools and developer ecosystem is popping up around Silverlight. Microsoft is embracing the fact that this ecosystem, for the first time, will extend beyond Windows and include tools and libraries for rival platforms, as well.

However, much like Apple's iOS developer IDE, Xcode, runs only on Mac OS X, Microsoft's revenue may be tied largely to the success of its Visual Studio line, which runs only on Windows.

In terms of cloud development, vendors such as Alpha Software, with its Alpha Five platform, are taking advantage of Microsoft's relative openness for integration with tools and software from other vendors. Although not purely a tools play, the openness that Microsoft is supporting opens the door to the heterogeneous deployments that other vendors have been proclaiming for years. The bet is that this strategy will overcome the "one-stop-shop" approach of companies like Oracle and IBM. Both strategies have advantages for customers — one-stop typically offers better integration, deployment, and support, while a heterogeneous approach avoids lock-in — yet the single vendor approach offers clear advantages to the vendor selling the integrated solution. It's yet to be seen how the open, heterogeneous strategy can be made into an advantage financially.

5 *Opportunities* FOR WINDOWS DEVELOPERS

1. Controls And Libraries There's a market for widgets and enhancements to .NET, Windows Presentation Foundation, Windows Phone 7 SDK, and Silverlight.

2. Open Source Projects like Minimalist GNU for Windows provide open source tools for native Windows development.

3. Mobile Expression Blend for Visual Studio 2010 Express can be used to design Windows Phone-friendly UIs, which can be deployed as WPF or Silverlight apps.

4. Cloud Computing Microsoft's cloud is fairly open for integration with tools and software from other vendors.

5. Tablets Microsoft has introduced touch capabilities with Windows 7. Whether it will tailor Windows Phone for tablets remains to be seen.

Tablet Development

The market for tablet and slate devices is growing. In fact, it's growing at the expense of netbooks and notebook class computers, which has the potential to further erode Windows' market share. For the past two years, a bevy of tablet computers has been demonstrated at CES, some of which run Windows.

Unfortunately, Windows Tablet edition hasn't progressed much

since 2002, when Microsoft experimented with pen computing. Today's Windows-based tablets are little more than the desktop version of Windows made to run on a touchscreen device, resulting in a less than optimal user experience.

However, Microsoft has introduced touch capabilities in all editions of Windows 7. The features included are based on the gestures API and touch UI enhancements introduced with Microsoft Surface, which are directly available to third-party developers, but is this enough?

Now if only Microsoft would tailor its Windows Phone platform for tablets, as opposed to shoehorning desktop Windows into these devices, it would potentially expand its ecosystem well beyond what it is today, as its tools and strategy would better address all markets — desktop, Web, cloud computing, netbooks, smartphones, and tablets. Time will tell if there's any chance of that happening.

The opportunity is there for third-party vendors to find niches associated with Microsoft's Azure cloud platform, Silverlight RIA approach, and Windows Phone technologies. Microsoft has a track record of profiting from a thriving third-party ecosystem, and if anyone can find a way to achieve ROI from a heterogeneous platforms approach, Microsoft has the most potential. The road won't be easy — after all, this strategy didn't work out well for Sun Microsystems. But one thing is clear: The opportunities for third-party vendors around Windows-based technology are greater than ever.

— *Eric is a* Dr. Dobb's *Contributing Editor and coauthor of the books* Real-Time JavaProgramming: With Java RTS *and* JavaFX: Developing Rich Internet Applications.

Tim Heuer on Silverlight

The Dr. Dobb's Microsoft Reference Center hosts a series of informative videos on topics relevant to Windows development, as well as whitepapers, training materials, and downloads.

Click on the screen above to watch a video wherein Microsoft project manager Tim Heuer discusses developer outreach for Silverlight.

Autonomous User Interfaces for Mobile Apps

A new way to customize and differentiate the look, feel, and function of mobile apps

By Robi Karp

The Autonomous User Interface (AUI) is a revolutionary approach to UI design and implementation that goes beyond the custom themes, icon sets, and color schemes common on many mobile phones and other intelligent devices. Through scriptable, autonomous UI coding, AUI lets OEMs, developers, integrators, and other ecosystem participants completely control and customize the look-and-feel of the end-user experience.

With traditional design methodologies, application code "owns" the particulars of UI implementation, determining the type, orientation, placement, and other attributes of objects on the display (buttons, widgets, etc.), the flow of their use and the callback code that powers those elements. The attributes of a UI design are thereby set in the original design and are only minimally mutable downstream, by channel partners, third-parties and end-users. Some UI and application frameworks support theming — customization of color schemes, menu text styles, window frames, widget sets, etc. However, the fundamental structure and flow of an application UI remains set in stone — a closed box as imagined by the original design team.

On the other hand, the concept of an Autonomous User Interface lets application developers specify generic or abstract presentation of controls, widgets and even content, giving downstream developers the freedom to brand and customize. Benefits to this approach include:

- Letting developers bind custom functionality to individual UI elements with run-time scripting
- Supporting addition and/or removal any item from an application UI, including images, videos, and widgets, without changing any application code (i.e., with binary program images)
- Enabling existing applications to integrate reaction with new device events and capabilities, like shaking and orientation (accelerometer), location and movement (GPS), and definable data and network events (calendars, stock quotes, sports scores, wireless traffic, etc.)
- Letting integrators, operators and end users easily add new UI personalities at run-time without changing shrink-wrap application code

- Supporting transformation of UI elements, including buttons, controls, images and videos with 2D/3D graphics effects, to scale, reflect, rotate, alpha blend, shadow, etc. with minimum pre-load or post-load software development effort

Autonomous UI Architecture

Breaking out application and presentation code doesn't require radical rethinking of the core application design. Application code can still solicit input and generate output. It is still up to the application design team to determine how much control resides inside the application itself vs. the amount exposed to subsequent modification. But decoupling does require specific support from the underlying graphical and multimedia framework.

Key enablers of an autonomous UI include:

- Providing safe binding between underlying graphical system APIs and an external, open programming environment
- Exposing inventories of (public) application objects that implement UI functions
- Supporting a protocol between presentation code and the application for information exchange

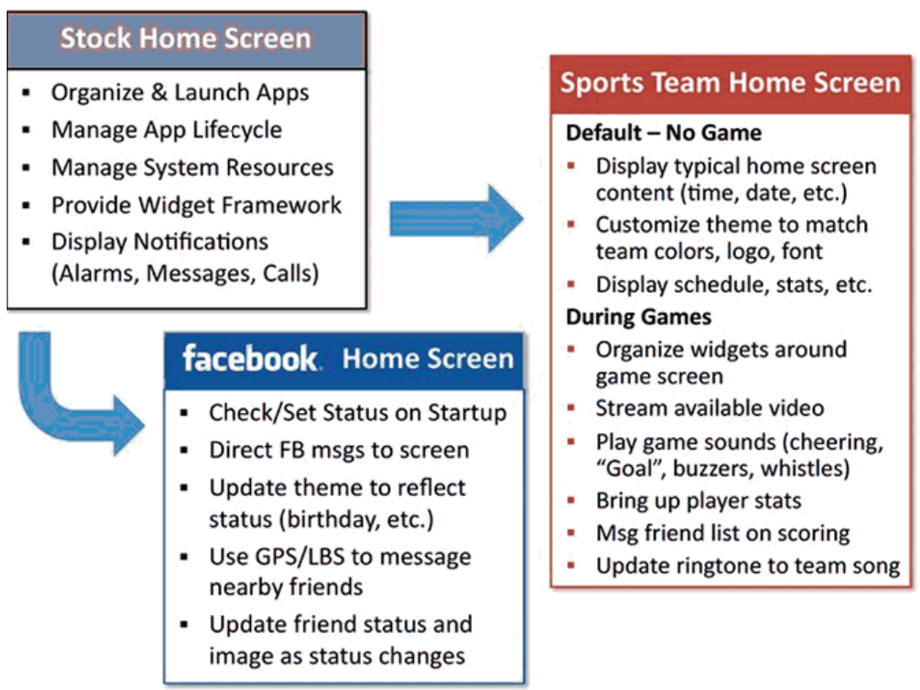


Figure 1: Adding/Extending Home Screen Attributes.

