DEVELOPING FOR THE PERVASIVE, UBIQUITOUS, M2M INTERNET OF THINGS

10 Tips to Prepare for the Future of Mobile
INTRODUCTION

There is little doubt that mobile devices have revitalized modern computing. Yet even as hardware and network providers prepare for emerging devices connecting the home to the office, appliances to sensors, cars to intelligent transportation, etc., software developers are still trying to tackle the highly fragmented mobile phone environment that this explosion of devices has created.

New tools and standards, such as BONDI, have recently emerged to assist developers, yet many do not fully support the wide range of sensors, routers, cars and embedded devices that are a part of our pervasively connected future.

Even a name for this connected world is not fully agreed-upon: pervasive computing, ubiquitous computing, M2M, the Internet of Things... each phrase shares a common vision, but the technical requirements remain ill-defined.

This eBook is a high-level overview designed to help developers think about the future of mobile app development and in the absence of firm standards, offers tips for choosing a pervasive software architecture that will future-proof your application. It is our hope that you will make it one of your most valuable resources to help shape your pervasively mobile strategy.

Victoria Romero-Gomez
Product Manager, Mobility Solutions
Recursion Software
The World Wireless Research Forum (WWRF) predicts there will be 7 trillion devices for 7 billion people by 2017. In other words, around one thousand devices for every man, woman and child on the planet. Efforts to connect these devices have many names:

- **M2M**
- The Internet of Things
- Network of Everything
- Ubiquitous Computing
DEFINING M2M

The common vision expressed by many of today’s market leaders can be used as a guideline for establishing functional requirements for the next generation of mobile, pervasive software. From their collective vision, a common theme emerges: **any content or service on any device, sharing content/services between devices and clouds, & the ability to discover and host device communities.**

"It's not the data center or the user device. It's any device to any content wherever it is in the world over any combination of networks wired or wireless to the home, to an Apple device, to a Microsoft device, to an IBM device, HP. Doesn't matter to us."

-- John Chambers, CEO Cisco

"The desktop PC, enterprise computing, mobile services running in the cloud and phone devices... [via] a rich platform."

-- Steve Ballmer, CEO Microsoft

"Data will be selectively pushed to the user based on context... combined with personal presence, creates transient communities..."

-- Nick Jones, VP Gartner
In their words, the future of pervasive computing is defined by any content accessible on any device over any network. Mobile services must be available to users both in the cloud and on the device, and will push content to users based on their location, interests, behavior, etc. Devices must be able to discover and host transient communities that exchange information, while providing customized security and data privacy options. Communities will be comprised of both human and machines participants collaborating and creating new content that is automatically pushed to the entire community in real-time.

In short, a checklist of minimum requirements to enable this common vision:

- **Any device, any content on any network**
- **Mobile services in the cloud & on the device**
- **Push content autonomously based on time, location, state of being (work, play, private, etc.)**
- **Form transient, ad hoc mobile communities**
- **Guaranteed message delivery**
- **Security and data privacy**
- **Enterprise integration**

"In what’s called the Internet of Things, sensors and actuators embedded in physical objects—from roadways to pacemakers—are linked through wired and wireless networks... When objects can both sense the environment and communicate, they become tools for understanding complexity and responding to it swiftly. What’s revolutionary in all this is that these physical information systems are now beginning to be deployed, and some of them even work largely without human intervention..."

-- McKinsey & Company
M2M’s TECHNICAL CHALLENGES

There are many technology barriers to reaching this vision which include:

- Conflicting standards
- Many networks and protocols
- Bandwidth scarcity and poor performance
- Various languages
- Device fragmentation (servers to TVs, phones to sensors)
- Content organization/separation (Public/private, work/personal)
- Security and privacy
#1: DESIGN BEYOND THE PHONE

The mainstream mobile model today is to deploy to one or two of the more popular handset platforms and forgo the rest of the market due to cost and time restrictions. It’s no surprise then that even the most popular mobile apps only work on PCs and a handful of select platforms, namely iPhone, Android, and BlackBerry.

