

Accelerate Web Application Development with Application Generation Software

A Guide to Developing and Deploying Web Applications and Web Services In Record Time with Iron Speed

*Iron Speed White Paper
August 2002*



editor@ironspeed.com
<http://www.ironspeed.com>
650.215.2220

Table of Contents

Table of Contents	2
The Next Frontier: Moving Applications to the Web.	3
The Challenges of Web Application Development	4
Three-Tiered Architecture Complexity.....	5
User Interface Complexity	6
Database Management Complexity	7
Scalability Complexity	8
Security Complexity	9
The Emergence of Standards Makes Application Generation Possible.....	10
User Interface Standardization:	10
Application Integration Standardization:	10
Database Standardization:.....	11
The Evolution of Application Generation	12
The 1980's: IDEs.....	12
The 1990's: Application Servers.....	12
Today: Web Application Generators.....	13
Application Generation Turbo-Charges Web Application Development	15
Gather User Feedback Early in the Cycle	16
Generate User Interface Pages and Code	17
Generate Applications, Not Prototypes	17
Use Business Analysts to Gather Feedback	18
Simplify Software Maintenance	18
Custom-build the Unique Aspects.....	19
The Iron Speed Approach.....	20
Iron Speed Accelerates Development By Generating Application Code	21
Iron Speed's Application Generation System.....	22
Contact Iron Speed	24

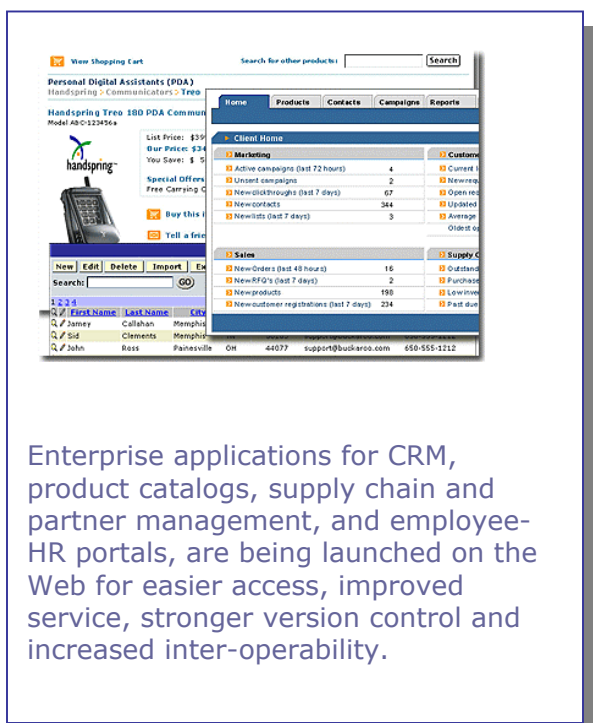
The Next Frontier: Moving Applications to the Web.

Having spent thousands of engineering man-hours figuring out how to build powerful new web-based applications as well as migrate existing client-server applications to the web, we are not afraid to speak the truth. The fact is that building web applications is harder and more complex than building client-server applications.

The user interface includes options for states you can't control. And, the stateless client is far removed from the database, so each transaction is a distinct request. Additionally, web applications must always also include provisions for database connection pooling, session management, scalability and authentication.

The benefits we've all heard relentlessly touted about web applications – those of simpler deployment, improved inter-operability and easier integration -- are so far removed from the early planning and development tasks that they hardly seem to matter. Really, who is thinking about the benefits of web services, SOAP and XML when he's struggling to correct the code surrounding the 50th SQL query in a multi-tier application?

There is a better way. A new category of software that we call **Application Generation** software generates the majority of an application's code faster and more efficiently than traditional hand-coding approaches. But, taking advantage of this opportunity means shedding the traditional client-server development approach in favor of a web-based application development strategy.