It is also important to provide an open high-level API for developer use. The API is of course the main point of entry for developers, and also simplifies translation of information between the language bindings using the protocol. At Fluffy Spider Technologies, we chose C for building FancyPants (our high-performance multimedia and graphics platform for embedded applications) and for underlying libraries. For enabling Autonomous UI code, we bind to the Lua scripting language at a high level, taking advantage of Lua features and rapid prototyping capabilities.

Real-World Examples

Most investment in differentiated user interface starts with the device home screen. Unfortunately, the investment often stops there too, providing a marginally unique look-and-feel or shallow theming of window adornment, background image and color of other elements. With AUI, device designers, their channel partners and third parties have myriad opportunities to customize the end-user experience in novel ways:

You can think of AUI in this context as an exercise in object-orient programming. The original, generic home screen is a base object inherited and extended by new objects (realized home screens) that integrate features and capabilities of specific devices and networks. A social media home screen would extend the base version to respond to status updates, messages from friends, location-based information, etc. Sports-themed home screen extensions could include live streaming video during game play, game sounds in real-time (buzzers, whistles, but no vuvuzelas, please), display of game scores and player stats, and social networking of game progress.

On a mobile handset, the device manufacturer will typically include an SMS (Short Messaging System) application, sourced from the mobile OS supplier, a third-party (ISV) or created in-house. This "preload" SMS application likely includes a traditional, display of messages and addressees. A mobile network operator (MNO) or other channel partner has few options for customizing or branding this kind of application, and is often forced to pass uninspired software through to end users "as is" or invest in replacing entire preload applications at considerable effort and expense.

Using AUI, an operator could enhance SMS addressee information with status and location-based data supplied by its network, or a third-party ISV could offer an alternate look-and-feel to that same SMS client and to other AUI-enabled application code as well. For example, MNO developers could enhance addressee information with status and location-based data supplied from the operator's network. Similarly, a third-party ISV could offer an alternate look-and-feel to that same SMS client, using previously unavailable functions like accelerometer input or GPS coordinates. All without modifying any original application code.

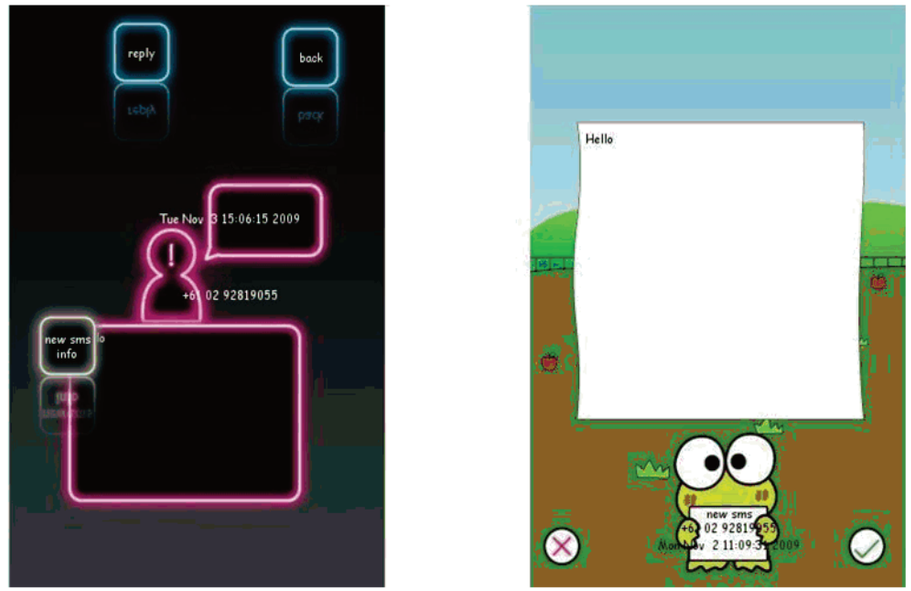


Figure 2: Two presentations of a single SMS application.

Depending on the industry, a new product (not just a new product version) can require between 2 to 10 or more man-years of engineering effort to reach the market. Mobile phones typically have a much quicker sales cycle and a much shorter market window (and usually involve a tremendous engineering investment). A significant part of the engineering effort lies in creating a compelling, differentiated user interface.

For OEM developers creating families of products with multiple members, being able to deploy the same application code base with a range of unique user interfaces saves time, money and can also help focus development effort on truly differentiating features. For

subsequent iterations of the same product line, an Autonomous UI helps new products in the family arrive to market more quickly and with confidence.

The arena where this phenomenon is most evident is in common operating platforms. Designers may choose a common, interoperable COTS OS like Android or WinCE to save on non-differentiating engineering, and to leverage existing (or evolving) ecosystems that revolve around those platforms, like the Android Market. However, these platforms typically leave little room for OEM branding and customization. Unless developers invest in significant incremental engineering, (as Motorola did with BlurUI), users will be greeted with the same UI as on every other gadget running the same OS, relegating the new device to the status of commodity (as in the PC market). And if OEMs do make the required investment, they will likely need to repeat that effort with each new platform release.

In 2009, OEMs launched dozens of devices and the Android Market (applications store) mushroomed to offer more than 30,000 applications. With projections for growing Android deployment in 2011, developers must increasingly invest in customizing Android to avoid delivering "me too" devices and applications. AUI, with its rich graphics and multimedia capabilities, provides the ideal vector to customize Android without forking the platform.

An AUI multimedia applications framework gives original equipment manufacturers (OEMs) of Android-based devices and their channel partners new capabilities to create visually rich applications and completely control and customize the look-and-feel of the end-user experience, thus easily differentiating their wares in an increasingly crowded marketplace.

Lua Scripting Language

Lua (http://www.lua.org/) is a powerful, fast, lightweight, embeddable scripting language. Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, runs by interpreting bytecode for a register-based virtual machine, and has automatic memory management with incremental garbage collection, making it ideal for configuration, scripting, and rapid prototyping.

Benefits of Decoupling Application and UI

Autonomous UI design is not just another way to subdivide application functionality. It actually offers developers benefits that emerge directly from decoupling UI and application code. These include:

- Shorter development time to create brand new, unique user interfaces without modifying application code
- Ability to add intelligence to existing UI code, e.g., to make decisions and process events unforeseen in the original design
- Ability to create a family of products based on a single application code base. High-end devices can have different user interface presentation and reactions from low-end phones.

