James Mateer

12303 434th Ave SE, North Bend, WA 98045
(425)503-2836 Jim@JimMateer.com

Note: this is an expanded version of my resume. A version suitable for machine scanning can be found at www.JimMateer.com

Professional Summary

A Technical Program Manager (TPM) is required to have one foot solidly in the Management domain, and the other solidly in the Development domain, and be ready to be an active participant in either or both. There are many solid managers that do not have a good grasp of the tools, technology, and platforms available, and there are a lot of good, solid coders that will craft good code, but lack the ability to see the end to end picture. The melding of these two skill sets is what good TPM's are made of.

Managing programs or product development requires a core set of skills that are typically developed and perfected over time through experience. The following section describes these core skills, as I have learned them, during my professional career.

- **Android Development and Deployment**

  After Microsoft killed the .NET Micro Framework, I started looking for what would fill the void it left. I felt the mobile device opportunity today shared many similarities with the infant personal computer boom of the early 1980's. I believed, and still do believe, that mobile devices are the wild wild west of technology growth and expansion today. I was sure somebody was going to come along with a platform that would unleash the creative talents of hundreds of thousands of programmers, all vying to develop the "next coolest thing".
  Google adeptly filled this void with Android. Google took a play out of the early Microsoft play book by providing a full-featured, solid, robust development platform, with state of the art tools and services, for free (think MS-DOS and tools, circa 1984).
  Android for mobile devices, coupled with Cloud services (Amazon Web Services, or MS Azure) make a powerful and compelling platform, and was a natural choice for my LogIT Commercial Driver logging and tools system.

  - Key Take-aways:
    - Pros: Affordable, Robust, Large and established Support infrastructure
    - Cons: Young, diverse (many different implementations), rapidly changing.

- **My philosophy on Functional Specifications**

  I approach functional specifications as a process, rather than a document. The functional specification document grows out of this process. The functional specification process starts with an idea, and progresses ahead of the development cycle.

  To wait until the functional specification is complete before starting development on a product is a guaranteed way to ensure your product never gets developed.

  Likewise, to begin development on a product, without a clear understanding of what the product is, or does, with an end to end picture, is a sure way to waste development resources.

  The art of the functional specification process is to find the sweet spot where the process is far enough ahead of development to provide guidance and direction, but not so far ahead that it can't remain nimble, and able to adapt as domain knowledge increases.

  Functional specifications define what a product will do, and are generally responsible for product success or failure. A well engineered functional specification will drive the product development process.

The functional specification process will:

- Focus the development efforts
- Provide guidance in feature prioritization
- Provide a roadmap to test
- Describe what "Done" means in the context of the product and its release cycle.

Often, Functional specifications are created as an afterthought, perhaps as a part of the product documentation. This invariably results in a poorly planned product, and a sub-standard functional specification. The effort required to fix the product and specification after the fact is most often greater than the effort required to scrap everything, and start anew.

As a Microsoft Program Manager, I managed the specification process, and developed solid functional specifications for:

- Several releases of NDIS (the Windows Network Driver Specification).

  The scope of this was enormous. Without NDIS, there would be no Networking in Windows. Without Networking, there would be no Internet, no Cloud, no corporate networks.

  NDIS is not the only networking component, but it is an essential component that makes it possible for third party network adapters to work with Windows.

- The WHQL Driver Test Manager (DTM)

  My experience with third party developers taught me that code developed outside of Microsoft's control could affect the stability of the Microsoft platforms, and thereby, the brand reputation.

  It was clear to me that it was in Microsoft's best interest to make sure third party drivers and core components were vigorously tested and certified before being allowed into the Microsoft platforms.

  The DTM was designed as a platform that would give third party developers the tools necessary to properly test and certify their components.

- The Networking and DPWS components of the .Net Micro Framework (.NETMF)

  .NETMF was a great idea, at the right time. Embedded developers could use .NET to write code for very small devices (as little as1MB flash). This was before Android. With .NETMF, Microsoft had a managed code solution for any size embedded device.

  My responsibility was to add networking, and a standard communications capability to the platform. I accomplished this by contracting for an existing embedded TCPIP stack, and contracting a developer for a DPWS stack.