There is also an assumption that smartphones will become the singular device for computing and content. While smartphones combined with mobile apps enable portable content, many companies have tried to produce a “one device fits all” device with little success. The reality is that people like to choose which device best fits their context at any given time, whether at work or at play, based on location or mood, or while accessing public or private content. Mobile users tend to have several wireless devices at their disposal, at minimum a PC and a Smartphone, with newcomers like eReaders, in-car entertainment and wireless TVs. The popularity of niche devices, such as the Kindle, iPad illustrate that when it comes to ease of use or media consumption, achieving the best experience trumps the economics and convenience of having a singular device.

Google certainly understands this mindset, and it has teamed with Sony to develop Android television. In March 2010, the company also co-launched the first Android-powered Roewe car for the Chinese market. But Android’s own fragmentation and technical bugs make its current incarnation unsuitable as a pervasive platform. The .NET Compact Framework (CF) is another platform yet to live up to its potential. And finally, a traditional web app cannot today provide the combination of face and faceless apps in a high performance, autonomous manner.

"We’ve seen numbers thrown out there from analysts that say because of the M2M space we’ll be at 500% penetration. I think it’s hard to walk into a Best Buy or Wal-Mart or RadioShack and see all of the devices on the shelves and find one that wouldn’t be a better device and have a better value proposition for the customer if it weren’t connected to the Web or to another device.”

-- Glenn Lurie, President Emerging Devices, AT&T

To deploy truly pervasive software, consider a **platform that supports a range of mobile and embedded software stacks, such as Java’s Micro Edition, Microsoft’s CF, and OSGi Containers**, on a wide range of embedded operating systems such as Windows Mobile, Symbian, and Embedded Linux to name just a few. The **platform must seamlessly integrate with .NET, JEE and legacy (MVS, CORBA, etc.) enterprise systems and services, in either a traditional Web Services architecture, or in a high performance manner.**
Even if your application initially targets a specific phone subset, the ability to control devices remotely or integrate data and services from other devices, many of which are machines and sensors, will soon be a necessity. The siloed mentality of one application with one user is no longer an adequate mobile strategy.

The role of communities, security and privacy will be discussed later in more detail, but the main take-away is that designing for advanced integration and communication with applications, systems, and machines is the foundation for our pervasive future. The number of devices, protocols and platforms grows exponentially when you consider initiatives to automate the home or connect the plant.

Your application must be designed to interact with a highly heterogeneous environment of human operated and non-human operated devices, with different operating systems and different languages. It will need to utilize intelligent software agents that can gather data, respond quickly based on this data as it changes, produce and distribute knowledge, and possibly initiate other agent activities. The underlying rules engine must be easy to use, provide very high performance against potentially large rule sets, and must be available in .NET, Java and C++.

Services need to be accessible via centralized Web Service Container such as Microsoft’s Internet Information Services (IIS) and those provided in the Java world, but also must be accessible in a decentralized fashion directly to agents, exposed as remotely accessible services, that are running on edge and wireless devices. The location of intelligent mobile agents and the mobile services they expose must be irrelevant to the client application.

"If you walk into the [M2M] space and you bring all of your baggage from launching a Smartphone or other things you’ve done in the past, you’re probably not going to be very successful. You will have to look at this very differently. Whether it’s a netbook or an eBook reader, the model is very different, and the approach with the [OEMs] is very different."

-- Glenn Lurie, President Emerging Devices, AT&T

((TECH TIP))
#3: CHOOSE YOUR ARCHITECTURE WISELY

The question of native vs. web has been long debated. And the answer is that no one-size fits all. The choice is largely dependent upon what kind of transactions your app needs to perform. **Rule of thumb: use a mobile web app when you need simple, less sophisticated apps and need to ramp up quickly. Native apps are best for high performance, for complex processing, multiple-devices types, and collaboration.**

As platforms continue to be highly competitive, the default option for most apps is to design for the mobile web to avoid fragmentation, although it should be noted that mobile browsers are also fragmented. The relative simplicity of the mobile web is often best if your application and business model offers enough differentiation within the noisy app market.

In contrast, native solutions offer better performance, consume less bandwidth, have offline capabilities, and offer more sophisticated transactions if fragmentation can be solved. **Additionally, more sophisticated apps need a robust object oriented language like Java rather than JavaScript which has inconsistent browser performance.**

The apps of our M2M future will likely be highly collaborative in a heterogeneous device environment, therefore they need pervasive native platform will outperform the web in performance, efficiency and security. They need to do more than just solve the fragmentation issue, they need to employ a distributed knowledge network that create meaningful content rather than simply post it.