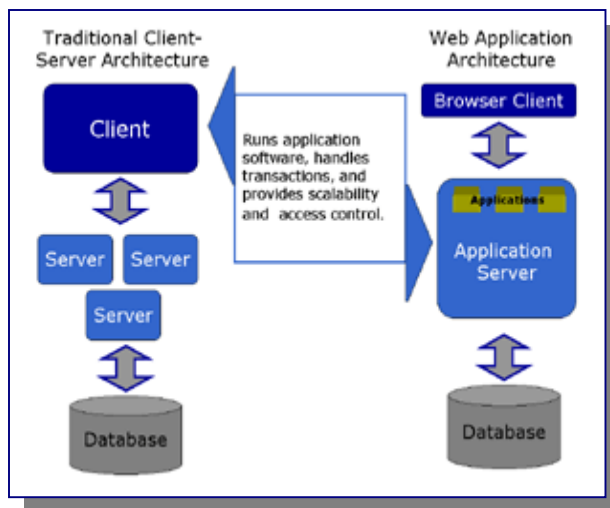


Enterprise applications for CRM, product catalogs, supply chain and partner management, and employee-HR portals, are being launched on the Web for easier access, improved service, stronger version control and increased inter-operability.

This White Paper discusses the challenges as well as possibilities of building web applications; explores the emerging category of application generators; and introduces the breakthrough methodology for accelerating application development invented by Iron Speed.

The Challenges of Web Application Development

Every corporate IT developer is familiar with client-server application development. The two-tiered architecture typically includes a thick client and a thick server that run in a controlled environment.



Developing with client-server architecture gives you lots of freedom to place heavy burdens on the client side. There is no need to ping the database for every transaction, as you can house functionality and data in the client itself, greatly reducing the need to scale the server-side software and database.

Enter now the challenge of building applications for the web, or migrating your legacy systems over to the web. At first blush, web-based applications sound great because they save on deployment and training. However between now and deployment is a whole lot of development work that adds several layers of complexity to your application development cycles.

Web application development puts the developer in a completely different development scenario:

- Web applications are fundamentally different. They are three tiered, with a thin browser client, a robust application server and a database.
- HTML is a thin client so most of the web application heavy lifting has to happen at the application server level, including the application workflow.
- HTML-based user interfaces lack the robustness of desktop windowing environments, like Microsoft Windows and the Apple Macintosh, making user interface development more difficult.
- Web applications are stateless, so they require complex session management in order to implement complex transactions. The HTML client is really designed for display (and not transactions), so the application server layer must

pick up the bulk of the coordination and management of database-driven transactions.

- Due to their reliance on standard protocols, web applications are frequently exposed to the public Internet, and so must accommodate tight security and access control.
- Although web applications are intended to operate as stand-alone systems, they frequently rely on the operation and connection with other web applications and web services. Ultimately, some of those inter-connections may be between enterprises, and lose the protection of the network firewall.

That is a very long list of challenges, even when building with a favorite programming language like Visual Basic, C++ or Java. Comparing client-server applications to web applications and web services is like saying a Morris Mini and a BMW are both cars. Both get you where you need to go, but lift up the hood and one is clearly more complicated than the other. Web applications are more complex to develop than client-server applications because they include more infrastructure, more features, more connections, and a more complex run-time environment.

Application generation technology speaks directly to this point: building web applications from the bare metal is complex and time-consuming. Accelerating the development cycle without sacrificing performance, features and interoperability are the key metrics by which application generation is being measured.

The **three-tiered architecture** of a web application adds challenges for developers in several areas:

- User interface
- Application Infrastructure
- Database
- Management
- Scalability
- Security

Three-Tiered Architecture Complexity

Specifically, the three-tiered architecture creates new challenges for developers in several areas: creating the web-based user interface, providing application infrastructure, database management, scalability and security.

Traditional client server applications didn't have to worry about those things nearly to the same level as web applications.

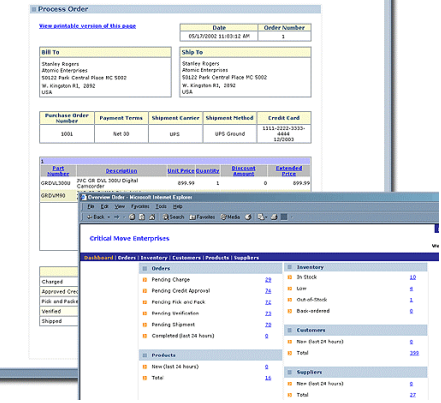
The Browser Client Tier: HTML is really designed to display content and is not robust enough to handle complex enterprise transaction management. Java scripting, rich media and active server/java server pages bring the language forward toward simple interactivity, but clearly HTML is a long way from the robust, controlled, desktop-windowing environment of client-server applications. This forces all the coding and interactions

that happen at the client level in traditional applications further down your application architecture.

