embarcadero | Whitepaper

RAD STUDIO

GUIDE FOR MANAGERS

by Stephen Ball - April 2021

TABLE OF CONTENTS

Introduction

The world of software development thrives on innovative concepts like Object Orientated Programming (OOP), Agile Development, Continuous Integration (CI), DevOps, Low-Code, Enterprise & Micro Services, UI / UX design, and many more.

There is arguably one key concept behind many of these buzzwords, one that has seen a huge resurgence in recent times. It continues to heavily influence the modern tooling and processes used in software development today, and is now being claimed by a broader set of products covering a host of innovative approaches. That term is **Rapid Application Development (RAD)**.

With big software firms like Microsoft, Google, Apple, Amazon and Salesforce, to name just a few, all evangelising RAD approaches, this paper will explore how RAD is evolving, and how **RAD Studio®** today continues to innovate tooling and frameworks and supports the latest development practices and protocols.

Who Should Read This Paper

This paper will have a broad appeal to CTOs and leaders of software development teams, and is written for those following or evaluating market trends and possible solutions to use for both desktop and mobile applications.

If you have legacy **Delphi®** or **C++Builder®** code, this paper is especially important as it will help you understand how coding has evolved, and evaluate and harness the value in your existing codebase.

RAD Studio® in The Evolution of Software Development

From RAD and ASD to Agile

Rapid Application Development has seen continuous innovation and a resurgence recently. RAD has evolved a long way since its popularity exploded when groundbreaking developer tools like **Delphi®** and **Visual Studio** were launched in 1995 and 1997. These RAD tools, combined with **Adaptive Software Development (ASD)** practices that became popular at the time, enabled the acceleration of **Business Process Re-Engineering (BPR)** that coincided with Microsoft Windows' domination of the PC market.

Early exponents of RAD adopted ASD to benefit from shorter product life cycles, with early feedback from customers based on prototypes that reduced risk to overall delivery and enabled early versions to return value sooner into businesses that used them. The evolution of RAD-based development processes over the last 25 years, and the complementary assets created, for example, around software design, testing and source code control, has largely culminated in what we know today as **Agile Development**.

Market Trends Impacting Software Development

Alongside the evolution of development processes, the field of software development has navigated tectonic changes in the last 15 years.

Here are a few of them:

- New mobile platforms and hardware have changed the way we access and process information and collaborate
- Mobile has overtaken desktop in terms of Internet-connected devices
- App stores dominate the market for mobile application deployment

- 32-bit is giving way to 64-bit as both the Apple App Store and Google Play have switched to 64-bit-only for new apps
- Bring Your Own Device (BYOD) policies have brought about a shift in device ownership and data access. This has introduced new deployment and security challenges due to a wider mix of operating systems and devices (and their varied capabilities) that need to be addressed by developers.
- PaaS (Platform-as-a-Service) has enabled a new rethinking of hardware ownership and fault tolerance issues
- PaaS has also enabled more stable SaaS offerings with low-recurring-cost models
- Users today expect systems to have high interoperability. The drive towards Open Innovation is such that API's are expected to coexist or integrate with other systems enterprise customers have in place.
- Microservices have grown as a way to rapidly enable new capabilities within systems (e.g. language translation services, push notifications, weather data, etc.), with HTTPS as the default transport mechanism and JSON (first standardised in 2013) as the default data structure
- Containerization and DevOps have changed established approaches to the development and deployment of software, leading to rapid and continuous deployment models
- Data storage has exploded with the onset of NoSQL and traditional SQL data storage powering systems. As a result, data is often referred to as the new oil.

