

Curriculum Vitae

Regan Russell BSc

January 2, 2026

Phone: +61 41 428 7577

Email: regan @ pymblesoftware.com

regan.russell @ gmail.com

<https://www.linkedin.com/in/pymblesoftware/>

GitHub:

<https://github.com/pymblesoftware>

Previous Apps

<https://hummm-group.com>

<https://apps.apple.com/au/app/humm/id1455391873>

<https://apps.apple.com/au/developer/humm-pro-pty-ltd/id1532343413>

<https://apps.apple.com/ca/app/humm-ca/id1586818651>

<https://apps.apple.com/au/app/milieu-climate/id1566271872>

<https://apps.apple.com/us/app/crowddj/id911666442?ls=1>

<https://apps.apple.com/au/app/heritage-mobile-banking/id386772598?ls=1> .

<https://apps.apple.com/au/app/parknpay/id1453474761?mt=8>

<https://apps.apple.com/au/app/rydo/id1150318596>

<https://apps.apple.com/gb/app/dopel/id1434777360>

<https://apps.apple.com/app/id1179563005>

Personal Apps:

<https://apps.apple.com/sa/developer/pymble-software-pty-ltd/id553990081>



Latest CV

Previous Experience

Contract roles through PymbleSoftware Pty Ltd since 2008 18 year(s), 0 month(s) .
(Personal contracts since 1998) 28 year(s), 0 month(s) .

Recent

Academic projects.

Business projects.

Personal software projects.

Real estate projects.

Working with publishers on technical manuscripts as part of the editorial cycle.

Cross-platform mobile developer with Flutter / PHP Laravel (Lumens) developer (Contract)

Charitable Organisation

Flutter (3.10.6), Flutter Realm (1.3.0), Dart, Stripe (flutter_stripe:9.3.0), credit_card_scanner (1.0.5), PHP, Laravel. Lumens.

GDPR (privacy by design) and PCI DSS (card data handling) compliant as possible. Only static content is kept in an in-memory database, not even in storage.

Over 12,000 lines of Flutter code. Over 800 lines of PHP web service code.

I read somewhere that the average US programmer averages 200 lines of code per day. In the first month, I averaged over 500 lines of code per day. **Greenfield** project from start to finish. Donation app for water and food aid projects. **Cross platform with flutter**. Did all the web services with Laravel/Lumens and MySQL database.

Payment gateway integration with Stripe.

iOS and Android Developer (Native)

Company **Humm Group**

<https://hummm-group.com>

<https://apps.apple.com/au/app/humm/id1455391873>

<https://apps.apple.com/au/developer/humm-pro-pty-ltd/id1532343413>

<https://apps.apple.com/ca/app/humm-ca/id1586818651>

Apps: Bundll, QANTAS Pay, Humm Pro, Humm Canada, Humm Cards and legacy apps which included the following tools and libraries:

AFNetworking¹, FLAnimatedImage², Auth0³, NVActivityIndicatorView⁴, Floating Panel, JWTDecode⁵, Kingfisher⁶, Firebase, Dynatrace, AppFlyerFramework, CardIO⁷, EMSMobileSDK, Alamofire⁸, Lottie⁹, KeychainSwift¹⁰, Branch, Swipe, Validator, SwiftEntryKit¹¹, SVProgressHUD¹², SwiftLint, Quick, Nimble, IOSecuritySuite, Snapkit, SwipeCellKit¹³, ObjectMapper, SwiftyJSON¹⁴, DateToolsSwift¹⁵, RangeSeekSlider, NVActivityIndicatorView, IQKeyboardManager, **SalesForce** market-

¹ <https://cocoapods.org/pods/AFNetworking>

² <https://cocoapods.org/pods/FLAnimatedImage>

³ <https://auth0.com/>

⁴ <https://cocoapods.org/pods/NVActivityIndicatorView>

⁵ <https://cocoapods.org/pods/JWTDecode>

⁶ <https://cocoapods.org/pods/Kingfisher>

⁷ <https://cocoapods.org/pods/CardIO>

⁸ <https://cocoapods.org/pods/Alamofire>

⁹ <https://cocoapods.org/pods/lottie-ios>

¹⁰ <https://cocoapods.org/pods/KeychainSwift>

¹¹ <https://cocoapods.org/pods/SwiftEntryKit>

¹² <https://cocoapods.org/pods/SVProgressHUD-0.8.1>

¹³ <https://cocoapods.org/pods/SwipeCellKit>

¹⁴ <https://cocoapods.org/pods/SwiftyJSON>

¹⁵ <https://cocoapods.org/pods/DateToolsSwift>

ing cloud¹⁶, kotlinkit, cardview, androidx, Gson¹⁷, threetenabp¹⁸, brentrielly navigation, picasso¹⁹, retrofit2²⁰, OkHttp3²¹, biometric, rootbeer²², fillrembedded, viewpagerdots²³, filetree, play services, swipe reveal layout, EML payments²⁴, RxAndroid/RxJava²⁵, epoxy, TMX, securebank, swiperefreshlayout²⁶, liveness