The Application Server: Much of the functionality found in the client layer of a client-server application is now pushed to the application server level for web applications – an expanded role for the traditional server tier that now sits between the thin HTML client (not doing much more than displaying the user interface) and the database. The new application server tier literally powers each web application and web service.

Back in the old days, this complete infrastructure could be built independently and from scratch for each application. Of course we didn't know to call it an "app server," but we would build a foundation layer that supported other functionality. While an application server provides some of the things you need, it isn't central to your development, and so it can't take you quite far enough for your web-based application needs.

The Database: Web-based applications separate the database completely from the server layer in order to allow continuous and non-synchronized requests to be transferred through the new Application Server middle tier.



Users expect a certain threshold of functionality and user interface look and feel. Appreciated features like navigation, search, pagination and table sorting can each take weeks to develop.

In a "stateless" environment like HTTP, each separate transaction must log in and out of the database. As many web applications are outward-facing toward customers, employees, partners or suppliers, a "transaction" now includes everything from confirmation page views to orders, requests and bids. Thus, the load on the database gets pretty big pretty quickly. The emergence of what we today call "application servers" was a direct effort to mitigate this need for database connection pooling

User Interface Complexity

Hand-in-hand with the complexity of stateless transaction processing is the user interface development.

HTML is a comparatively low functionality user interface relative to a Windows or Macintosh environment. That "glossy magazine" feel of an HTML user interface is misleading. While HTML is great for presenting static pages, the rich interaction that we desire in

Accelerate Application Development
using Application Generation Technology

Iron Speed, Inc., 1953 Landings Drive, Mountain View, California, 94043, USA
www.ironspeed.com

enterprise applications is missing. Additionally, and perhaps as a result of that glossy look and feel, there is added pressure on transaction-based applications to be more user-friendly and forgiving.

Consider the difference between the web and a Microsoft Windows application like Excel or Outlook.

In Windows applications, multi-step processes can be performed within a single application screen. For instance, you can drag-and-drop an email from one folder to another. This functionality is not possible in HTML, nor can complexity be structured as conveniently as in a controlled client-server application. In Windows, complex functionality like "find" and "spell check" is handled with a pop-up window that keeps the original screen in view. That sort of functionality has to be handled on the web via a series of static pages that only mimic what is a fluid and simple user experience on the desktop. Although the web excels at lower volume, lower complexity transactions, those are not always the types of transactions required by high-use enterprise-class applications.

HTML falls short when you try to offer the selection choice of anything more complicated than a 15-field drop down menu. Seemingly simple features that are commonplace on most web applications can take weeks to develop – search, table sorting, pagination and hierarchical navigation. Those features are more than "nice to have," they are expected.

Your users don't see the complexity of your application code; they only want a premier experience. They want to change their own account information; change their mind half-way through a transaction and save only the relevant fields; ensure one customer is forbidden from seeing another customer's data -- and they want all the pages to load in under three seconds.

Database Management Complexity

In a web application, database management concerns manifest in two key areas: user session management and database connection pooling. Since the very nature of a web application is stateless, there is a great need to manage transaction integrity and completion. Web applications require an environment that supports the creation and management of processes built from these stateless transactions.

In traditional client-server applications, a "session" was straightforward: the user logged in, stayed in, and then logged

out when they closed the application. The entire “session” logged in once to the database. In the stateless environment of HTTP, there is no such thing as a controlled “session.” A web browser client can’t tell when a user is logged out once the transaction has begun. In essence, every transaction – even in a multi-transaction process – is potentially a unique session.

A web browser client can’t tell when a user is logged out once the transaction has begun. In essence, every transaction – even in a multi-transaction process – is potentially a unique session.

A new way of managing the database connections is obviously required to support this stateless session environment. Web application servers handle this by maintaining a pool of database connections – keeping open a number of connections and re-using them for each transaction request. This way requests from multiple users as well as multiple requests from individual users – regardless of the order they are received – can be handled without repeatedly logging in and logging out of the database.