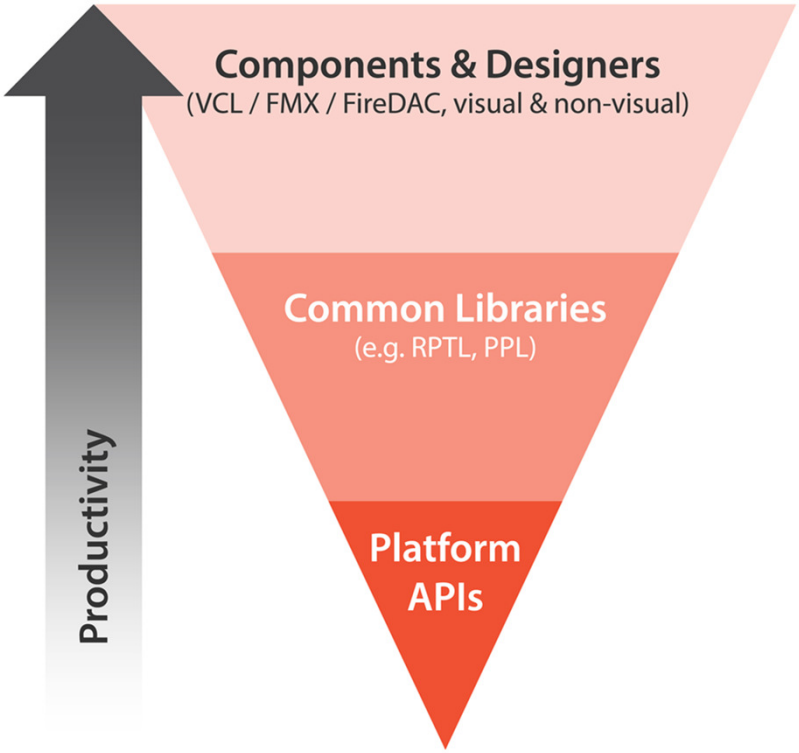
As this overview of market trends highlights, the environment for development today is a long way from RAD's first emergence in the era when Windows dominated requirements.

RAD Studio® Today

RAD Studio® uses a single code base that compiles to true native code on Windows, Linux, macOS, iOS and Android. This unique approach provides both speed and high flexibility in development, and rapid runtime speed. Time and again, the **RAD Studio®** approach proves to be **5x faster** than other languages and frameworks. With its single code base, **RAD Studio®** excels in ensuring faster times to market across all platforms through shorter development and testing times.

The original designs for the libraries and components in **Delphi®** have enabled **RAD Studio®** to provide long-term code investment security to developers for over 25 years. Indeed, the patterns are so strong they are easily identifiable in many languages and frameworks (like C# and .Net) that have followed since.

If you are not familiar with this approach, then suffice to say the true value components bring to RAD development is the abstraction of platform APIs into easy-to-use building blocks that take care of the low-level system calls. These components are underpinned by libraries that do the heavy lifting of enabling reusable core functionality (such as runtime libraries for file system, date time, screen info, etc.).



As market innovations and trends have emerged, the component and object orientated approach used in **RAD Studio®**, strongly supported by interfaces, has enabled the introduction of new features, platforms and capabilities, all while maintaining a high level of backward compatibility.

Importance of Interfaces

One key contributor to the unique success of **RAD Studio®** in creating **true native code across multiple platforms** is the way interfaces are implemented in **Delphi®**.

The way a steering wheel can be used to steer a variety of different vehicles is an effective metaphor. If you know how it works, you can use the same tool to drive a truck or boat as easily as you drive a car.

In other words, the interface is the same for the driver, but what happens under the hood can be radically different. In the same way, developers can write code that checks for and employs common interfaces that are then provided with platform-specific implementations.

Now let's see how **RAD Studio®** has enabled developers to continuously ride the waves of innovation in both processes and platforms through examples over time.

Process-Centric Innovation - The Example of Unit Testing

Unit testing enables continuous testing of code programmatically to help reduce regression issues and eliminate bugs from the code base. Through open-source projects that have integrated with the **Open Tools API** of the **RAD Studio®** IDE, developers the world over have been able to see how their code performs against tests in real time. Unit testing can also be run as part of a continuous build process, with the results going back into the external systems.

The introduction of unit testing around the year 2000 also encouraged a number of developers over time to adopt different design patterns such as **Model View Controller (MVC)** and **Model View-View Model (MVVM)** to enable easier testing for the logical parts of their systems. While these approaches haven't forced immediate changes to existing code, IDEs, languages or frameworks, they have driven the evolution of how code is written and also seen the introduction of additional frameworks such as **Spring4D** to support complimentary approaches such as dependency injection.

Product-Centric Innovation - The Power of FireMonkey

While unit testing is about how code is written and which processes are followed, other innovations require major product enhancements. September 2011 saw the introduction of **FireMonkey (FMX)**, the cross-platform framework that at first glance mirrors in many ways the **Visual Component Library (VCL)** that enables RAD development for Windows.

Under the hood of FMX, the new components, using the existing updated runtime, support cross-platform coding with a single code base that works natively across all platforms. This is no small feat, and the FMX framework has evolved over the last decade to provide the most comprehensive framework for **single-code native development** on **iOS**, **macOS**, **Android**, **Linux**, and **Windows**. This has been supported with additional compilers added to the product, along with differing build configurations to support local development and also compilation and packaging directly for app stores from inside the IDE.