Humm Group white labels credit cards and provides financial services such as small to medium-enterprise equipment leasing. Specifically, Humm provides consumer credit cards / Store cards such as the Farmers²⁷ chain in New Zealand and the QANTAS Pay card.

Since I have had experience with financial services and credit card processing online in one form or another since working with Plink libraries at BDE in 1997; I was able to quickly adapt to the workflows, codebases/business practices of the credit card services provided by Humm Group to provide bug fixes and minor maintenance on 192342 lines of Swift, 879 lines of Objective C, 334037 lines of Kotlin, 47,982 lines of java, excluding libraries. To develop the changes to these codebases I reviewed Figma diagrams and incorporated changes to UI elements in Android and iOS code as per the requirements of the JIRA tickets.

To modernise the build environment, I added Azure DevOps / Fastlane CI/CD build pipelines for a few of the apps.

(Native) iOS and Android Developer

Company **Milieu Labs**. (Contract)

<https://apps.apple.com/au/app/milieu-climate/id1566271872>

Consumer thermostat systems.

Pre-existing tech stack: Amazon **IoT**²⁸, AWS, SwiftFromat, RxSwift²⁹, RxCocoa³⁰, ReSwift³¹, AWS SDK iOS, Alamofire, Crashlytics, Freddy, Butterknife³², MQTT³³, RxJava²⁵, Redux³⁴, Mockito, Jetpack. AWS Cognito, AWS IoT, Dagger2, OkHttp²¹, retrofit2²⁰

Milieu Labs provides controls for domestic air conditioning systems. The control unit of the air conditioner is replaced with a module that receives MQTT messages from a server which is indirectly managed through a user's account on the app. The project added a Zone information screen. Zone state change handling which entailed unpacking MQTT Zone state messages, and translating that through a middleware layer onto the screen through the UI controls.

Company **Nightlife music** (Contract)

<https://apps.apple.com/us/app/crowddj/id911666442?ls=1>

Java, Objective-C maintenance work and small projects including work on 8-year-old code base in Objective-C on iOS and Java on Android.

¹⁶<https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/mobile-push-sdk.html>

¹⁷ <https://github.com/google/gson>

¹⁸<https://github.com/JakeWharton/ThreeTenABP>

¹⁹<https://github.com/square/picasso>

²⁰<https://github.com/square/retrofit>

²¹<https://square.github.io/okhttp/>

²²<https://github.com/scottyab/rootbeer>

²³<https://github.com/afollestad/viewpagerdots>

²⁴<https://www.emlpayments.com/>

²⁵<https://github.com/ReactiveX/RxAndroid>

²⁶<https://developer.android.com/jetpack/androidx/releases/swiperefreshlayout>

²⁷<https://www.farmers.co.nz/>

²⁸<https://aws.amazon.com/iot/>

²⁹<https://github.com/ReactiveX/RxSwift>

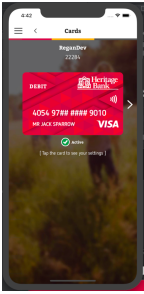
³⁰<https://cocoapods.org/pods/RxCocoa>

³¹<https://cocoapods.org/pods/ReSwift>

³²<https://github.com/JakeWharton/butterknife>

³³<https://mqtt.org>

³⁴<https://cocoapods.org/pods/Redux>



Heritage

Bank

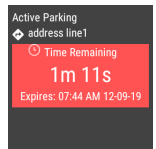
Client **NSW Government DFSI** (Contract)

iOS and Android developer

Technologies: Swift / Kotlin

<https://play.google.com/store/apps/details?id=au.gov.nsw.onegov.parknpay.release>

<https://apps.apple.com/au/app/parknpay/id1453474761?mt=8>



Added feature of Braintree payments integration in Swift and Kotlin in an app for NSW Government including Apple Pay and Google Pay.