Functions to address these issues are best centralized and managed as a separate collection of functions on their own machines. An application server middle tier is a second-generation approach, recognizing that much more support is needed than with client server applications.

Scalability Complexity

Since web applications are intended to scale to enterprise or even inter-enterprise levels, scalability must be ensured.

In client-server applications, application scalability is managed primarily through the client. Each end-user ran the client-side application on a desktop machine, so no extra computing horsepower was needed. And since so much of the application functionality was fully contained in the client side, there wasn’t as much concern about scaling applications

In web applications, the entire application functionality is concentrated in the new middle layer, running on the application server. This application server gives you the scalability you need to have one database support multiple applications and multiple servers. Now, the application code has to scale as well as support multiple users. Fiscally, web applications are more expensive, since application server machines and software are now required to run them. None of the application functionality runs on the desktop machine any longer.

There is a benefit, however. The strategy of managing scalability at the application server level lets you enable multiple server machines to run your applications and process transactions against a single, centralized database (the database tier). It's much more difficult to create a multi-machine distributed database architecture in a client-server development, so creating scalability at the application server tier erases that concern.

Security Complexity

Running on the public Internet means that by nature your applications – and the underlying databases that drive them -- are exposed to increased levels of risk. This requires increased attention and support for security. Even intranet-only applications run on standard IP protocols, which are easier to hack. Security is a top of mind concern at every step of web-based application development and deployment.

Multiple layers of protection are needed from both the application itself in the form of authentication and user access control, as well as from your network firewall and other security measures.

The Emergence of Standards Makes Application Generation Possible

How can IT departments build these increasingly complex web-based applications with the increasingly pressured and limited resources of today?

The answer is Application Generation.

And what makes application generation possible now when so many attempts in the past have not succeeded?

The answer is Standards.

While applications used to have to accommodate so many different platforms, today, standardization is appearing in many areas. This makes it easier to generate substantial portions of an application.

User Interface Standardization:

During the last ten years, the end-user interface environment has become standardized with HTML web browsers becoming the client interface of choice. For desktop applications, Windows remains the de facto choice for corporate environments. Standardizing the user interface environment makes it easier to generate user interface code that runs on almost any machine at any location, especially if it's HTML.

Moreover, most users see the web browser – with its ability to combine rich text and graphics with traditional data display and form based input -- as a more friendly, intuitive user interface than traditional desktop windowing metaphors. In fact, most of the major ERP and CRM vendors have rewritten, or are in the process of rewriting, their applications for the web. It won't be long before most other applications follow suit.

Application Integration Standardization:

More recently, SOAP and XML are becoming established as application integration standards, and make it possible to develop applications that can adapt to new business rules and connect across systems. While most application integration is deployed behind a company's firewall, web services technology offers the promise of cross-enterprise integration, tying together systems running on entirely different platforms.

Ours is now a vastly different application development world compared to traditional client-server environments. This gradual standardization of user interface environments, application integration protocols and database languages has brought us to the point where application generation technology can be built and deployed to accelerate web application development.


Database Standardization:

Most enterprise applications standardized over the last 20 years to operate on SQL-based relational databases. More recently, database middleware like ODBC and JDBC make it easier to retarget applications to different databases (e.g., Oracle and DB2) without significantly rewriting the query and access code. This type of standardization makes it possible for an application generator to target one type of relational database and generate queries in a single language (SQL).


The Evolution of Application Generation

The goal of computer-generated code has always been to do more with less and develop robust applications more quickly.

In the 1980's, Integrated Development Environments promised it. In the '90's, HTML and XML promised it. Today, in the '00's, SOAP and web services promise it.



The promise of application generation is that robust, full-featured applications can be developed rapidly and with consistently high quality.



Each of those promises got developers part of the way to the goal. Each took one segment of the development cycle, and identified the elements that were common, could be standardized, and were not custom or career-building exercises. As web services standards offer new layers of standards and applicability across many enterprise and departmental arenas, the emerging application generators of today really can deliver on the promise first proposed three decades ago: that software applications can be generated rapidly and with high quality.

The 1980's: IDEs

The first generation of "software that generates software" was the Integrated Development Environment. The code that was actually generated wasn't that significant, but it did give you a standard looking application framework. The fact that each application was "familiar" due to these shared components and carried a common look and feel was found to be tremendously valuable, and made development from enterprise to enterprise much easier. Developing with one of these IDEs became a transferable skill set, and many application developers gained experience with these tools.