Additionally, new features have been introduced, such as **Visual Live Bindings**, that enable the RAD binding of user interfaces to data and object models alike, while other features have evolved, like **FireDAC**, which enables broad database connectivity components that work cross-platform. The component approach used by **RAD Studio®** means database code that has worked on Windows for decades is now in reach of mobile platforms, enabling new life to be rapidly injected into existing code.

RAD Studio® is also using **Open Innovation** to enable fast-paced improvement. The new compilers are built using the LLVM project, providing the fastest runtime performance for each possible platform, and the IDE has recently introduced support for the **Language Server Protocol (LSP)**. This standards-based approach is also laying the foundation for a bright future.

Innovation Through Partnership

Some innovations have been realized through partnerships. **RAD Studio®** was the first tooling anywhere in the world to enable the **Microsoft Desktop Bridge**. This opened the door to the Microsoft store, and also to the latest enterprise deployment mechanisms for traditional Windows applications. Since then, support for Microsoft's new **Edge** browser has also been added prior to its official launch.

Innovation Through Acquisition

Embarcadero, now owned by **Idera Inc.**, is part of a rapidly growing technology group. This has opened the door to the adoption of many more tools to help developers work faster. Tools like **Ranorex**, **Sencha** and **Aqua Data Studio** were added to the **RAD Studio® Architect** product, enabling developers to test user interfaces with the powerful Ranorex tool, develop faster with simplified access to manage a wide range of databases using Aqua Data Studio, and bring RAD to JavaScript through Sencha Architect.

In addition, a strong third-party ecosystem of component vendors create frameworks for using **RAD Studio®** to provide the compiled backend, with the same language and code to power web applications.

RAD Studio® continues to evolve with an open framework for enhancing code and an active third-party ecosystem providing components and add-ons that support the core features of the IDE, and is now in a renewed growth stage within an exciting developer-technology-centric group.

Modernize or Rebuild?

If you have existing code built in **Delphi®** or **C++Builder®**, the first questions must be about what use case it is being used for, and whether you will still be serving that use case going forward. If you will, then there is value in the code. The next question is how to get the logic from where it is today to where it can provide future value.

The key points when assessing your next steps are:

- Working out what your future system architecture will look like, (e.g. whether you need to enable remote/mobile access)
- Identifying and managing risk and cost
- Looking at timelines and the future cost of ownership
- What phases this will require

Let's look at a few of these points now.

Looking for Safe, Quick Wins That Buy You Time

One challenge for the continued use of legacy code bases is that often, coding practices in the company have evolved a long way since the first version was released. While the code and project works, there may be parts your team would approach in a different way if it were starting fresh today.

If part of the plan is to modernise your practices with code, then the safest approach is to use a phased update of the existing code. This keeps a customer-facing product always available with the latest options, while also enabling the update towards new practices.

User Interface Testing as a Migration Approach

Returning to code testing, one common desire is to add testing to the code to ensure it tests faster and more efficiently as part of the development process. While unit testing is one part (and can easily be added for new code or new standalone functions and classes), another popular approach is to look at **User Interface Testing**. This is where **Ranorex** (part of the Architect edition) truly complements **VCL** development today, and provides an easy-to-add foundation for checking future code changes. RAD developers create UI tests for their product to check their applications prior to UA testing. Because they work on the UI layer, they leave the developer free to completely rewrite the code underneath if they so desire, and verify everything by making sure the user experience hasn't changed.

These tests are also a great way to check the software when migrating from old versions of **RAD Studio®** to newer versions. Because they work on the Windows control handle, they even allow you to rearrange the UI without breaking the tests. Furthermore, common repeatable actions (such as logging into a system) can be saved and added to multiple tests to enable extended flexibility and speed up test creation. Because the tests are built onto the product after it's created, they can also be passed to contributors outside of the development team to create, freeing up valuable resources.

Adding in The Latest Windows 10 Features

Thanks to the component design of **RAD Studio®**, it is possible to rapidly enhance any existing application's UI with a few simple steps. Uniquely, the **RAD Studio®** components also make it possible to maintain support for older versions of Windows, ensuring you don't block specific users upgrading if they are stuck on versions older than Windows 10.