Watch App

The previous payment **integration** had reportedly been problematic and the payment gateway vendor for Braintree was almost effortless.

It took less effort and resources than expected so I added the **features** of Apple Watch and Android Gear Watch integration into the app for the NSW government.

Did a custom notification where the countdown timer and buttons appear in the phone notifications even on the lock screen.

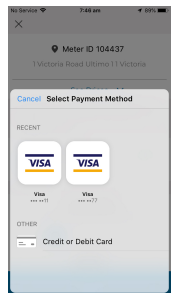
The app communicates with an internal server, which acts as a proxy for another server that manages parking meters.

Part of the credit card processing integration occurred on the parking meter servers and part was within the app.

The idea of the app is to pay for parking meters from the phone and be notified of the need to top up the parking meter if it is about to expire by tapping on the watch and the phone notifications.

Complex integration due to the number of servers involved and the number of moving parts and the level of security.

All the watch app extensions, all the app to payment gateway integration.



Payments

Client **Rydo Taxi app** . (Contract)

<https://apps.apple.com/au/app/rydo/id1150318596>

<https://play.google.com/store/apps/details?id=com.Rydotechnologies.Rydo&hl=en>

iOS and Android developer.

Technologies: Swift / Kotlin

Maintenance work and small projects involving C#, Swift, Java, and Kotlin (Mostly Java on the Android side of things). Fixed about a dozen bugs each week on Android then about a dozen on iOS the next week. The code was breathtakingly bad, if the server found no results in the database it would return 404 instead of 200 and an indication of an empty dataset in the JSON payload.

The server would 500 or 400 frequently on requests that worked a minute ago.

Somehow previous developers had considered the foreground and background of the app to be handled in view controllers instead of the app delegate, there was a lot of surreal code.

Whoever wrote it had no idea of the application life cycle or standard industry practices like not abusing HTTP error codes.

I was doing work on iOS, Android and ASP.NET and completing **features** in two weeks they had waited 2 years to be completed.

Client **ServCorp** (Contracts)

<https://www.servcorp.com.au/en/oneap/>

Technologies: Swift / Kotlin / Objective-C / Java / PHP / Larvel / CodeIgniter / L^AT_EX

Comments: Replaced back-end login process with Auth0³ (an OAuth implementation). **Maintenance** work in PHP and iOS and Android code to replace naive login and session verification.

Did a **localisation** for the Japanese (日本語) version. I have some understanding of Japanese having once worked in (Citrix 2005/2006) and visited the country several times.

Wrote extensive documentation, and fixed bugs. **Refactored** local login to an OAuth login with Auth0³.

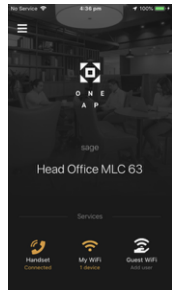
Client: **eDale** / **ServCorp** (Contracts)

Technologies: Swift / Kotlin / Objective-C / Java / Node.js / Heroku / PHP / Larvel / CodeIgniter / L^AT_EX / Android

Comments: Several overlapping short-term contracts. Some clean-up of previous work for Edale, bug fixing, and random ad hoc changes. Hand over and manage a developer I hired through my own company to take over work.

Short-term work on Bluetooth tile beacons for finding car keys - type tag beacons. Another USB-based Bluetooth beacon (Sensoro) and QR code project for allocating desktop handset phones and Wifi access. Worked 3 days a week at Servcorp doing maintenance work on an iOS and Android app that allows for provisioning of desktop handsets and (guest) wifi access

Sensoro beacons, translating Chinese comments into English. **Implemented feature** of Android circular reveal animation like the iOS animation. completed button handlers, differentiated between my wifi and guest wifi buttons and called different web service endpoints accordingly. Scale and slide up login animation to match the iOS version. Made UI consistent with the iOS version. Added disconnect handset screen in Kotlin. Added **feature** of pull to refresh the menu on the Android version to be consistent with the iOS version. Added site features refresh endpoint call and UI update. Added background handset polling. Used to be location dependent. Fixed refresh, device count wrong and other bugs in. both iOS and Android versions. A branch was created in the source code control system called allow_handset_choice to allow the choice of a beacon/handset other than the nearest found beacon. An extra button was added to the found handset screen. Touching this button launches another screen with a list of handsets detected. On both the iOS and Android versions all the beacons that were available in the area are stored in collections in the bowels of the Bluetooth code. In both versions, the collections were propagated up through the software layers until they are presented in a UITableView or a recycler view and the on-item selection handlers then put the selected MAC address in place of the nearest found beacon MAC address and then call the occupy handset code which calls the web service endpoint (Function 16).