The 1990's: Application Servers

Application servers became available in the late 1990s for much the same reason that IDEs had become popular and valuable: the application servers included a base level of functionality that was common for all applications and didn't need to be re-built every time. Plus, the fact that this functionality was handled in the same way for each particular application server makes application development more predictable. Again, developers gained experience with one of the vendor's products and that

experience was transferable around the enterprise and the marketplace.

This "second generation" includes components for database connection pooling and session management. Although this is a valuable contribution to reducing the development cycles and makes those components more standardized across the market, these features still have to be customized and integrated into each specific application. Additionally, the components included in most application servers don't go nearly far enough for the stateless environment of today's web applications. No application server thinks at all about non-controlled or non-sequential states within a complex end-user transaction. And, no application server accommodates the need for user interface functionality in the middle server software layer – required because HTML is a thin client.

Today: Web Application Generators

What Types of Code to Generate?

General categories of application generation for web-based applications and web services:

- User Interface Code
- Application Infrastructure
- Database Management

Now that enterprise applications are migrating to the web, the user interface, application infrastructure and database management functionality becomes a prime candidate for application generation for the same reasons the past attempts at application generation become popular in their day: these components can be easily standardized and building them by hand doesn't necessarily build your career.

Building a search engine or navigation into your application is usually a one-time job, and it's not all that interesting or career building as other custom projects. So, having it all generated for you is of pretty high value.

One of the reasons application generation is so important and timely now is because of the near universality of HTML and XML and the expectation that SOAP will achieve a similar status. As a concept, application generation works best when you can generate software that runs anywhere in any environment. Application Generators (like Iron Speed) build in many features that you might not have the time to do by hand because you have an application backlog to work down and deadlines to meet.

Several products claiming to be Application Generation technologies or application assembly tools are in market and each offers different features and functionality. The tools vary in depth and breadth of application code generated, and so does the amount of time and complexity it takes to reach a point where the generated software leaves off and the rest of the application must be developed by hand.

A thorough review of the Application Generation options should consider these three general categories of application generation that focus on the routine or common elements found in most web-based applications and web services. Specifically these are web-based user interface and user interface code, workflow and SQL queries, and the database schema and database access code.

- **User Interface Functionality and Features:** Web-based user interface pages, user interface code, search, navigation, pagination, filtering, sorting and reports.
- **Complex Application Infrastructure:** Workflow, business rules, SQL queries, session management, access control and authentication.
- **Database Management:** Database access code, database schema, database connection pooling and data migration.

By generating some or all of these elements, you quickly add a level of robustness to the application that makes it more complete and powerful, and yet still lets you focus on the more interesting and important work that is specific to your business needs.

While hand-built code would surely get the job done, it's not the most interesting or signature components of an application, nor is it the most efficient use of developer resources.

Application Generators like Iron Speed provide deep functionality "for free":

- Generate the routine program code and provide a predictable base for all your web-based applications;
- Generate code of consistently high quality and performance and so you have a jump-start on your

development and can predict how the foundation of each application will behave;

- Allow developers to focus on developing the unique aspects of their applications, which typically involves proprietary and career-building development work.

Application Generation Turbo-Charges Web Application Development

How Application Generation Turbo-Charges Web Application Development:

- Generate full-featured, running applications, not prototypes;
- Generate User Interface pages and code;
- Use business analysts to gather end-user feedback;
- Gather user feedback early in the development cycle by leveraging application generation.

Since web applications are so much more complex to build, it is imperative that the design be flawless when the developer sits down to begin coding. Spending three weeks adding functionality to your application only to find out that the user community wanted something outside the specification is not productive or enjoyable (for anyone). Yet, it is nearly impossible to predict user preferences before users begin to use an application, even if you have a very detailed requirements specification document. Inevitably it seems, the directive to be able to search in a product catalog application against the notes, not just the product description, comes **after** the three-week investment to build the search feature against the original specifications, which did not include that functionality.

In the days of client server applications, some of this frustration was accepted, as users had little expectation that any application would be familiar to them the first time they opened it. Application training was an expected and required condition of use. This was acceptable since applications were built for relatively small and controlled groups of users – i.e.: the customer service team or the sales team.