One quick win in **RAD Studio®** that enables Windows 10 support for high-DPI and multiple monitors running different DPI's is the replacement of the traditional **TImageList** with a new **TVirtualImageList** and **TImageCollection**. By encapsulating the latest per-monitor high-DPI support APIs, these components provide a no-code update that can serve the correct image based on the screen resolution for each monitor.

Designed to Save Money and Time

Another big advantage of **RAD Studio®** is that it reduces manpower requirements, which means keeping fewer skill sets and code bases updated. This can be a great time and cost saver, especially when trying to build and deploy to multiple platforms at the same time. **RAD Studio®** is seeing a significant surge in interest because of this. An IoT boot camp was held in 2017 with developers from over 180 countries signing up to the week-long online conference. There are also great online resources today like **LearnDelphi.org** and the **EmbarcaderoAcademy.com**.

Although there is a large global **RAD Studio®** community, demand in certain areas can at times outstrip available developers. To cope with this, a growing number of companies are onboarding and upskilling C# developers to gain **RAD Studio®** experience. Two to four weeks typically suffices to get comfortable with **RAD Studio®** and the frameworks, depending on the developer. As C# was written by Anders Hejlsberg, who originally wrote **Delphi®**, there is a strong influence in the way C# mirrors the **Delphi®** approach, making this a feasible path.

The architecture of **RAD Studio®** has proven the test of time and continues to lead the way in innovative cross-platform solutions. Thanks to the component model that follows the best OOP practices, code is easily modernised to take advantage of market changes with minimal effort, providing exceptional long-term investment. Upskilling existing developers to **Delphi®** is easily achieved, and as **RAD developers are 5x more productive** than they are with other frameworks, it provides exceptional long-term productivity gains for any team.

Best of Both Worlds - Why RAD Studio®

Evolution of Cross-Platform Development Tools and Approaches

Earlier in this paper we looked at innovations impacting software development over the last 25 years. Time has shown that any emerging technology typically takes around 10 years to mature from the constant changes of the early growth cycle as standards gradually solidify. The early days of any technology carry higher risk, so typically providers look for the lowest-risk approach to delivering long term solutions. This early market mindset has led to the launch of a wave of web-centric solutions as a way to reach into new devices. With a web browser on every device, it was a quick win to test and establish full market needs.

An early version of **RAD Studio®** support for the Web came in the late 90's when IntraWeb was added to **RAD Studio®** to enable development for multiple web formats, including early browsers on phones and PDA's (anyone remember WAP?). Since then, web standards have improved, JavaScript has risen in popularity, and libraries have emerged to provide faster development and multi-device support that are far closer to the desktop browser's code standards.

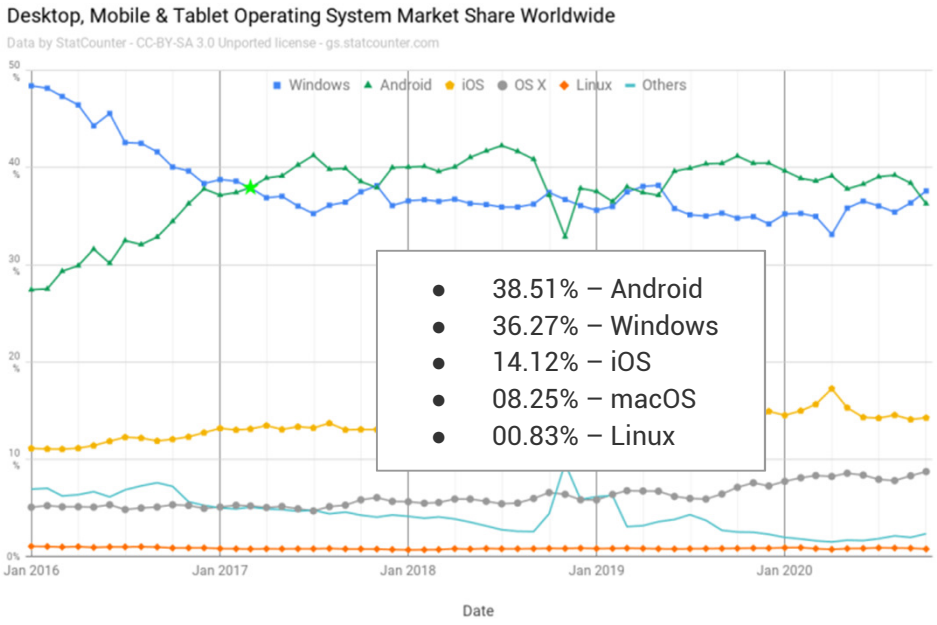
While **IntraWeb** and other **RAD Studio®** components like **TMS Webcore** have supported RAD web development in **Delphi®**, many standalone HTML and JavaScript frameworks have emerged as well. A leading example of this JavaScript-based approach in the RAD world is **Sencha**. Sencha offers **best-in-breed JavaScript-based web components** with properties to set and link to, enabling the rapid development of rich web applications. Sencha is used by and embedded with many solutions around the world, including some from the likes of Oracle, and since joining the Idera Group, it is included in the **RAD Studio® Architect** product.