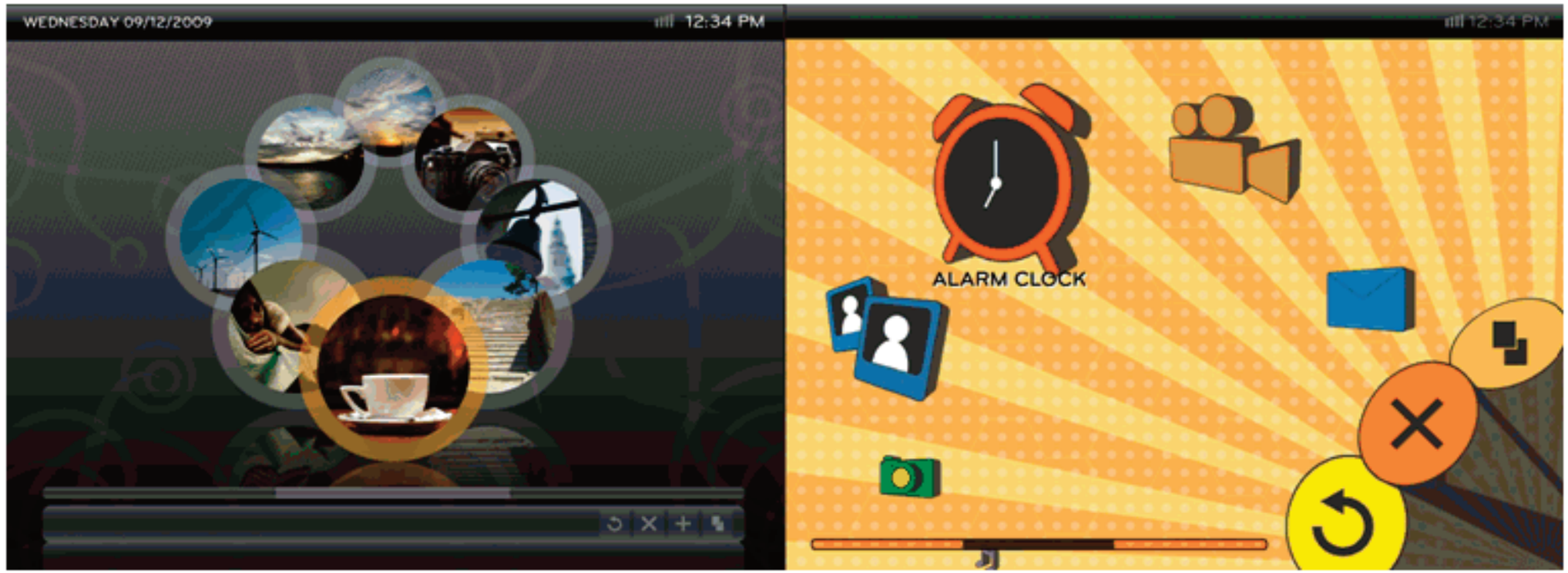


Figure 3: Two Android Home Screens.

- Shorter time to market for product iterations as the Q/A time and software development time is reduced
- Enhanced brand retention — the ability to build devices with unique look and feel, even on commodity software (and hardware) platforms

Conclusion

This article has explored the concept of decoupling UI and application design principles. This capability expands the market and extends the lifetime of application code by offering developers and other ecosystem players new opportunities to brand, differentiate and refresh device software. In today's dynamic landscape of multiple application OSes (especially in mobile), it's important to build a strong base product that can be easily tailored for different packages, channels, and markets.

— Robi Karp is CEO of Fluffy Spider Technologies (http://www.fluffyspider.com/).

Q&A with Jason Beres

The mechanics of Windows third-party development

By Deirdre Blake

Windows success as a platform can be attributed in part to its thriving community of third-party developers. Jason Beres, Infragistics' VP of product management is a notable member of that community. He's worked as a .NET architect and evangelist, and helped found Florida and New Jersey .NET user groups. He's also coauthor of *Professional Silverlight 4, C# Bible,* and *Visual Basic .NET Bible. Dr. Dobb's* managing editor Deirdre Blake recently talked with Beres about the mechanics of third-party development.

Dr. Dobb's: How do you coordinate your development with Microsoft releases and updates?

Beres: As part of the Microsoft developer ecosystem and partner program, we have direct access to early previews of Microsoft technologies. This lets us develop community technology previews and release new user interface components and UI development suites 30 to 60 days after Microsoft introduces a new technology or platform. We also participate in Microsoft developer labs and on-site meetings.

Dr. Dobb's: What are some of the tradeoffs you face in development, and how does your team balance them?

Beres: The balancing act comes as we continue to invest in our UI control toolsets for the mature platforms — still heavily in use — and develop new UI controls for emerging technology and platforms.

Dr. Dobb's: What are the greatest challenges and opportunities for third-party developers this year?

Beres: The greatest challenges are the emerging technologies and how to make the right investments in them while still satisfying our customer

base — some of whom may not be moving to these technologies.

We see numerous opportunities: The continued adoption of Silverlight as a rich line-of-business platform with a great deployment model, as well as Web development in general with the continuing explosive growth of jQuery, and the higher adoption rates of Microsoft's MVC stack as well as the emerging HTML 5 standard. If that isn't enough, there's a greater focus on mobile and tablets in the enterprise.

Dr. Dobb's: Has mobile's transformation and the emergence of cloud platforms affected your efforts?

Beres: No. Development effort and goals for mobile and the cloud are no different than UI control development for any emerging technology. With parity in our features and UI controls across multiple platforms, we can build UI control toolsets for new platforms and devices that enable our customers to create applications with the best user experiences possible.

— Deirdre is Dr. Dobb's *Managing Editor.*

Silverlight Developers, Take Note

Visual Studio 2010 SP1 Beta 1 Brings Concurrency Profiling to Silverlight

By Gastón Hillar

Microsoft's latest service pack for Visual Studio 2010 has something Silverlight application developers will find particularly useful. It's the addition of performance wizards to do profile concurrency, which lets them know if multithreaded apps are executing properly. The tool spits out a graph showing how each thread is behaving — especially important for developers to make sure multithreaded applications are scaling properly on newer multicore machines. Visual Studio 2010 Premium and Ultimate versions introduced powerful concurrency profiling features for Windows applications through new options in the Performance Wizard. However, those performance profiling tools weren't available for Silverlight projects.

Visual Studio 2010 Service Pack 1 Beta 1 (www.microsoft.com/downloads/en/details.aspx?FamilyID=11ea69cb-cf12-4842-a3d7-b32a1e5642e2) — or SP1 Beta 1 — removes this restriction and lets you select a Silverlight project as the target for a profiling session. Thus, you can visualize the behavior of a multithreaded Silverlight application.

By default, a Silverlight application uses multithreaded code under the hood, which you have to take into account when profiling concurrency. (Meaning, don't be confused by the additional threads that will result in the concurrency analysis.) The first time you run a concurrency profiling session for a Silverlight project, you might get a bit overwhelmed by the number of worker threads, even if your code just works on the main thread. However, the profiler has detailed information that will help you focus on your threads.

When you use the profiling tools with a Silverlight project, it's very important to finish the execution of the application as soon as possible to avoid collecting unnecessary information. Once the code you want to analyze has finished its execution, you must close the browser with the Silverlight application so that your profiling session will be properly targeted.

The profiler generates a collection of graphs that provide information about the behavior of a multithreaded Silverlight application, and these can be used to detect common problematic patterns in parallelized applications. However, it's best to isolate any suspicious algorithm in a simple application to shorten its detailed analysis. If you try to profile a

very complex Silverlight application, the process will require more time and effort to detect certain important problems, such as a serialized execution. The result of a profiling session provides detailed information about CPU utilization, the threads, and how these threads are mapped to the available logical cores.

Because Silverlight 4 doesn't let you work with the Task Parallel Library introduced in .NET Framework 4, it's still necessary to use threads to take advantage of multicore functionality in your Silverlight applications. The threads view provides timelines and the execution profile of each thread. You can use this information to optimize your code.