Web application development opens new challenges in both areas. Users do expect web applications to be familiar – they expect the web to conform to commonly accepted standards, features and navigation. In addition, training is not really an option as users of web applications are often broad groups with dynamic memberships – i.e.: suppliers, customers and all employees.

Application Generation technology addresses this central challenge: quickly generating a working application, so user feedback can be collected in real time, and in a format that is helpful and relevant to the developer.

Gather User Feedback Early in the Cycle

Application Generation technology goes one significant step further than Rapid Application Development (RAD) approaches of the past, including the more recent Extreme Programming trend. RAD is about putting a prototype in front of users to obtain feedback during the development cycle, not just at the end when it is too late and too complicated to make any significant changes.

RAD is really helpful, since most users can't articulate what they want in an application because they can't envision it. It's like asking someone who lives in the desert to describe the rain forest. They can use words like "cool" and "wet" but it is just not in that person's lexicon or experience to imagine the lush jungle environment.

Using Application Generation technology, developers can generate a basic, but working model of their full application, and populate it with real world data. Taking this experience to users gives deep and meaningful feedback at all levels of the application.

The screenshot displays a web application interface. On the left, a 'User Profile' form is visible, containing fields for Name, Screen Name, Login ID, Password, Role, Email, Business Phone, Fax, Company, Title, Department, Address, and Country. Below these fields are links for 'Change Password', 'Activate Contact in Lists', and 'Update User Profile'. At the bottom of the form, there is an 'Email Bounce Back(s)' field and a section for 'Created At', 'Created By', 'Last Modified At', and 'Last Modified By'. An 'OK' button is located at the bottom of the form. Overlaid on the right side of the form is a 'Send Password' dialog box. The dialog box contains a message: 'The password for user name 'demo' has been successfully sent to that user's Email address.' and an 'OK' button.

User interface pages that reflect important workflow sequences are created along with the rest of the user interface pages when using Application Generated technology. This helps developers and end-users give relevant feedback early in the development cycle.

Generate User Interface Pages and Code

In web-based applications, the user interface is significantly more important than in client server applications. The web is already familiar to corporate end-users, any web-based application is perceived to be easier to use.


In fact, the user interface has become the bell weather for application robustness. Even experienced developers (who might know better!) look at a user interface and make judgments about the complexity and depth of the underlying application. The user interface sends a more powerful message about the application than the performance and the architecture.

However, most application developers are not trained to build highly usable and interactive web-based interfaces. While large commercial e-commerce and corporate sites have teams of people designing the web pages, most IT departments don't have this luxury.

Application Generation technology automatically generates the web-based user interface pages and the underlying user interface code. Moreover, they do so using contemporary user interface standards, making most generated applications more user-friendly than those built by hand.

Included are not just the top-level pages and reports, but all the routine account information and data management pages. Real data can be imported so end-user testers can offer feedback based on actual scenarios. So, application developers do not need specific graphics training to generate attractive and compelling graphical interface pages.


Generate Applications, Not Prototypes



The user interface has become the bell weather for application robustness. The **user interface sends just as powerful a message** about the application as the performance and the architecture.

Typically, development requires some working model of the application in order to gain relevant and helpful feedback. The RAD approach prototypes the application in order to obtain some of this feedback. But RAD doesn't usually build much of the final application.

the user community requesting an application from tment will employ a graphic artist to develop the web-based interface in order to guide the discussion with the



Accelerate Application Development
using Application Generation Technology

Iron Speed, Inc., 1953 Landings Drive, Mountain View, California, 94043, USA
www.ironspeed.com

development team. This can be helpful, but this tactic usually involves designing about half a dozen of the top-level screens. However, most web applications have hundreds or even thousands of screens. Some of the really important workflow screens – like account maintenance pages and data importing wizards – get left out of this process. And yet, often those workflow experiences represent the kind of feedback developers need most.

Use Business Analysts to Gather Feedback


Another benefit of Application Generators is that they allow business analysts, program managers and information architects to generate a basic working application. This differs considerably from RAD and extreme programming methodologies that require developers to create prototypes before end-user feedback can be gathered.