The flexibility to both develop in web languages and compile native code (which is faster to execute) provides a rich set of capabilities for **RAD Studio®** users. However, as there are other options in the market as well, it would be useful at this point to summarise the differences between these approaches. Let us also look at the market each platform must reach.

Impact of Mobile on Business Processes

While mobile is not the primary platform for business applications, it's receiving intensive focus thanks to a new wave of **Business Process Reengineering** that is taking advantage of remote data capture capabilities. In a drive towards increased automation, many tasks that would traditionally have been completed centrally are being distributed to field workers to reduce paperwork and shorten process workflows. A good example of this would be that of an engineer making a site visit and completing a log entry, including photos. Mobile services are able to grab additional key data points, such as geo-coordinates.

With the aim of enabling this new wave of BPR, approaches such as BYOD have been adopted primarily to improve user adoption, but have also been embraced by companies as a way of reducing logistics around workforce enablement. This has become possible thanks to the broad market penetration of smartphones.



Operating System Market Share as of November 2020

Source: https://gs.statcounter.com/os-market-share

Although Android accounts for more users than iOS, the broad adoption of both frameworks combined with BYOD approaches means software needs to reach both iOS and Android for successful rollouts. To make things more complicated, successful products for each platform adhere to different user interface design guidelines. Failing to adhere to these guidelines can cause user acceptance friction that lowers the chance of a successful rollout.

In short, to enable a positive mobile-centric BPR capability today, software engineers need to be able to **reach iOS and Android at the same time**, with a look and feel that matches each platform's idiosyncrasies.

Reaching Multiple Platforms

While web technologies provide a quick way to reach a new device and platform, this approach is limited by browsers and their capabilities, causing constraints for BPR. Serving web pages is also slow, causing a poor user experience. For the best reliability, speed, user experience and access to device features (especially where security permissions are required), applications need to be installed on the device.

Two approaches are available to create mobile applications: those that are fully compiled and native, and hybrid applications that run a local web app inside a shell that provides a browser with enhanced access to features of the device.

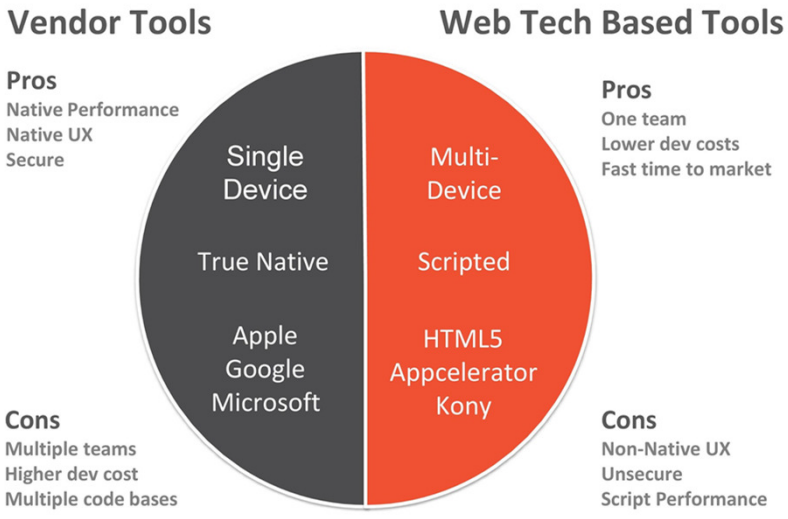
True Native Versus Hybrid Applications

True Native code is typically associated with vendor tools such as Xcode, Visual Studio and Android Studio. The upsides of True Native development are a native user experience, native speed and performance, and a high level of security as the code is compiled down to binary. Inversely, using vendor tools to achieve these benefits means having multiple code bases to develop and manage, and multiple skill sets and developers, which all lead to a higher development cost.

This **increased time**, **logistics** and **costs** are a reason many developers look to scripted/hybrid applications. Hybrid approach enables a single team of developers to create applications with a lower cost, that reach multiple platforms at the same time, but trade off security, performance and the native user experience.