This is where the phrase “Internet of Things” is a bit of a misnomer and the “Network of Everything” makes better sense. To understand the dis/advantages of each approach, we will review their basic architectures.

---

**Five Common Reasons for Native Apps:**
- Performance
- Offline Mode
- Findability [market channel]
- Device Attributes
- Monetization

-- Jason Grigsby, Cloud Four

**((TECH TIP))**
COMPARING ARCHITECTURES

- **Thin Client-Server (Centralized)**
  - Mobile Web (HTML, JavaScript)
  - Mobile Enterprise solutions (ex. Sybase, WebSphere)
  
  Description: This model uses mature technologies, such as HTML and JavaScript or an enterprise platform

- **Smart-Client (Distributed)**
  - P2P Edge Platforms (ex. Voyager)
  
  Description: This model uses mature technologies in a new way, by relying on mobile software agents to deploy intelligence

  - **Bidirectional communication**
  - **Bandwidth Efficiency**
  
  Each client has server intelligence and capabilities.

  Clients can be faceless, such as sensors, and use any language or OS

  Intelligent client performs transactions autonomously until a request to share is made.

  Content or messaging is sent P2P/Group (guaranteed messaging)

  High bandwidth use

  Client enters data, each click sends data to server

  Server hosts all services and content, acting as a gateway. It receives requests and data, performs filtering or transactions and sends output back to client(s) if shared

  Pull only

  Client displayed data. Process repeats for each click or transaction.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Enterprise</th>
<th>Mobile Web</th>
<th>P2P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Mid</td>
<td>Worst</td>
<td>✓ Best (+20-50%)</td>
</tr>
<tr>
<td>Shared services</td>
<td>No</td>
<td>No</td>
<td>✓ On device</td>
</tr>
<tr>
<td>Push content</td>
<td>No</td>
<td>No</td>
<td>✓ P2P/Group</td>
</tr>
<tr>
<td>Ad hoc Communities</td>
<td>No</td>
<td>No requires invite</td>
<td>✓</td>
</tr>
<tr>
<td>Guaranteed Messaging</td>
<td>✓</td>
<td>No</td>
<td>✓ Bidi, NAT</td>
</tr>
<tr>
<td>Security</td>
<td>✓</td>
<td>✓ Limited: https</td>
<td>✓</td>
</tr>
<tr>
<td>Efficient Bandwidth</td>
<td>No every click=data</td>
<td>No every click=data</td>
<td>✓</td>
</tr>
<tr>
<td>Offline persistence</td>
<td>No</td>
<td>No</td>
<td>✓ On device</td>
</tr>
<tr>
<td>Data to Knowledge</td>
<td>✓ at the server only</td>
<td>No</td>
<td>✓ On device</td>
</tr>
<tr>
<td>Legacy Integration</td>
<td>✓ via webservices</td>
<td>No</td>
<td>✓ Multiple</td>
</tr>
<tr>
<td>Multi-Language</td>
<td>No</td>
<td>N/A (HTML, JS. No OO)</td>
<td>✓ Java, C#, C++</td>
</tr>
<tr>
<td>Self-healing</td>
<td>No</td>
<td>No</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Comparing Services & Features:**

- **Traditional Method**
- **Innovative Method**
Abstraction platforms have emerged that aim to ease the deployment to multiple mobile platforms.

* Due to a change in the Apple iOS developer agreement, apps built using 3rd party platforms and tools may be prohibited from listing on Apple iTunes (consumer apps). Enterprise deployments are still possible.
MOBILE WEB DIS/ADVANTAGES

- **Device support**: wide availability for mobile phones and PCs, however browser fragmentation creates an inconsistent experience. Unavailable for embedded devices.

- **Network efficiency**: increasingly poor performance as wireless devices proliferate. Good for simple applications.

- **Collaboration/Groups**: limited to email/chat and a shared folder on server for file transfer.

- **Security**: good, but unable to manage device remotely.

- **Performance**: relatively slow. HTML and JS are not as robust as Object Oriented languages. Centralization offers relatively slow processing and scalability.

- **Cost**: free and open. Support through user groups or consultants.
ENETERPRISE DIS/ADVANTAGES

- **Device Support**: Limited to specific operating systems for which the code is written/ported. Typically only mobile and PCs running certain platforms.