Typically, these members of the IT department are specifically trained and experienced at gathering feedback from end-users. Application Generation technology gives these non-developer members of the IT department an expanded role in the development cycle, freeing developers to focus on the aspects of the application which are not generated, and require custom programming.

Simplify Software Maintenance

As soon as web applications are launched, requests for modifications start rolling in. Sometimes these are issues end-users did not anticipate; sometimes they reflect changes in company strategy or market conditions that were unanticipated. However, sometimes user interfaces change because of advances in the other websites. As certain features become commonplace, every web application is expected to keep pace. Why is it that web-based portals change their look and feel continuously and yet "Time" magazine does not? The web makes everything more accessible, and increases the competitive pressure for pushing more, rather than less, out toward the user community.

This pressure to constantly evolve and improve lands squarely on the IT department and makes application maintenance more significant than with client-server applications. The concept of



IT departments are pressured to keep web applications up to date with corporate and user demands as well as market advances. **The concept of RAM – rapid application maintenance** – becomes more important as applications must be dynamic and responsive systems.

RAM – rapid application maintenance – was never a part of the application development tool set.

There is nothing worse than trying to guess what “the other guy” was thinking when he wrote the original code. Typically, maintenance is a developer’s least favorite task. **Application Generation solves the RAM problem.** Application Generators fully re-generate the entire code base with each modification. The generated code keeps up with each new revision and each new management order. So finally, no one has to go back into the code to re-figure someone else’s work.

Custom-build the Unique Aspects

Application generation is not a silver bullet. Application Generation software can only hope to generate about 80% of any enterprise application. The remaining 20% contains unique algorithms and business logic and must be developed by hand. It’s important that any Application Generation software offer convenient and comfortable interfaces for developing and integrating this remaining, but crucial, 20%.

There are two common types of integration that must be considered when using Application Generation software: integrating business logic and communicating with external applications.

Integrating business logic requires a close coupling between the 80% that is generated and the 20% that is built by hand. This is best accomplished by a set of API’s, or set of objects, for connecting your code with the generated code. Ideally, this integration is persistent from regeneration to regeneration, meaning that your integration work is not lost each time the application is regenerated by the Application Generator. If it’s not, then you can lose much of the benefit of using Application Generation software in the first place.

Integrating external applications, such as a third-party CRM package, is best handled using application integration protocols. Increasingly, web services protocols like SOAP are being used for external integration, and ideally, your Application Generator supports these web service protocols as well.

The Iron Speed Approach

Iron Speed recognizes that of the tens or hundreds of thousands of lines of code that make up every web-based application, very few of the aspects are truly “unique.” Of course, the couple thousand lines that are unique are what make an application and company successful. But from a development point of view, the vast majority of the code in a web application is not unique.

The Key To Application Success:

Of the tens or hundreds of thousands of lines of code in every web application or web service, very few are actually “unique.”

While this relatively small percentage of unique aspects is the key to making each application successful, developing the non-unique code typically takes up 80% of the development time and energy in traditional hand-coding approaches.

Application Generators let developers focus the majority of their time on the 20% of the application that is truly unique.

Particularly in this era where IT departments are being increasingly challenged to provide higher levels of business agility and connectivity, web applications built using application generation offer the advantage of modularity and inter-operability

To that end, Iron Speed takes what we call an “Atomic Parts” approach to the challenge of web application development. Most transaction-based applications can be broken down into their “atomic parts” – with many of those parts being core and common elements across many applications. As with atoms, each element on its own can be identified with certain properties, but it’s not until you start to put these elements together that things get interesting. So too does Iron Speed break down the parts that are common to all web applications, generate them with consistently high quality, and then let developers focus on the interesting business rules that are truly unique to each application.

The Application Generator generates aspects of applications based on two filters. The first filter is standardization: which elements can be standardized so they can support any application, any system and any enterprise?

The next filter is uniqueness – if it is not unique, generate it! Those base elements that can be very time consuming to build but are common across all web-based applications are prime candidates for application generation rather than hand coding. Every developer would concur that time consuming does not always equate to fascinating work.