The reason why **security and performance are low** in scripted/hybrid applications is because the scripted language needs to be interpreted at runtime. With the code executing at runtime, this provides a performance bottleneck and also creates a potential security hole for hackers to exploit. Additionally, the memory required for web applications is far higher than for native applications.

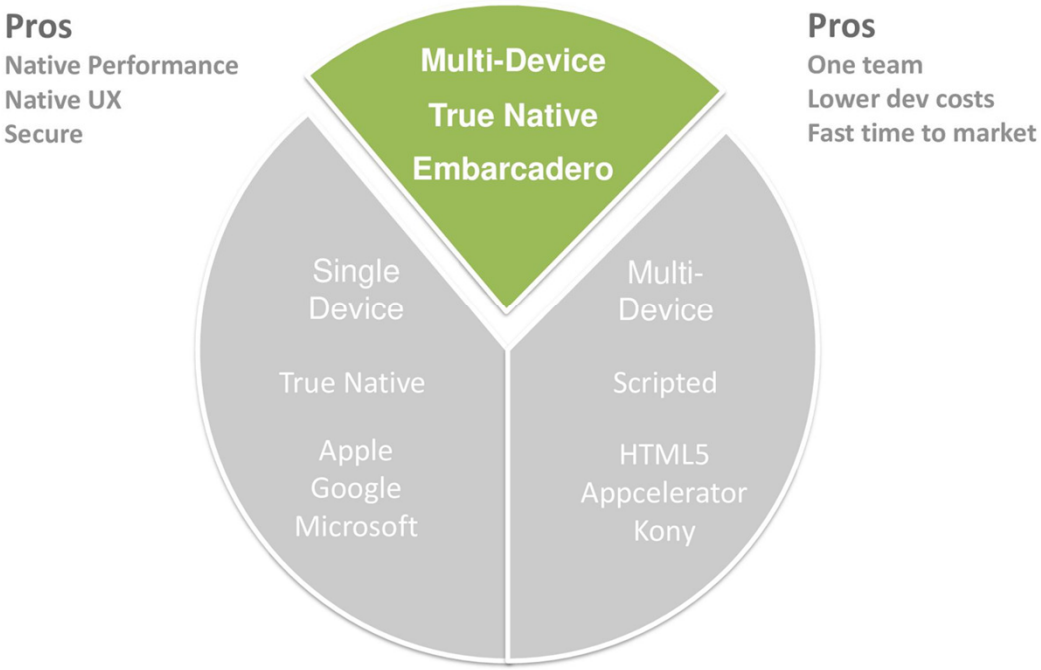
Finally, if your application is using a runtime such as .Net or JavaScript RunTime Webkit or JavaRuntime, this poses an additional security threat. Runtimes also come with considerable memory overhead, in part through the addition of a garbage collector for memory management.



No Compromise - The Best of Cross-Platform AND True Native

RAD Studio® brings a unique no-compromise approach to cross-platform development, enabling one development team to reach every targeted platform at the same time with a fully compiled True Native application, user experience, blazing performance, and the highest levels of security.

Using LLVM, every RAD Studio® application is compiled and highly optimized for each platform. This means RAD Studio® uses the same compiler for iOS and macOS that Xcode uses, and compiles to a lower level than Java to reach the same CPU and GPU access games developers target on Android while also avoiding the need to bloat memory with a garbage collector.



Launched in September 2011 as part of Delphi® XE2, the **FireMonkey (FMX) framework** has matured to offer the value of a single code base combined with the performance and security of a truly native user experience. Platform defaults for controls make it easy to achieve a platform-specific look and feel instantly, but still leave the developer in full control to customize where needed. High productivity features such as **FireUI** enable developers to immediately see (on any device) exactly how the UI design will look and behave as they develop, dramatically shortening development time.

Thanks to the use of a common compiler architecture, full language and framework support is available on all platforms. **RAD Studio®** FMX code can also run on Windows, macOS and Linux, providing even higher levels of productivity.

RAD Studio® additionally brings the ability to package applications ready for the notarization or deployment to the leading app stores directly inside the IDE, simplifying the entire build process.

How FMX Differs From Xamarin Forms

In May 2014, **Xamarin Forms** was launched in an attempt to provide an experience similar to FMX while using XAML. This provides Xamarin developers with a subset of controls for cross-platform development. Unfortunately, Xamarin falls short when compared with **RAD Studio®** in two key areas.