"As a bonus, SP1 Beta 1 includes the most updated version of Silverlight 4 Tools"

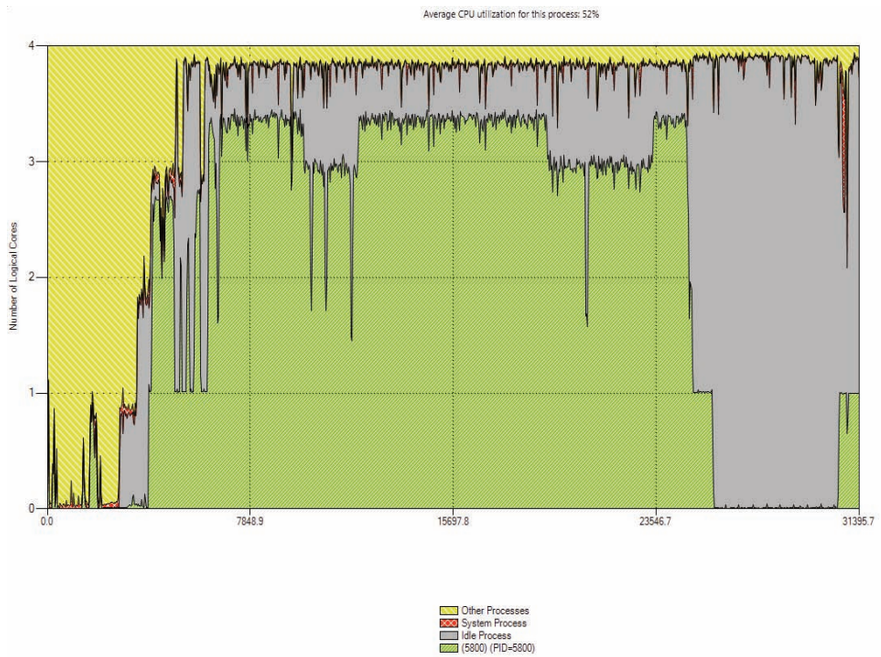


Figure 1. The CPUutilization graph for a multithreaded Silverlight 4 project.

Silverlight 4 is becoming very popular for developing line of business applications and Rich Internet Applications that consume data and services. And if you use asynchronous managed APIs, such as SharePoint Silverlight Client Object Model, you will work with multithreaded code, because the callbacks run on new threads. You can use the SP1 Beta 1 concurrency profiling features to optimize the execution of these asynchronous calls and translate the available multicore power into application performance.

As a bonus, SP1 Beta 1 includes the most updated version of Silverlight 4 Tools. The debugger adds support for IIS Express, and you'll also welcome the new local Help Viewer — a client application that makes it easier and faster to navigate through the help contents. If you work with Silverlight 4 integrated into SharePoint 2010 sites, the IntelliTrace support for SharePoint projects will improve your debugging experience. SP1 Beta 1 comes with a go live license; therefore, you can use it for production-related work. However, keep in mind it's still a beta version, with all that entails, and isn't yet compatible with Visual Studio Async CTP.

— Gastón is the author of **Professional Parallel Programming with C#** *(http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470495995.html).*

Tough Apps

Pretty Good at Half the Speed

The traveling salesman project gets the glory,
but the mother of practical scheduling algorithms is the shop scheduling problem

By Dennis Shasha

Dennis Shasha is a professor of Computer Science at New York University. His latest puzzle book *Puzzles for Programmers and Pros* outlined some general approaches to programmatic puzzle solving. This column challenges you to find the solutions to the puzzles that lie at the core of some cool tough applications. He solicits your help in identifying such cool apps.

If you've been reading *Dr. Dobb's* for many years, you may remember Dr. Ecco's Omniheurist Corner. In that column, Professor Shasha posed puzzles that required computer solutions. Dennis is returning to Dr. Dobb's, but with a new twist on puzzles. The idea is to take inspiration from the challenging apps that readers have faced and to turn those into puzzles.

If you have written an application that required extensive use of heuristics (because no algorithm would solve it) or that is otherwise algorithmically interesting, please send mail to Dennis at shasha@cs.nyu.edu. Put "Tough Apps" in the subject line. If he likes it, he will contact you. Then he will try to understand what you've done and create a puzzle explaining the algorithmic core. You will write a little blurb about yourself and the problem if you want.

In many sports, a factor of two from the world record is still respectable. If you can swim 400 meters in less than twice the time of Michael Phelps, climb El Cap half as fast as Lynn Hill, or windsurf at half the speed of Antoine Albeau, you are doing well.

Similarly, many problems that have no fast ("polynomial time") algorithm have fast approximation algorithms that give results within a factor of two of the optimal conceivable solution. Heuristics will often help improve beyond the factor of two, but starting with a half-as-good-as-optimal approach gives a foundation from which you can build.

Many such problems are scheduling algorithms. While the traveling salesman project gets the most press, the mother of practical scheduling algorithms is the shop scheduling problem. The problem also enjoys the virtue of being relevant to parallel and distributed computing.

Shop scheduling goes like this: You have **M** machines and **J** jobs. Any job can be done on any machine, but some jobs take more time than others. How can you get all jobs done in as little time as possible?

Remarkably, the following simple algorithm is within a factor of two of the optimal. Put jobs in a list in some order (any order will do). When a machine is idle, assign the next job in the list to that machine. How can such an apparently naive algorithm give any guarantee at all? Let's see.

Suppose the last job to complete begins at time **t**. Let **L** be the time to do that last job. Therefore the total time to complete all the jobs is **t + L** using this algorithm. Note that before **t,** all machines are used, because the naive algorithm uses every idle machine as it becomes available. Therefore the optimal schedule cannot use less than **t** time units.

Further, the optimal schedule cannot complete before time **L,** because at least one job takes **L** time. So, if **t >= L**, then **t + L <= 2t <= 2 * time of the optimal schedule**. Similarly, if **L >= t**, then **t + L <= 2L <= 2 * time of the optimal schedule**. So, the naive algorithm yields a schedule that is no more than twice as long as the optimal possible schedule. What's cool is that you can prove this without even trying to find the optimal schedule.

This analysis immediately suggests a heuristic: Put the most time-consuming job remaining on the next available device. That will tend

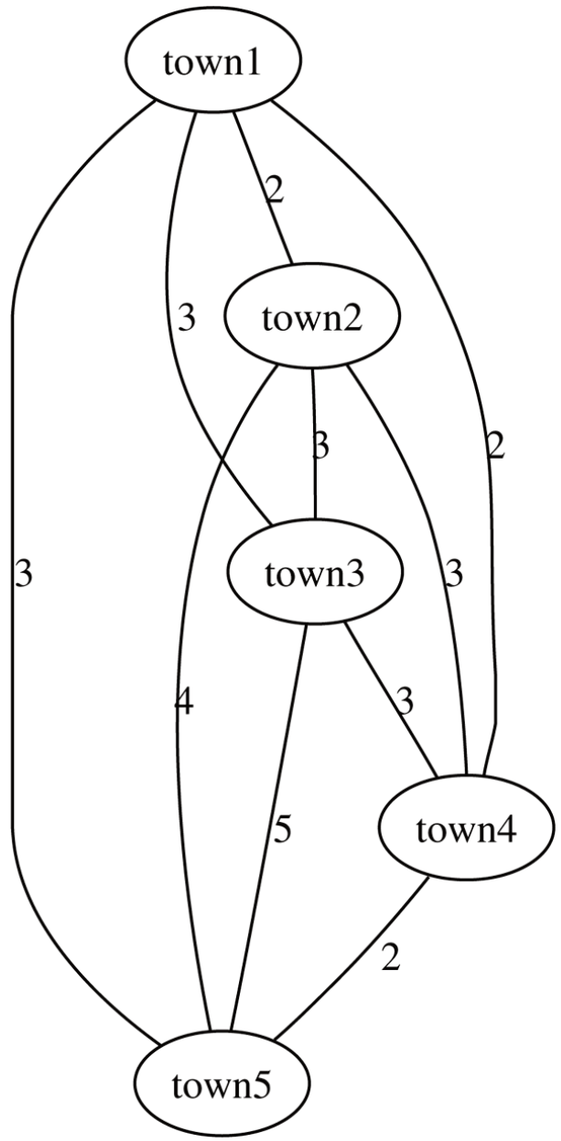


Figure 1: A simple graph.

to avoid having the longest task scheduled near the end. It does not however guarantee an optimum.