- **Network efficiency**: Some offer offline input capability, a huge advantage as bandwidth becomes more scarce.

- **Collaboration/Groups**: Limited to email/chat and a shared folder on server for file transfer.

- **Security**: On device security can be achieved, yet vulnerability occurs with collaboration and data exchange.

- **Performance**: Good on-device performance, but collaboration requires servers and a network connection, slowing performance and creating a bottleneck.

- **Cost**: Proprietary solutions are more expensive, but better supported than open source. Cost rises w/ additional platforms.
P2P DIS/ADVANTAGES

- **Device Support** - Designed to support a wide range of edge devices. By definition does not need a central server.
- **Network efficiency** - Offline capability plus intelligence. Turns clients into servers that host/share services autonomously.
- **Security** - (Voyager) Secure Firewall/NAT Traversal support via BIDI same socket/port communication & client initiated requests.
- **Collaboration/Groups** - P2P/Group application messaging, chat and file transfer. Ad hoc community formation/discovery.
- **Performance** - (Voyager) P2P/Group ad hoc messaging. 1.5x faster than RMI, 300x faster than web. OO support (Java, C#, C++).
- **Cost** - Proprietary solutions are initially more expensive, but better supported than open source. Relative cost lowers exponentially with “write once” capabilities.
The next generation of apps will have social or collaborative elements that incorporate data from other applications or third-party sources. Once data can be shared freely, the opportunity to transform data into knowledge or new content and share it with other applications on a device or distribute to a community elevates collaboration to its true potential. This is a leap from traditional collaboration solutions which are limited to mobile email or other messaging, voice and sometimes shared files on a server.

As our world becomes wirelessly connected, a network of devices, primarily sensors and handhelds, will form the foundation of a community that continuously gathers, transmits and receives data, whether it be environmental data, geo-tagged content, or real-time alerts, such as people’s location or communication threads. The community cannot be human-dependent, rather devices will have embedded intelligence that will enable them to act anonymously on new information they receive from within the community.

Next generation applications must support the formation of **P2P** and **P2-Group ad-hoc communities** of devices, as well as the ability to support filtering of messages across these communities. Transactions must extend from the enterprise to edge devices and allow for distributed, but *coordinated tasks among peers, peer groups, and the enterprise*. Support must be provided to allow for intelligent agents running on the edge to participate in guaranteed message delivery with *enterprise transaction managers* such as those provided in .NET’s MTS, Java’s Transaction Service, and OMG’s Object Transaction Service.

---

“**The web is like an encyclopedia, where you always start with the index. Mobile business 2.0 will be more like a theme park where you explore what’s around you. Data will be selectively pushed to the user based on context, matching the users' needs, interests, mood, location and even recent behaviour.**”

-- Nick Jones, VP Gartner

((TECH TIP))
“Mobile business 1.0 (or mobile commerce) failed because the industry tried to transplant all web principles into the phone (e.g. WAP). Truly valuable services have not been widely available and the location based services that we have seen have not been very sophisticated...

((TECH TIP))

Location, combined with personal presence, creates transient communities, which offers an opportunity for better use of time or marketing.”

-- Nick Jones, VP Gartner

#5: LOCATION & DISCOVERY

While your address book will always form the hub of your private and work communities, location and discovery services can extend communities beyond your address book or a check-in venue, to become apps that connect and rally sports fans, exchange photos and communications amongst travelers, discover potential customers and build loyalty among existing ones, and many more iterations on the same theme: ad hoc collaboration. Ad hoc communities exist for a period of time and space and are made possible by taking advantage of IP multicast and location APIs.

Combined with rules engines, such as a RETE-based rules engine integrated with some pervasive platforms, engineers can also weave artificial intelligence and cognitive capabilities into the location-based apps they deploy to all targeted devices and servers. Engineers will be able to produce applications that provide real-time intelligence, situational awareness, and coordination at the edge not found today.