First, **language features are limited** depending on the platforms (e.g. generics can not be used on iOS meaning code often needs to be specifically written for each platform). Second, as Xamarin is based on .Net, it suffers from **inefficient memory management** due to a Garbage collector being added to iOS and a second one running on Android (both the .Net and a Java one).

Enterprise Data and Remote Data Connections

While **RAD Studio®** provides powerful mobile and desktop application development options, the complete stack requires serving data to remote devices.

Over the years, **RAD Studio®** has introduced a range of methods for enabling remote access to data. Today, the FireDAC Components enable local or remote connectivity to over 15 traditional SQL and NoSQL databases (plus ODBC for any others), and has also been expanded by partners to enable direct access to **180+ enterprise and big data systems** (such as Jira, Salesforce, Microsoft Teams, Google Drive, eBay, Facebook, Slack, Twitter, Amazon Marketplace) via standard SQL, with many included as part of the Enterprise product.

This ability to rapidly link to multiple databases and enterprise data sources makes **RAD Studio®** an ideal middle tier. With **WebServices**, **DataSnap** (based on Midas to enable a session's base connections) and **RAD Server** (a fully RESTful MEAP with Docker configurations and inbuilt usage analytics and reporting), there are options for creating and integrating with a range of systems, and exposing data rapidly, often with zero code.

Low-Code Application Platforms and RAD

Gartner defines a Low-Code Application Platform, or LCAP, as "an application platform that supports rapid application development, deployment, execution and management using declarative, high-level programming abstractions such as model-driven and metadata-based programming languages, and one-step deployments. LCAPs provide and support user interfaces (UIs), business processes and data services".

A growth in interest around LCAPs primarily goes back to the need to increase business automation through software, with a focus on business process review and the creation of apps to support a changing business. To achieve their rapid deployment and cross-platform support, LCAPs use a form of hybrid application based on web technologies to deliver their software.

Low-code options in the market are often sold based on their promise of enabling "citizen developers", where anyone can make an app with a little training. While the theory is good, the reality dictates a **steep learning curve** of a **bespoke limited-scope product** and a **heavy reliance on the vendor** to provide third-party integrations and maintain them. Some systems do provide advanced web-based RAD tooling for more seasoned developers to go beyond the simple tasks once they have been upskilled in the frameworks and models specific to the LCAP. A word of caution: the citizen developer route can also leave a business's software highly fragmented without a clear application strategy in place, producing a long-term technical debt that is hard to shake or manage.

For many of the low-code platforms, the **costs substantially mount up** through ongoing **subscriptions** charged per user for accessing the apps. Additionally, Gartner cautions that many low-code vendors earn substantial revenue from professional services, suggesting that professional developers who know the frameworks, rather than citizen developers, are required to support the tools. This means the true cost of LCAP ownership is substantially higher than initially expected for most.

With the history **RAD Studio®** brings to low-code development and support for business process review, rapid prototyping of applications and an open ecosystem for adding any connectivity required (either off-the-shelf or independently), the question for those looking at LCAPs is where the investment is better placed. Is upskilling a niche skill set belonging to a specific LCAP provider, or for enhancing existing team members with knowledge of **Delphi®**? With the latter able to deliver native applications that enable faster and more secure output across the leading platforms, it makes a compelling case to look at **RAD Studio®**.

RAD Studio® Today - Investing in Your Future

RAD and DevOps

The current industry trend for development process enhancement is DevOps. The understanding that developer and deployment teams need to work together to achieve a successful outcome is driving innovation around deployment configurations and setup.

The focus of DevOps could be a white paper in itself, but sometimes a picture does much more than words can. As we touched on earlier in the paper, because of the open ecosystem linking into **RAD Studio®**, many developer productivity tools are available in all stages of development. A subset of these tools is highlighted in the "**Power of RAD Studio®**" infographic on the next page.

embarcadero

The Power of RAD Studio

Code once, Natively target

Windows · Linux · macOS · iOS · android

CODE

Languages

Delphi · C++

UI Frameworks

VCL for Windows · FireMonkey Universal Cross Platform UI · Sencha Web Components · HTML · JavaScript · Intraweb* · TMS WebCore* · uniGUI

Cross Platform RunTime Frameworks

AppTethering · Parallel Programming · XML
REST & JSON · RTTI · Graphics
RAD Server · Web Broker
DataSnap · SOAP

Data

FireDAC Univeral Database Access

SAP Advantage Database Server · Firebird · IBM DB2 · Informix Software · InterBase · MariaDB · Microsoft Access · Microsoft SQL Server · MongoDB · MySQL · Oracle Database · PostgreSQL · SQLite · SAP SQL Anywhere · ODBC Generic ODBC Driver · teradata.