The Puzzle

1. Can you show an example in which this algorithm does not attain an optimal schedule (though it is within a factor of two)?

This factor-of-two idea works in a variety of contexts. Suppose you are sending machine parts from Trenton to Scranton on identical flatbed trucks. The width of each machine part extends the entire width of a truck. Further, no part can be stacked on top of another and the part must lie completely on the flatbed.

2. Can you guarantee to use no more than about twice the minimum conceivable number of trucks?

As long as we're in the transportation business, let's return to the traveling salesman problem. That problem goes like this. A salesman wants to travel to a set of towns by road, starting say from town **A**. The question is which route to take to minimize the total time spent in traveling. It's OK for the salesman to go through a town twice provided he visits every town at least once.

For the factor-of-two method to work, we need two assumptions. First, symmetry: it should take the same time to go from town **x** to **y** as to go from **y** to **x** for any towns **x** and **y**. Second, triangle inequality: the time it takes to go from town **x** to town **y** through town **z** is greater than or equal to the time of the shortest path from **x** to **z.** That is, it shouldn't be faster, though it can be just as fast, to take a detour. As a consequence of these assumptions, a shortest route through the towns never needs to loop. So the shortest route is a special case of a tree, which is a graph structure without loops.

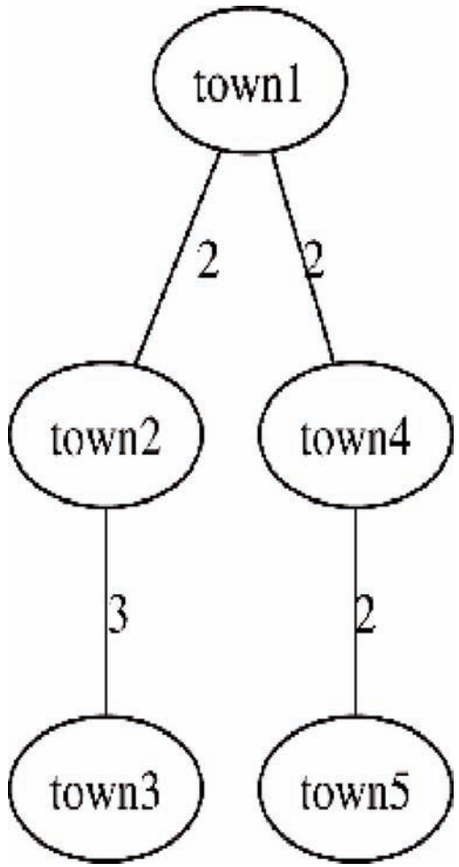


Figure 2: A minimum spanning tree.

Finally, there is a very efficient algorithm to find a smallest cost tree that connects all the towns. The minimum spanning tree algorithm, in the version developed by Joseph Kruskal in the 1950s, works like this: Consider the towns to be nodes and roads to be edges. Form the edges of the minimum spanning tree **TreeEdges** as follows starting from the empty set:

```
Until you have a tree that connects all nodes,
  if the time to traverse the edge between x and y
       is among the smallest for any edge not already
       in TreeEdges and TreeEdges does not yet contain a path
between x and y,
          add the edge between x and y to TreeEdges
  else discard the edge between x and y.
```

"Why is the heuristic of doing the most expensive job first not optimal?"

As an example, Figure 1 shows a simple graph and Figure 2 shows a minimum spanning tree.

3. How can you use the minimum spanning tree to find a route that is no more than twice the optimal route in cost? How would you improve this?

Solutions

1. Why is the heuristic of doing the most expensive job first not optimal? Well, suppose that there are 5 jobs: 45 min, 40 min, 15 min, 11 min, 9 min and two machines. According to the heuristic, they should be scheduled this way m1: 45 min, 11 min m2: 40 min 15 min, 9 min

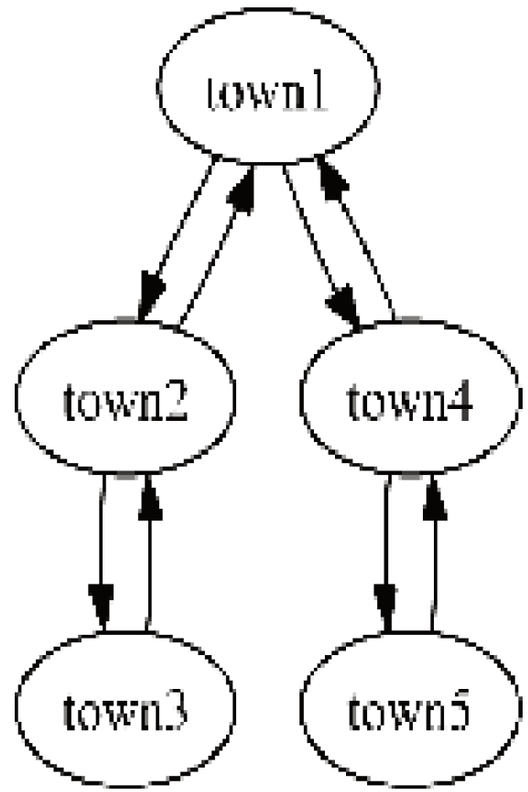


Figure 3: Going up and down tree edges.

But it would be better for m2 to do 11 and 9 when it's finished with the 40 minute job. Then all jobs would complete in 60 minutes rather than 64 minutes.

2. Order the items from longest to shortest. Put the trucks in a line. For each item, put it on the first truck on which it will fit. All trucks but one will be at least half full. Here is why: If any two trucks **A** and **B** are less than half full and **A** precedes **B** in line, then all the items on **B** would have gone on **A** or onto some other truck. So this is within a factor of two of the minimum possible number of trucks (possibly with one extra truck). In practice, it will usually be much better.

3. Go up and down the minimum spanning tree edges as shown in Figure 3. Since the minimum spanning tree costs no more than the optimal route, traversing every edge both ways cannot cost more than twice the optimal route. This can be improved by taking shortcuts as shown in Figure 4.

Further Reading

How to Solve It: Modern Heuristics by Z. Michaelewicz and D. Fogel, Springer Verlag, ISBN 3642061346.

Approximation Algorithms for NP-Hard Problems by Dorit Hochbaum, Brooks/Cole Pub Co; ISBN: 0534949681; 1st edition (July 26, 1996).

— Dennis Shasha is a professor of Computer Science at New York University. You can contact him at shasha@cs.nyu.edu.