To enable these applications pervasively, your platform must support dynamic discovery via UDP or IMS/SIP and the formation of P2P and P2-Group ad-hoc communities of devices as previously discussed. Finally, all agents need to be accessible by a Service Description in a Yellow-pages directory, ideally one that also supports Universal Description, Discovery, and Integration (UDDI).
#6: BANDWIDTH MAKES OR BREAKS AN APP

While ‘unlimited’ data plans are common among Smartphone contracts, network availability it is far from ubiquitous or guaranteed. App-intensive phones, like the iPhone, arguably consume the most bandwidth and customers are often complaining about poor performance. There is also a great demand to utilize networks more efficiently, as additional consumer devices like eReaders are charged by the kb.

The high volume of users resulting in slow network performance at mobility tradeshows as well as concentrated subscribers in metro areas like San Francisco serve as a microcosm for what our pervasive experience will be when mass market devices become connected. While each network provider is scrambling to expand to 4G networks, the sheer number of devices in our connected world will surely reach a critical mass where efficiency is no longer an option, but a requirement.

Scalability is also a concern at the application messaging level. A centralized server can become a bottleneck for messaging and risks a single point of failure. The most infamous example of this was when Twitter suffered day-long downtime due to a sub-optimal architecture. A solution to this is to consider a platform that offers decentralized communication between groups of devices/systems without the need of a central messaging server.

Efficient network utilization is accomplished by removing the requirement to push all transactions and data to the cloud and pull data back to the device again with each transition or download. This can be accomplished by a peer-2-peer architecture and/or by implementing RULES engines and software agents to perform many edge transactions as possible. Apps should be able to process data at the source, handle unreliable and/or limited network connections, and adjust to hardware failures or CPU load. Therefore, these devices must also persist data via a micro database.

“"The final consideration is whether the user and the company understand two basics about wireless and Internet access: Do they understand that the Internet is NOT a mission-critical network? Do they also understand that wireless connectivity is NOT always available?”

-- Andrew M. Seybold

((TECH TIP))
#7: NETWORK SURVIVABILITY AND MOBILITY

Bandwidth issues and inconsistent network availability lead to the next tip which is that apps need to communicate without uniform standards, in multiple languages, multiple networks, function when offline.

A platform should provide a simple way to access databases, regardless of the type of database, be it relational, object, xml, and a multi-user enterprise database or single-user embedded. Developers need to be shielded from the intricacies that exist with these various flavors and have support for data synchronization between the edge and enterprise.

And when back online, devices must support passing messages over standard centralized messaging servers for integration with enterprise and legacy systems. More specifically, there needs to be seamless integration with Microsoft’s Message Queue (MSMQ), Java’s Message Server (JMS) and other Enterprise Service Buses.

Additionally, the software components or agents running on edge devices need to support multiple wireless protocols (GSM, CDMA, Wi-Fi, UWB, Bluetooth, NFC, RFID) and associated networks (Telco, Wide Area, Local, Personal). Ideally, they will dynamically reconfigure themselves to use a communication protocol that best matches the capabilities of their current network connection and the current node(s) they are in communication with.

To achieve seamless mobility across various wireless and telco networks, support must exist for devices joining and leaving the network, which will result in changing “internet” addresses. The platform needs to be support multiple IP’s of network hopping/firewalled devices/servers either via NAT support, integration with a SIP Server and/or other means. This will allow devices to continue to communicate and share data services.

“Our wireless devices can best be described as being always on and most of the time connected. There is no guarantee of connectivity, nor is there a guarantee that you will always be able to access the information you want and need when on the road if you are connected. Data that resides only in the cloud or behind your corporate firewall is not always accessible.”

-- Andrew M. Seybold

((TECH TIP))
In a ubiquitously networked world, it will often be necessary to maintain data on edge devices. The security concerns facing enterprises today will need to incorporate solutions that extend into a collaborative environment. Applications must provide an extremely high level of security to ensure privacy and protection from rogue/viral clients and software agents. This will involve security agents and agent managers that provide capabilities above and beyond the current encryption, authentication, and authorization that are currently employed in client-server environments.

When implementing communities, offering the user complete control over his/her status and visibility is crucial. Furthermore, community apps should allow separate public/private streams and store data to the secure portion of SD card.

Community apps will need to participate in real-time data sharing with other devices, systems and clouds regardless of the networks that the participating devices and systems are on. Current firewall restrictions inhibit this ability, so the ability for applications to easily and securely span networks is increasingly important. **Support for Network Address Translation (NAT), and the resolution of application clients/services with multiple addresses is key** for solutions to utilize multiple networks and spanning their firewalls. Once apps can communicate securely over the same socket and port, you no longer compromise network firewalls.