Enterprise Connectors

70+ Sources included with Enterprise and Architect Edition.
Up to 180+ sources via subscription*, the largest number of supported data sources in the industry.

PayPal · Facebook · MailChimp · Microsoft Teams · Twitter · Slack · Salesforce & Force.com · Wordpress · ebay · JIRA · DocuSign · Office · SurveyMonkey · Trello

BUILD

Database Management

AQUAFOLD Aqua Data Studio · Data Explorer · IBConsole

Platforms

Windows · Linux · macOS · iOS · Android · Web

Version Control Integration

Mecurial · Git · SVN · Github · Assembla

Build Systems

CMake · MSBuild · FinalBuilder*

TEST

Unit Testing (Automated)

Dunit · DUnitX · TestInsight* (IDE plugin) · Sencha Test · Delphi Mocks*

UI / UA Testing

TestRail · Ranorex · TestComplete

Code & Runtime Analysis

TMS FixInsight* · CodeHealer* · AQTime Pro* · Pascal Analyzer* · Deleaker*

INTEGRATE

Monkey Builder* · Jenkins* · Continua CI* · RAD Studio IDE Integrated Deployment · Configuration Management · Azure Pipelines

DEPLOY

Microsoft App Store · Google Play Store · iOS App Store · macOS App Store · Docker · Web Server: Apache · Microsoft IIS

OPERATE

FireDAC Database Services

Backup · Restore · Database Validation

MONITOR

RAD Server Analytics · CodeSite · Code Site · FireDAC - Tracing and Monitoring with FDMonitor · EurekaLog* · MADExcept* · Grijjy Logger* · SmartInspect*

* RAD Studio is also supported by a range of open-source and commercial offerings. A few are featured in this infographic for illustration purposes. Always check the feature matrix for the latest on what is included with RAD Studio. Features subject to change.

Investing for Both Present and Future Gains

RAD Studio® has arguably the best long-term return for code investment in any development tooling, and continues to focus on ensuring code is portable from the older version to the latest.

The roadmap for the future, regularly updated and shared with the **Embarcadero** community, shows the adoption of the latest innovations around Apple Silicon, Microsoft MSIX packaging and AppStore changes.

Summary

As we have seen in this paper, the desire for RAD development is growing with many choices in the market for hybrid or native development. **RAD Studio®**'s unique True Native approach brings the fastest speed, performance and security to mobile development built on the same compiler technologies used by native-only tool vendors while providing the value of a single-source cross-platform code base.

RAD Studio® continues to develop and embrace market trends, providing code security that is second to none in the market, ensuring the highest possible return on investment for software development.

In response to the leading trends identified, **RAD Studio®** enables RAD development for middle tiers and microservices, deployment to the leading desktop and mobile app stores, and wide-ranging data connectivity to both SQL and NoSQL databases as well as leading enterprise systems.

RAD Studio®'s ability to offer a unique solution in the market, along with a framework that has already proven able to expand without compromise to Windows, macOS, iOS, Android and Linux, points to a bright and secure future for **RAD Studio®** code.

Beyond This Paper

While not strictly the scope of this paper, it is worth adding a few notes for how developer productivity has been enhanced in RAD Studio in recent years.

RAD Studio now includes compilers, enabling native app development into Windows (32-bit and 64-bit). Linux (64-bit), Android (32-bit and 64-bit) iOS (64-bit) and macOS (64-bit). All compiled from a single code base, reducing testing and management costs across all target platforms.

Developers connecting into the latest version of the IDE will benefit from many developer productivity updates in the IDE too, including:

- Integrated **source code repository support** for Git, SVN and Mercurial
- A **modernised IDE layout** and optional Dark Style
- **Larger memory support** to support compiling even larger projects
- **LSP (Language Server Protocol) support** that enables faster coding with background processing of Code Completion, and error insight tasks
- And more…

For further information, please contact an Embarcadero sales representative.

The Embarcadero Way

Single-source, native multi-platform software development

The RAD Studio IDE and frameworks enable you to write your code in modern Delphi or C++ languages, and compile your single code base to natively target Windows, Linux, macOS, iOS, and Android.

Design it, Build it, Run it - Today!