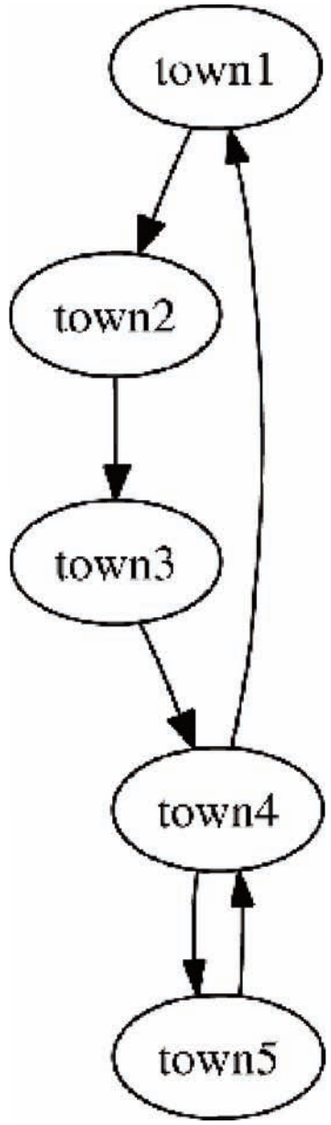


Figure 4: Shortcuts.

Blog of the Month

Conversion Constructors and Subtle Dangers

Oh the suprises you'll find!

Matthew Wilson

Pretty much every old C++ war-horse knows to use the explicit keyword when defining constructors with one only (non-defaulted) parameter. Even so, the subtleties of what can go wrong with the best laid plans of library designers can still surprise. Recently, a user of the Pantheios diagnostic logging library (http://www.pantheios.org/) reported a subtle vulnerability involving conversion constructors in a particular set of circumstances.

Background

For reasons of robustness, Pantheios does not accept arguments of fundamental types — **int, char, double, void*** — and so on. All arguments must be of a string type, or one which the Pantheios Application Layer knows how to interpret as a string. Thus, a statement such as the following will be rejected by the compiler: [Note: all the log statements here would usually be expressed on one line. Limitations of this medium require I split them.]

```
#include <pantheios/pantheios.hpp>

pantheios::log_NOTICE(
  "secret of life, universe, and everything: "
, 42);
```

To incorporate a fundamental type instance into a log statement a user must first convert it to a string. One (bad) way would be to convert it to a string yourself as follows:

```
#include <pantheios/pantheios.hpp>

char num[21];
snprintf(num, 21, "%d", 42);
pantheios::log_NOTICE(
  "secret of life, universe, and everything: "
, num);
```

There are several reasons why this is bad: it is verbose; it is not practically portable, because it uses **snprintf()**, which is "deprecated" by Microsoft's later compilers in favour of so-called safe equivalents (in

this case the famous **_snprintf_s());** it is potentially inefficient, because the conversion from **int->string** is made even when statements at log level NOTICE are disabled; it requires you to use magic numbers or some kind of **dimensionof()** macro. A better way is to use one of the stock inserted classes/functions provided by Pantheios, in this case the **pantheios::integer** inserter class, which may be used to insert all integer types (and with different width, radix, etc.), as in:

```
#include <pantheios/inserters/integer.hpp>
#include <pantheios/pantheios.hpp>

pantheios::log_NOTICE(
  "secret of life, universe, and everything: "
, pantheios::integer(42));
```

Incidentally, if you (understandably) think that's getting a tad verbose, you can use some of the provided namespace/class/function aliases to shorten the log statements, as in:

```
#include <pantheios/inserters/i.hpp>
#include <pantheios/pan.hpp>

pan::log_NOTICE(
  "secret of life, universe, and everything: "
, pan::i(42));
```

Problem

Now you know how insertion of an integer should look, let's consider how things should go if you try to insert it directly. With Visual C++ 9, I get a set of compilation errors along the following lines:

```
h:\freelibs\pantheios\main\1.0\include\pantheios\internal\generated\log_sev_functions.hpp(9998) : error C2665: 'stlsoft::winstl_project::c_str_len_a' : none of the 17 overloads could convert all the argument types

h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string\time.hpp(672): could be 'stlsoft::winstl_project::ws_size_t stlsoft::winstl_project::c_str_len_a(const SYSTEMTIME &)'

h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string\time.hpp(720): or 'stlsoft::winstl_project::ws_size_t stlsoft::winstl_project::c_str_len_a(const FILETIME &)'

h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string\time.hpp(753): or 'stlsoft::winstl_project::ws_size_t stlsoft::winstl_project::c_str_len_a(const UDATE &)'
```

```
h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string
\hwnd.hpp(561):   or   'stlsoft::winstl_project::ws_size_t
stlsoft::winstl_project::c_str_len_a(HWND)'

h:\stlsoft\releases\1.9\stlsoft\include\comstl\shims\access\string
\variant.hpp(556): or 'stlsoft::comstl_project::cs_size_t stl-
soft::comstl_project::c_str_len_a(const VARIANT &)'

h:\stlsoft\releases\1.9\stlsoft\include\comstl\shims\access\string
\guid.hpp(298):   or   'stlsoft::comstl_project::cs_size_t
stlsoft::comstl_project::c_str_len_a(const GUID &)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\time.hpp(254):         or         'stlsoft::ss_size_t
stlsoft::c_str_len_a(const tm &)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\time.hpp(143):         or         'stlsoft::ss_size_t
stlsoft::c_str_len_a(const tm *)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\basic_string.hpp(392):     or     'stlsoft::ss_size_t
stlsoft::c_str_len_a(const std::string &)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\exception.hpp(299):      or      'stlsoft::ss_size_t
stlsoft::c_str_len_a(const std::exception &)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\fwd.h(100): or 'stlsoft::ss_size_t stlsoft::c_str_len_a(const
stlsoft::ss_char_a_t *)'

h:\freelibs\pantheios\main\1.0\include\pantheios\pantheios.h(1459)
:    or    'size_t    pantheios::shims::c_str_len_a(const
pantheios::pan_slice_t &)'

h:\freelibs\pantheios\main\1.0\include\pantheios\pantheios.h(1549)
:    or    'size_t    pantheios::shims::c_str_len_a(const
pantheios::pan_slice_t *)'

h:\freelibs\pantheios\main\1.0\include\pantheios\pantheios.h(1592)
: or 'size_t pantheios::shims::c_str_len_a(pantheios::pan_sever-
ity_t)'
    while trying to match the argument list '(const int)'
     h:\publishing\blogs\ddj\code\conversion_constructors\exam-
ple1\example1.cpp(8) : see reference to function template instan-
tiation 'int pantheios::log_NOTICE<const char[43],int>(T0
(&),const T1 &)' being compiled
    with
    [
     T0=const char [43],
     T1=int
    ]
  h:\freelibs\pantheios\main\1.0\include\pantheios\internal\gener-
ated\log_sev_functions.hpp(9998)    :    error    C2665:
'stlsoft::winstl_project::c_str_data_a' : none of the 17 overloads
could convert all the argument types

h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string
\time.hpp(448):  could  be  'stlsoft::basic_shim_string<C>
stlsoft::winstl_project::c_str_data_a(const SYSTEMTIME &)'
    with
    [
     C=stlsoft::ss_char_a_t
    ]

h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string
\time.hpp(482): or 'stlsoft::basic_shim_string<C> stlsoft::win-
stl_project::c_str_data_a(const FILETIME &)'
    with
    [
     C=stlsoft::ss_char_a_t
    ]
```

```
h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string
\time.hpp(514): or 'stlsoft::basic_shim_string<C> stlsoft::win-
stl_project::c_str_data_a(const UDATE &)'
    with
    [
     C=stlsoft::ss_char_a_t
    ]

h:\stlsoft\releases\1.9\stlsoft\include\winstl\shims\access\string
\hwnd.hpp(529):                                                 or
'stlsoft::winstl_project::c_str_ptr_HWND_proxy<C> stlsoft::win-
stl_project::c_str_data_a(HWND)'
    with
    [
     C=stlsoft::winstl_project::ws_char_a_t
    ]

h:\stlsoft\releases\1.9\stlsoft\include\comstl\shims\access\string
\variant.hpp(489):                                              or
'stlsoft::comstl_project::c_str_VARIANT_proxy_a
stlsoft::comstl_project::c_str_data_a(const VARIANT &)'

h:\stlsoft\releases\1.9\stlsoft\include\comstl\shims\access\string
\guid.hpp(256):                                                 or
'stlsoft::comstl_project::c_str_ptr_GUID_proxy<C> stlsoft::com-
stl_project::c_str_data_a(const GUID &)'
    with
    [
     C=stlsoft::comstl_project::cs_char_a_t
    ]

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\time.hpp(241):   or   'stlsoft::basic_shim_string<C>
stlsoft::c_str_data_a(const tm &)'
    with
    [
     C=stlsoft::ss_char_a_t
    ]

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\time.hpp(100):   or   'stlsoft::basic_shim_string<C>
stlsoft::c_str_data_a(const tm *)'
    with
    [
     C=stlsoft::ss_char_a_t
    ]

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\basic_string.hpp(226): or 'const stlsoft::ss_char_a_t *stl-
soft::c_str_data_a(const std::string &)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\std\exception.hpp(223): or 'const stlsoft::ss_char_a_t *stl-
soft::c_str_data_a(const std::exception &)'

h:\stlsoft\releases\1.9\stlsoft\include\stlsoft\shims\access\strin
g\fwd.h(93):      or      'const      stlsoft::ss_char_a_t
*stlsoft::c_str_data_a(const stlsoft::ss_char_a_t *)'

h:\freelibs\pantheios\main\1.0\include\pantheios\pantheios.h(1437)
: or 'const char *pantheios::shims::c_str_data_a(const pan-
theios::pan_slice_t &)'

h:\freelibs\pantheios\main\1.0\include\pantheios\pantheios.h(1527)
: or 'const char *pantheios::shims::c_str_data_a(const pan-
theios::pan_slice_t *)'

h:\freelibs\pantheios\main\1.0\include\pantheios\pantheios.h(1574)
:                or                'const                char
*pantheios::shims::c_str_data_a(pantheios::pan_severity_t)'
   while trying to match the argument list '(const int)'
```