“**The simple expedient of placing a Windows PC behind a Nat router changes the equation considerably, giving even unsecured Windows PCs an environment in which they can operate safely...**

**((TECH TIP))**

As long as Windows is the primary operating system for internet-connected hosts (a condition that is unlikely to change any time soon), Nat will be an important part of most users' security perimeters.”

-- Daniel Golding, Computer Weekly
To achieve John Chamber’s vision of “any device to any content wherever it is in the world over any combination of networks”, an advanced pervasive M2M application will need to support most, if not all of the depicted device components.
An advanced pervasive M2M application with an enterprise layer will need to support most, if not all of the depicted components.

#10: UNDERSTAND THE ENTERPRISE ARCHITECTURE
Each technology approach discussed offers a tradeoff between performance, ease-of-development, reliability, cost, time-to-market and differentiating features. The decision criteria relies on finding the best solution for your app’s vision and audience.

Requirements to consider:

- **Web browsers** offer the most widely supported mobile phone support, with limited fragmentation, extending in some cases to feature phones (non-smartphones), but does not support device classes outside of phones. In contrast, smart clients were built specifically to reach a range of devices, from sensor to phone to server. Is there any value to extending your application beyond the phone or PC?

- Compared to smart-clients, thin-client apps are slower, less reliable, support fewer devices and device classes, and in general only gather rather than turn data into knowledge and act upon it. Yet, thin clients are an affordable choice for simple, less transaction-oriented applications that are not mission critical or require guaranteed messaging. Smart clients are highly transactional, autonomous and function offline when needed. What level of sophistication and longevity does your app require?

- **P2P** platforms excel at secure collaboration and maintaining in-state application transfer from device to device. Yet, most low-level collaboration tasks like email and cloud storage are met with a web app. Does your content or data need to be private (P2P platform) or can you use a hosted cloud solution? Do you need to collaborate across organizations outside your firewall?
TAKE-AWAY: 300 BILLION REASONS TO GET CODING

With the second generation of mobile, pervasive apps poised to explode, and with standards in limbo, it’s important to surround yourself with the best-of-breed tools, technology approach and a roadmap that will future-proof your application for the years to come.

Additional Resources
The following are some additional links and resources to assist you with your marketing and business models:

- [Ad Mob Mobile Metrics Reports](#) - insight into purchasing behavior and marketing. They are published monthly.
- [Harbor Research](#) - Pervasive Internet & Smart Services Market Forecast
- [Development Costs](#) - discusses risks, opportunities, and typical development costs

For technical advise on fineing a platform solution that meets your needs, Recursion assists customers with tools, prototype development and consulting services. Visit [recursionsw.com](http://recursionsw.com) for more information.

It is truly an exciting time for our industry. It will be interesting to see which of today’s mobile app players translate their technology and business models to the pervasive, ubiquitous, M2M, internet of things. Will you be one of them? ■
ABOUT THE AUTHORS

Victoria Romero-Gomez, Product Manager  
vgomez@recursionsw.com  
Vicky has worked at technology start-ups for the last 10 years, including MadetoOrder.com and NextBill.com, where she held various roles including Director of Marketing and Project Management roles. She has led the marketing department at Recursion since 2006 and she sets the company’s strategic approach to new verticals.

Vicky holds an MBA from the University of Texas at Austin’s McCombs School of Business with additional post-graduate work at Southern Methodist University’s Cox School of Business, and a BA from the University of California Berkeley.

Bob DeAnna, CTO Recursion Software  
bdeanna@recursionsw.com  
Bob has more than 24 years of experience in distributed frameworks such as JEE, CORBA, .NET and ATMI, has authored or co-authored multiple patents in this area. He is also a contributing writer and thought leader on mobile technology for publications like Embedded Computing and .NET Developers Journal. Bob is a visionary who has worked to achieve the technology behind community-based collaborative networks and is currently shaping the product roadmap at Recursion towards this end.

Previous roles include VP of architecture, senior consultant and lead engineer for companies like Verizon, ATT, Swiss Re-Insurance and State Farm Insurance. He holds a BS in Mechanical Engineering from Rutgers University, with post-graduate work in C/C++ and Unix Programming at New York University.