  Microsoft cancelled the .NETMF project in 2009, citing an emphasis on cloud services over platforms.

- <u>DRM Breech Management data repository</u>
  Microsoft and Digital Rights Management (MSDRM). In addition to developing safeguards and locking technologies to prevent the copying of copyrighted material, they also began developing a network to detect the illegal copying and sharing of Microsoft protected material. This copying and sharing was termed a breech in Microsoft nomenclature. I was contracted to design a data repository to store and allow analysis of, the breech related data collected by this network

- <u>Drawbridge, a light weight VM technology</u>
  Microsoft Research (MSR) is a great place, where wonderful ideas bounce off the walls like sunbeams on a warm spring day. Occasionally, one of these ideas will germinate, and grow into something that might be product worthy. Drawbridge was such an idea. I was contracted to help move Drawbridge from MSR into a product group.
  Unlike a well defined product development effort, Drawbridge was developed by a few bright guys, sitting down and pounding out code. Whereas I really liked the idea (think of MSIE running in its own sandbox. You get malware, you kill off the sandbox, and malware is gone), the implementation was somewhat haphazard. This was a classic example of the functional specification process lagging the development process. I did manage to get MS Product Group interest, and, in fact, even moved the technology into a product group. Unfortunately, the lack of a solid design foundation eventually caught up with Drawbridge. It lacked the stability necessary to make it a shipping product.
  - Key Take-aways:
    - Short cuts will ALWAYS cost you in the end.
    - 1st revisions of any product are almost guaranteed to suck
      - Products grow, improve, and evolve
    - Developing solid, well designed products requires solid, well designed engineering practices.

- <u>**Third party relationship management**</u>
  Third party relationships typically fall into one of 3 basic buckets:
  - The third party has something you can use, such as a service or component. Typically, the third party is compensated for providing the service or component, and some level of on-going support. Properly managing this relationship is really about mitigating risk. This is particularly important because often your product will rely on code outside of your immediate control. A good example of managing this type of relationship was my contracting with EBSNet.inc for an embedded TCPIP stack for the .NET Micro Framework. By contracting for an external stack, rather than developing one in-house, I was able to shave at least two years off of the development cycle. Risks were mitigated by:
    - Requiring EBS to provide buildable source code to a Microsoft Vault. This source would only be accessed in the event of a breach of contract.
    - Designing a pluggable architecture that would allow the EBSNet stack to be replaced by an MS stack when available
    - A solid contractual support obligation from EBSNet.
    - A contract term, with sufficient options, that would provide Microsoft ample time to develop an internal stack.

  - A second type of third party relationship is where the third party expands or enhances your product, to the benefit of both parties. Consider the case of a video card manufacturer. The manufacturer stands to be much more successful if his video card works with Microsoft Windows, and Microsoft benefits from being able to support a wide range of video cards. I have broad experience in managing these types of relationships:

- As a network driver support engineer, I was not only instrumental in the success of many network adapter vendors, but was also in the optimal place to see what worked well, and what didn't, in terms of operating system support.
- My experience in working with third party vendors, and the operating system designers, made me an ideal choice to drive the development of a Driver Test Manager, a set of tools to help third party vendors certify their drivers.

- The next third party relationship consists of those who can affect the development or release of your product, yet have no financial gain or incentive. These are typically the relationships with law makers or regulators. As a Program Manager for a large, multinational corporation, I was responsible for understanding the requirements of these regulators, and certifying that the products I was responsible for met or exceeded all requirements. Products that required my certification of compliance included:
    - The Microsoft DDK
    - Many versions of NDIS
    - The WDK with the DTM
    - The .NET Micro Framework and its externally supplied components
    - Drawbridge betas

    - Key Take-aways
        - The most important part of managing a third party relationship is to have a good grasp of who is the seller, and who is the buyer.
        - Managing third party relationships is more about managing and mitigating risk

- **Industry Evangelization of Microsoft products and platforms**

  When developing platforms, such as NDIS, the WDK, or the .NET Micro Framework, it is important to evangelize the platform to the people you want to use it. This is usually accomplished in several ways:
    - Through presentations at major trade shows. I have presented multiple times at all major trade shows for the technology I supported, including:
        - WINHEC
            - Windows Hardware Engineering Conference
            - Both in the US and Abroad
        - NCC (Networking Communications Conferences)
            - US and Abroad
        - ARM Developer's Conferences
        - TechReady
        - TechEd
        - EmbeddedWorld
            - US and Abroad
        - CEBIT – Hanover
        - Makers Faire
    - By creating and hosting early-adopter events
        - I have created and executed Alpha, Beta, and other pre-release programs for:
            - NDIS
            - DTM
            - .NET Micro Framework

- Drawbridge
- Key Take-aways
  - If your product is good, and fills a need, evangelizing it will be easy.
  - If you are having trouble getting traction for your product, you should re-evaluate your product.