Servcorp
OneAp

Half of the endpoints of the ServCorp server were in CodeIgniter and half were in Larvel in another repository. I set up a local copy of the code bases on my development machine for debugging server-side and client-side issues.

Client: **Edale Holdings Pty Ltd** (Master contract)

The app has been removed from the app store.

Technologies: Android/iOS/Heroku/Amazon S3, Kotlin, retrofit2²⁰, Swift4, node.js, mongoDB.

Front-end and back-end developer. Team leader

Comments: **Greenfield development** Created a node.js web service with about a dozen endpoints. Created an iOS app that allows users to register, log in and like other users of a dating app. User images on S3 displayed on a dating app. Created an Android version of the same app to talk to the same web service. Full control over everything changing web services to fit the needs of mobile apps and changing mobile apps to deal with web service restrictions. Next to no formal specification. Lots of client hand-holding, random inconsistent changes, About 2700 lines of Kotlin, about 1000 lines of node.js and about 2200 lines of **Swift4** in the first week.

I eventually hired another developer through my own company. **Complete app start to finish**

Client: **Seven Studio at the Easter Show (2 weeks), AMP Banking App** (Multiple Contracts)
Technologies: Android, iOS (iPad), AlamoFire, ffmpeg, Swift 3, AWS, S3., Carthage, mongoose.js, node.js, MongoDB, Kotlin, **iOS/Node.js, MongoDB, Android developer.**

Comments:

AMP Android banking app - Kotlin, RxJava, Dagger dependency injection. A mess of deeply nested Kotlin templates. “Clean design” – model, view, presenter, use case, contracts.

7Studio, an App that records videos, displays a teleprompter. The recorded video gets an overlay graphic applied and then merged with a news break intro and news break closing theme. RecordedUI XC unit test. Memory leak debugging with instruments. Run at the Easter show for kids to be a reporter for a day.

Client: **Invocare** (Contract)

No link: Internal Corporate portal app

Technologies: iOS. Objective-C (pre-dating ARC), iOS, Web Services

iOS developer, Architect. Team leader.

Comments: Maintenance work on an iPad app used to sell funeral services. **Salesforce**, Magic Record, SDWebImage, Masonary, Cocopods. There were about 100 cases in JIRA. I go through, pick a case, read a chain of emails going for 6 months, put in a fix for 5 lines of code in 20 minutes, stick it on a branch in bitbucket. I end up creating a dozen or so branches, all different fairly trivial fixes, never code reviewed, never tested. I’m told they’re doing a hybrid app now (Cordova, JavaScript) and my role is now “Application Architect”. I have to look at issues like the flow of data app to SalesForce to Epicor is truncating data and organise workshops with end users, and I digging around in T-SQL on SQL Server and APEX classes in SalesForce triggers and I end up churning out about 1000 pages of documentation and writing 50 to 100 lines of code in total. I hired a handful of developers. Lots of meetings, gathering requirements with end users.

Resolution of issues that had been outstanding for over a year

Client: **RaceNet** (Contract)

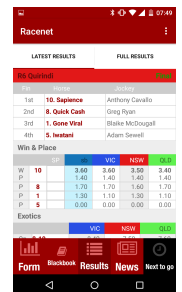
No link: The app was replaced after several years.

Technologies: Android and iOS. Java and Swift.

iOS, Android, Web Services developer.

Comments: **Greenfield development.** For the Android version I wrote about 20,000 lines of code in the first month. About 5,000 lines a week. GSON, Android studio. Got the basic version of the Android app completed. Worked with Web services developer on defining data models and endpoints. Used about 30 web service endpoints. I used Cocoapods and SwiftyJSON. Wrote the iOS version in Swift. I ended up churning out about 30,000 lines of Android code in Java and about 20,000 lines of code in Swift.

Complete app start to finish



RaceNet

Client: **AiiMS** (Contract)

Technologies: iOS 8, CoreData, Xcode 6, XCTest unit testing framework.

Team leader iOS, Android, Web Services.