Although this is scary at first, it actually holds the essential truth of the problem, something often not the case with C++ compile errors (particularly with moderate-heavy use of templates). The problem is precisely as described (all 17 times!): There are no overloads of **c_str_len_a()** and **c_str_data_a()** for **"the argument list '(const int)'"**. These two function-overload suites are string-access shims, which are how Pantheios meets its design parameters of expressiveness, flexibility, and performance — I'll get into this in more detail in an upcoming article series on diagnostics I will be writing for *Dr Dobb's* this year. (The articles will also discuss why the library must reject fundamental type arguments, via the absence of string-access shims defined for numeric types; for now, you'll just have to take my word.) Many string access shim overloads are provided by the

STLSoft libraries, on which Pantheios depends, as well as other open-source libraries (and several of my closed-source commercial developments for clients, allowing their types to be succinctly logged).

So far, so good. This behavior is all by-design, and is actually of benefit to the programmer; it brings potential runtime problems into firm compile-time problems. In all the years I and others have been using Pantheios, this mechanism had never been subverted. Or, at least that was the case until late last year, when a user reported that, after passing an integer directly in a log statement, it had not been displayed correctly in the log. Alarm bells still ringing, we investigated and, lo and behold, it transpired that the integer was indeed being passed to a pair of string access shims, via the conversion constructor of the ATL type **CComBSTR**, for which string access shims are defined (although only in wide-string builds, hence **c_str_len_w()** and **c_str_data_w()**). This only occurred when using ATL on Windows and building a wide-string version of an application, in which case, the pre-processor symbol **UNICODE** is defined. What actually happens is that the integer value is passed by the compiler to the (non-explicit) single-parameter **CComBSTR(int nSize)** constructor, as part of the compiler attempting to find match for the argument. That constructor is used to pre-allocate a buffer of the given size for later use. So what happens in the case of our log statement is that the statement written out is:

```
    "secret of life, universe, and everything: " <<- we are left
to ponder ...
```

What should have been "**42**" is actually an empty string, since the secretive **CComBSTR** instance is empty of content (although it has a buffer of 42-character ready to receive some content). As you can imagine, had the integer had a very high value we may also have had an unwanted memory allocation failure to go along with the malformed log statement.

Naturally enough, when we'd got to this point, I wondered with some heat why on earth the ATL designers had not seen fit to declare that constructor explicit. But what to do about it? I cannot proscribe the use of ATL to Pantheios' users. I cannot remove the requisite wide-form string access shim overloads — **c_str_len_w(CComBSTR const&)** and **c_str_data_w(CComBSTR const&)** — from STLSoft, since they are established and used in other things. I can recommend that users compile a multibyte-string version of their code, as well as the wide-string one they want, to act as a guard, but that is often not possible. One thing I could do would be to remove the implicit inclusion of the requisite STLSoft libraries from Pantheios when compiling for wide-string in the presence of ATL. But anyone using Pantheios under those conditions who wanted to be able to insert **CComBSTR** instances into a log statement — a pretty common requirement — would be disadvantaged.

I confess I thought long and hard about this, and came up with several partially formed, baroque, and long-winded solutions, until the light finally shone on me. I'd even written about such things in the first chapter of my first book, *Imperfect C++*, back in 2004: the answer is constraints.

So, all that's been required to allow Pantheios once more to claim 100% type-safety in all circumstances is the insertion of constraints along the lines of that shown in the following 3-parameter application layer **log_NOTICE()** statement function template:

```
template<typename T0, typename T1>
int log_NOTICE(
  pan_sev_t severity
, T0 const& v0
, T1 const& v1
)
{
  STLSOFT_STATIC_ASSERT(0 ==
    stlsoft::is_fundamental_type<T0>::value);
  STLSOFT_STATIC_ASSERT(0 ==
    stlsoft::is_fundamental_type<T1>::value);
  . . .
```

The constraint is pretty canonical for these things: a static assertion is used to enforce a compile-time characteristic elicited via a meta-programming type. Specifically, **is_fundamental_type** is used to verify that the types **T0** and **T1** are not fundamental types. If either of them is, as in the original case, the compiler will fail on that line.

So far, all the compilers that used to support Pantheios are happy with the additional compile-time load — remember, these are compile-time constraints, and make absolutely no different to runtime speed/behavior — except for GCC 3.x and Digital Mars C/C++, for which the constraints are diluted/elided.

Conclusion

1. When you're designing a library involving C++ classes, be sure that you mark explicit every one-parameter (or one-non-default-parameter) constructor, unless you explicitly want to allow implicit conversion construction. Yes, I am aware that that's a tongue-twister. And aware too of the irony. But a language as long-lived and successful as C++ has a lot of backwards compatibility to maintain. Such wrong-way-round situations extend to more than just explicit/implicit, and maybe I'll have a carp about them on another day.

2. When you're designing a library involving arbitrary heterogeneous types, remember the utility of constraints (particularly compile-time ones) for controlling how far the envelope of accepted types may be stretched.