Iron Speed Accelerates Development By Generating Application Code

The Iron Speed “Top Ten Least Favorite Late Night Development Tasks:”

1. User interface web pages
2. Database normalization
3. Wrapping display code around SQL queries
4. Data migration
5. Writing SQL queries
6. Integrating with other applications
7. Database connection pooling
8. Building full-text search features
9. Creating user authentication protocols
10. Building with non-standard languages or protocols

These and others must be built for each application, but are not necessarily the most interesting or career-building tasks. This makes them prime candidates for application generation.

Iron Speed automatically generates up to 80% of any enterprise application, including the web-based user interface and user interface code, workflow and SQL queries, and database access code. This frees developers to focus on the other 20% of the application that is truly unique and proprietary.

We call the Iron Speed Platform an Application Generator to reflect our role in the application development cycle. Building applications using Iron Speed condenses the design and development stages and jump-starts the development cycle.

Especially in multi-developer environments, it is helpful to develop quality, consistent code foundations for all web applications without doing the programming by hand. For instance, if you need to change a data field or adjust the workflow based on new business rules, then Iron Speed generates the code, does the data migration in the production database, and produces a new application for you.

Unlike art where the thrill is in the creation, application development is more like industrial construction. Most developers get the thrill from the sense of accomplishment. The most

interesting and career-building aspects of each application are those that are unique to the application itself. Using the Application Generator technology from Iron Speed, it can take you a fraction of the time to get to that same thrill of accomplishment.

Iron Speed's Application Generation System

Iron Speed generates a basic, working application in just a few hours, freeing developers to focus on implementing the application's business rules and not the application infrastructure. This breakthrough methodology for accelerated web application development both relieves the time-to-market pressures on IT departments as well as allows more efficient development resource allocation.

Applications built with Iron Speed software include many features often cut to meet deadlines when applications are built by hand:



Iron Speed Designer is the leading Application Generation tool: A web-based development environment for the rapid development of robust enterprise web applications and services. Applications built with Iron Speed Designer currently run on the Microsoft .NET web services platform.

➤ **User Interface Code:**

Web-based user interface, including web-based screens and data input pages. Advanced features include search, navigation, data validation, table pagination, dashboard pages, import/export wizards, display tables and reports with filtering and sorting.

➤ **Application Infrastructure:**

Workflow logic and the SQL queries, session management, database access code, user access control and authentication, security, caching, complex data types, email notification and audit trails.

➤ **Database Management:**

Database schema, database connection pooling, record locking and transaction management. Automatic data migration keeps the database synchronized with each maintenance release.

In addition to generating up to 80% of the application code including complex features like database connection pooling and session management, Iron Speed also offers developers considerable advantages that accelerate the development cycle and improve IT productivity – all without sacrificing features, functionality or choice of programming language.

➤ **Speed development and reduce testing time.**

Iron Speed automates application development for any web-based application, saving days or even weeks in the development cycle. Plus, code generated by Iron Speed is

consistently of high quality, releasing you from many routine QA and testing cycles.

- **Generate standard code.** The code Iron Speed generates is specifically targeted to certain platforms so there is no new program language to learn. Currently, Iron Speed generates native PL/SQL, native ASPX and other Oracle and .NET standard code.
- **Focus developers on highest priority tasks.** Since Iron Speed generates about 80% of the application code, developers are free to focus on the other 20% of the application that is unique and proprietary.
- **Open development to new categories of developers.** Since you don't need experience with Visual Basic, C++ or Java, Iron Speed allows new categories of developers such as business analysts and information architects.
- **Keep development ownership in-house.** Since Application Generation speeds development and lets you keep the entire application development project in-house, it compares very favorably to the only alternative: outside professional services. Outsourcing is often impractical or overkill for application development, and can be very costly both in terms of fees and management time and attention.
- **Boost your company ROI.** Deploying applications faster means they start working for your company sooner. Calculate your ROI based on new revenue, improved productivity and streamlined operations.

##



Contact Iron Speed

We welcome feedback and commentary on this white paper and invite you to visit our website or call us for more information.

info@ironspeed.com
<http://www.ironspeed.com>
650.215 2200

Let us help you get started! Call or email us for a Live Demonstration of the Application Generation technology from Iron Speed, and build a sample application with us.

sales@ironspeed.com
<http://www.ironspeed.com>
650.215 2200

Accelerate Application Development
using Application Generation Technology

Iron Speed, Inc., 1953 Landings Drive, Mountain View, California, 94043, USA
www.ironspeed.com