Comments: **Mentoring** Taught PHP developer Objective-C, the use of libraries like Cocopods and Cococontrols.com. Taught use of AFNetworking¹ and JSONModel. Taught C# .NET developer Android, Gradle, Maven, use of libraries and search.maven.org. Taught both the use of git. Evaluation of web service frameworks such as node.js, and Ruby on Rails before settling back on C# .NET. **Technical leadership** on various things such as selecting CMS - bespoke vs Joomla vs others.

Client: **Macquarrie University finance department, TriBeeCam** (Multiple Contracts)

Technologies: Perl on Windows, NAB data source, Excel Spreadsheets and VS Script. iOS 8, CoreData, Xcode 6, XCTest unit testing framework.

Freelance project contractor.

Comments: Did maintenance work for a university that brings me back every couple of years.

Did AFNetworking¹ to JSONModel wrapper for a camera app. UIKit, everything is done in code, no storyboards, nib files, or anything. Git, JSONModel, JSON.

Client: **DoppelTime.** (Contract)

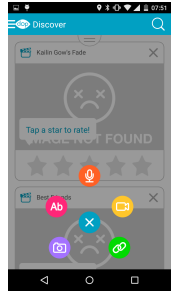
Link: <https://apps.apple.com/gb/app/dopel/id1434777360>

<https://play.google.com/store/apps/details?id=com.dopel.dopel>

Technologies: iOS 7, CoreData, Xcode 5, Unit testing framework.

iOS developer.

Comments: Completed an iOS app, added social elements, camera roll picker, camera control, voice recorder, bug fixing and finished off half-completed code. The company is a start-up in a start-up incubator. High pressure for quick results. Used Cocopods, AFNetworking¹, UIKit, GCD, Flurry Analytics, GoogleMaps, Core Graphics, added a bunch of UIViews to a core animation layer, did some cute explode-out pseudo-button animations and added gesture recognizers to the UIViews. Core Location, AVFoundation for Camera & Audio record/playback. Some node.js debugging of the services the iOS app used. Set up node.js for local testing of mobile code.



DoppelTime

Client: **SmartBill** (Contract)

No Link: Removed - App Store policies

Technologies: Android, Sqlite3, iOS 7, CoreData, Xcode 5s Unit testing framework.

iOS/Android developer.

Comments: **Greenfield development.** Developed an Android and iOS app to gather data usage and call log data and send it to a server for 'smart bill' analysis against phone plans. Used silent push notifications to wake the app up for processing. AFNetworking¹, working on ASP.NET code for Apple push notification servers. SQL Lite.

Complete app start to finish

Client: **Kordia** (Contract)

No link: The app was available through the corporate portal only.

Technologies: JSONKit, YAJL JSON parser, Telerik C# controls, Visual Studio 2010, SQL Server 2010, Team Foundation Server. ASP.NET web services.

iOS Developer.

Comments: **Maintenance** work on KST, Kordia's iPod app that communicates with a web service pulling down JSON data for Telco site planning, such as Telstra, Downer and NBN. Converted JSON requests to background SAX style streaming requests updating a UITableView as large projects with lots of assets were taking a long time to update with no indication to the user of any activity. Employed code blocks, ARC, Multi-threading and other more recent or advanced iOS techniques. A little C#/XAML but mostly Telerik controls.