— Matthew has been programming professionally for nearly two decades, and by now knows enough about C, C++ and C#/.NET to be able to point out their imperfections with authority and affection. A development consultant for Synesis Software, he's also the creator of the FastFormat, Pantheios, recls and VOLE libraries, and the author of several books, including **Imperfect C++** *and* **Imperfect C#.**

Join the discussion at www.drdobb's.com!
To comment on Matthew's blog, go to
www.drdobbs.com/blog/archives/2011/01/conversion_cons.html

IN THIS ISSUE

Dr.Dobb's

Jon Erickson Editor in Chief, Dr. Dobb's
jerickson@drdobbs.com

Andrew Binstock Executive Editor, Dr. Dobb's
alb@drdobbs.com

Deirdre Blake Managing Editor, Dr. Dobb's
dblake@techweb.com

Amy Stephens Copyeditor, Dr. Dobb's
astephens@techweb.com

Sean Coady Webmaster, Dr. Dobb's
scoady@techweb.com

CONTRIBUTING EDITORS

Mike Riley

Herb Sutter

DR DOBB'S
UBM TECHWEB

303 Second Street,
Suite 900, South Tower
San Francisco, CA 94107
1-415-947-6000

UBM TechWeb

IN THIS ISSUE

Dr.Dobb's Business Contacts

DR. DOBB'S

Sales Director, Michele Hurabiell
(415) 378-3540, mhurabiell@techweb.com

Account Executive, Shaina Guttman
(212) 600-3163, sguttman@techweb.com

INFORMATIONWEEK BUSINESS TECHNOLOGY NETWORK

CMO, Scott Vaughan
(949) 223-3662, svaughan@techweb.com

VP of Group Sales, InformationWeek Business Technology Network, Martha Schwartz
(212) 600-3015, mschwartz@techweb.com

Sales Assistant, Group Sales, Kelly Glass
(212) 600-3327, kglass@techweb.com

Publisher's Assistant, Esther Rodriguez
(949) 223-3656, erodriguez@techweb.com

SALES CONTACTS—WEST

Western U.S. (Pacific and Mountain states) and Western Canada (British Columbia, Alberta)

Western Regional Director, Matt Stovall
(415) 947-6245, mstovall@techweb.com

District Sales Manager, Rachel Calderon
(516) 562-5338, rcalderon@techweb.com

Account Manager, Alison Rubino
(415) 947-6248, arubino@techweb.com

Inside Sales Manager, Vesna Beso
(415) 947-6104, vbeso@techweb.com

Sales Assistant, Ian Doyle
(415) 947-6105, idoyle@techweb.com

Strategic Accounts

Account Director, Sandra Kupiec
(415) 947-6922, skupiec@techweb.com

Account Manager, Shoshana Freisinger
(415) 947-6349, sfreisinger@techweb.com

Sales Assistant, Matthew Cohen-Meyer
(415) 947-6214, mmeyer@techweb.com

SALES CONTACTS—EAST

Midwest, South, Northeast U.S. and Eastern Canada (Saskatchewan, Ontario, Quebec, New Brunswick)

District Manager, Jenny Hanna
(516) 562-5116, jhanna@techweb.com

District Manager, Michael Greenhut
(516) 562-5044, mgreenhut@techweb.com

Account Manager, Cori Gordon
(516) 562-5181, cgordon@techweb.com

Inside Sales Manager East, Ray Capitelli
(212) 600-3045, rcapitelli@techweb.com

Sales Assistant, Elyse Cowen
(212) 600-3051, ecowen@techweb.com

Strategic Accounts

District Manager, Mary Hyland
(516) 562-5120, mhyland@techweb.com

Account Manager, Tara Bradeen
(212) 600-3387, tbradeen@techweb.com

Account Manager, Jennifer Gambino
(516) 562-5651, jgambino@techweb.com

Sales Assistant, Kathleen Jurina
(212) 600-3170, kjurina@techweb.com

SALES CONTACTS—NATIONAL

Global CIO, InformationWeek 500 Conference

National Account Manager, Amy Neidlinger
(212) 600-3163, aneidlinger@techweb.com

Dr. Dobb's

Sales Director, Michele Hurabiell
(415) 378-3540, mhurabiell@techweb.com

Account Executive, Shaina Guttman
(212) 600-3163, sguttman@techweb.com

SALES CONTACTS—EVENTS

Director, Event Operations, Jennifer Russo
(516) 562-5094, jrusso@techweb.com

MARKETING

VP, Marketing, Winnie Ng-Schuchman
(631) 406-6507, wng@techweb.com

Director of Marketing, Sherbrooke Balser
(949) 223-3605, sbalser@techweb.com

Marketing Manager, Monique Luttrell
(949) 223-3609, mluttrell@techweb.com

AUDIENCE DEVELOPMENT

Director, Karen McAleer
(516) 562-7833, kmcaleer@techweb.com

BUSINESS OFFICE

General Manager, Marian Dujmovits

United Business Media LLC
600 Community Drive
Manhasset, N.Y. 11030 (516) 562-5000

UBM TECHWEB

Tony L. Uphoff CEO

John Dennehy CFO

David Michael CIO

Bob Evans Sr. VP and Global CIO Director

Joseph Braue Sr. VP, Light Reading Communications Network

Scott Vaughan CMO

Ed Grossman Executive Vice President, InformationWeek Business Technology Network

John Ecke VP and Group Publisher, Financial Technology Network, InformationWeek Government, InformationWeek Healthcare

Martha Schwartz VP, Group Sales, InformationWeek Business Technology Network

Beth Rivera Senior VP, Human Resources

David Berlind Chief Content Officer, TechWeb, and Editor in Chief, TechWeb.com

Fritz Nelson VP, Editorial Director, InformationWeek Business Technology Network, and Executive Producer, TechWeb TV

UNITED BUSINESS MEDIA LLC

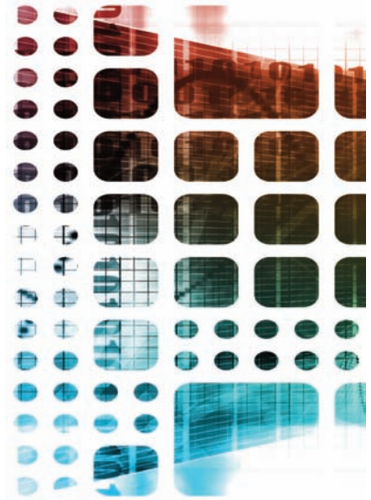
Pat Nohilly Sr. VP, Strategic Development and Business Admin.

Marie Myers Sr. VP, Manufacturing

Dr. Dobb's Journal

CONTENTS

February 2011



COVER STORY

PLUS

More on DrDobbs.com

The Rise And Fall of Languages in 2010
Python and Objective-C surge, VB falls off the cliff, and Ruby treads water. What else happened?
www.drdobbs.com/windows/229100358

Certified ScrumMaster Examined
Taking certain "certification" to task
www.drdobbs.com/architecture-and-design/229000996

NetBeans and Ruby Part Ways
Is the break overdue, or is it a symptom of Oracle's cavalier view of open-source projects?
www.drdobbs.com/open-source/229200212

Cloud Computing: Data Services, NoSQL and SQL Databases
There have been announcements of new service offerings, new platforms, and new concerns about architectural issues that introduce security vulnerabilities.
www.drdobbs.com/blog/archives/2011/02/cloud_computing_2.html

Shades of Gray
One of the hardest fields related to computers is knowledge capture. That's the art of getting the knowledge about something out of someone's head for incorporation into something like an expert system.
www.drdobbs.com/blog/archives/2011/01/shades_of_gray.html

Ajax Calls in jQuery
A handy way to control all aspects of your web requests.
http://www.drdobbs.com/java/229000921