Education

Mobile

Certificate Kotlin for Android: Best Practices **LinkedIn Learning**
Certificate Advanced Kotlin Database Development **LinkedIn Learning**
Certificate Android Dependency Injection with Dagger 2 and Kotlin **LinkedIn Learning**
Certificate Android App Development: Enterprise Integration **LinkedIn Learning**
Certificate Android Development: Understanding Intents **LinkedIn Learning**
Certificate Building an Android App with Jetpack Libraries **LinkedIn Learning**
Certificate Android App Security: A Structured Approach to Pen Testing **LinkedIn Learning**
Course Jetpack Compose crash course **Udemy**
Certificate Android Development: Data Binding **LinkedIn Learning**
Certificate Android App Development: Accessibility **LinkedIn Learning**
Certificate Android Development: Retrofit with Kotlin **LinkedIn Learning**
Certificate Building Android Apps with AWS **LinkedIn Learning**
Certificate Android App Development: Design Patterns for Mobile Architecture **LinkedIn Learning**
Certificate Java: Testing with JUnit **LinkedIn Learning**
Certificate iOS Development Tips **LinkedIn Learning**
Certificate Coding Exercises: Swift **LinkedIn Learning**
Certificate iOS App Development: Accessibility **LinkedIn Learning**
Certificate iOS App Development: Test-Driven Development **LinkedIn Learning**
Certificate iOS Development: Security **LinkedIn Learning**
Certificate Building iOS Apps with CloudKit **LinkedIn Learning**
Certificate SwiftUI Essential Training **LinkedIn Learning**
Certificate Make SwiftUI Playgrounds Applications **LinkedIn Learning**
Certificate Apple watchOS App Development: Advanced APIs³⁵ **LinkedIn Learning**
Certificate Biometric Authentication for iOS in Swift **LinkedIn Learning**³⁶
Certificate Biometrics: Security and Privacy Considerations **LinkedIn Learning**
Certificate iOS App Development: Design Patterns for Mobile Architecture **LinkedIn Learning**
Certificate iOS 17 Development Essential Training **LinkedIn Learning**
Certificate Swift 5 Essential Training **LinkedIn Learning**
Certificate iOS App Development: Core ML **LinkedIn Learning**
Certificate iOS and watchOS App Development: Notifications **LinkedIn Learning**
Certificate Swift: Delegations and Data Sources **LinkedIn Learning**
Certificate Composable SwiftUI Architecture Using Redux: 1 Introduction **LinkedIn Learning**
Certificate Composable SwiftUI Architecture Using Redux: 2 Building the App **LinkedIn Learning**
Certificate Building an App for All Apple Platforms **LinkedIn Learning**
Certificate Xamarin.Forms Essential Training **LinkedIn Learning**
Certificate Xamarin Essential Training: Create Your First App **LinkedIn Learning**
Certificate Moving from Xamarin.Forms to .NET MAUI **LinkedIn Learning**
Certificate Dart Clean Code: Writing High Efficiency, Maintainable Dart Programs **LinkedIn Learning**
Certificate React Native Essential Training **LinkedIn Learning**
Certificate Introduction to SQLite **LinkedIn Learning**
Certificate Learning React Native **LinkedIn Learning**
Course Create a tiny app with ReactNative **Udemy**
Course Reactive Programming in iOS with RxSwift **Udemy**

Publications

Russell R., (2024) "Thirty-five years in Software Development", Kindle from
<https://www.amazon.com.au/dp/B0DK4FHMZ4> .

Russell R., (2024) "Thirty-five years in Software Development", Paperback from
<https://www.amazon.com/Thirty-five-years-Software-Development-Leadership/dp/B0DP7CR7SV> .
ISBN 979-8301444326



35 Years

book

Russell R., (2024) "Thirty-five years in Software Development", Hardcover from

<https://www.amazon.com/Thirty-five-years-Software-Development-Leadership/dp/B0DPDLWPRC/> .

ISBN 979-8301885365

Russell R., (2012) "Programming bada", Kindle, iBooks and PDF file from www.pymblesoftware.com/book .

www.pymblesoftware.com/book/bada-short.pdf .

<http://itunes.apple.com/au/book/programming-bada/id543013439?mt=1&ls=1>

<https://www.amazon.com/Programming-bada-Regan-Russell-ebook/dp/B007LFX608>



bada
book

Russell R., (Nov 1999) "BeRays: A ray tracer for BeOS", Doctor Dobbs Journal.

Russell, R., & Ryan C (1994) "METAcoder for windows: real-time and multi-pass.

event logging and analysis in the social and behavioural

sciences." Psychology Teaching Review.

Russell, R., & Ryan C. (1994) "METAcoder for windows: real-time and multi-pass.

event logging and analysis in the social and behavioural.

sciences." Psychology Software News.



Doc Dobbs
Journal

Review of Windows NT Device Driver Development Doctor Dobbs electronic review of computer books (ERCB).

Review of The Windows NT Device Driver Book: A Guide for Programmers ERCB.

Review of Developing Windows NT Device Drivers ERCB.

Review of Writing a UNIX Device Driver, Second Edition. ERCB.

Review of Panic: Unix crash dump analysis, ERCB.

Review of Advanced animation and rendering techniques, ERCB

Review of Windows TCP/IP. ERCB

Review of Open source development with CVS. ERCB

Review of System performance tuning. ERCB

Review of Learning the vi editor. ERCB

Review of Ada for experienced programmers. ERCB.



BeRays
Article

Publications Officer, James Cook University SCUBA Dive Club 1992 - 1993.



Latest CV



Book: 35
Years in
Software
Develop-
ment



Book: Pro-
gramming
bada



Book a
meeting
with me