- **Product release cycle management**

  The Product Release Manager is responsible for ensuring:
  - All components of a product have been properly vetted and meet all applicable requirements.
  - That all team members have met all product requirements
  - That the product is thoroughly tested , and test has signed off
  - That the product meets the functional and design goals.

  I have been the release manager responsible for:
  - Several releases of NDIS
  - DTM
  - .NET Micro Framework
  - Drawbridge
  - Owning product features, customer needs, and developing and defending product roadmaps
  - Managing SDLC implementation including schedule, objective and resources to ensure the right product is shipped at the right time with the resources available.
  - Ensuring all legal, security, quality, and compliance requirements are met

Experience

Dates of Employment
6/2012-Present

Shale Mountain Software

Founder

www.shalemountain.com

I founded Shale Mountain Software in 2012. The Shale Mountain mission is to target niche markets with mobile and connected apps.

As technology improves and expands at an ever increasing pace, many bold new adventurers are pounding forward, developing new, heretofore unimagined ways of transforming life as we know it.

Examples abound, like Facebook, Twitter, LinkedIn, EBay, Craigslist, Angie’s List, and so many more.

In this mad rush to invent the “next big thing”, so many smaller, yet hugely beneficial applications of technology are being bypassed, “left on the way-side”, waiting for someone to come along and sweep up the crumbs.

Because Smart Phones are becoming ever more ubiquitous, and connectivity is to be expected and available, Shale Mountain’s focus is to leverage these trends, using existing and established technologies, to solve every day problems.

Shale Mountain’s launch product is the LogIT System (www.eLogIT.net).

LogIT consists of an Android application, Cloud Storage, and a Web Portal designed to give commercial truck drivers' tools to help them perform their job. LogIT includes :

- Electronic Logging (Hours Of Service logs),
- FMCSA compliance calculators
- Android based DashCam,
- Log storage and archiving meeting DOT requirements, and
- Web based Log viewing and analysis tools

Shale Mountain Software also provides Engineering services for SmartMarineTechnologies (SMT. SMT, founded in 2005, is the marketing company for SmartTrim (U.S. Patent #7311058). SmartTrim, designed and developed by me, is a revolutionary embedded controller for small power boat trim tab control. By using GPS, accelerometers, and electromechanical controls, SmartTrim improves handling and fuel consumption in power boats from 17' to 100' in length.

Populus Group

Dates of Employment
5/2011 – 6/2012

Senior Program Manager – Drawbridge, Microsoft Research

Drawbridge was a promising technology that came out of Microsoft Research. The general idea (really a rehash of an old idea) was to package an application along with the parts or the operating system that were required by that application, into an atomic unit that could be run, as its own sandbox, in the host operating system. I was contracted to help move Drawbridge from a research project to a product group. Like many research ideas that morph into products, Drawbridge did not have the solid engineering foundation required to make a sustainable product.

This was a case where I was tasked with generating the engineering documentation and processes necessary to make Drawbridge a product, and then convince a product group to adopt and own the technology. I was able to get Drawbridge adopted by the Microsoft Azure group, but was unsuccessful at getting Drawbridge to a point of stability and compatibility necessary of a shipping product.

Although I was unable to realize the ultimate goal of turning Drawbridge into a shipping product, I do consider this a valuable learning experience. Drawbridge is clearly a case study of what will happen if good engineering practices are not followed, regardless of the skill level or talent working on the product.

Populus Group

Dates of Employment
4/2010 – 10/2010

Senior Technical Program Manager – Microsoft Media DRM Breach Response

Microsoft has a presence in Digital Rights Management, or DRM. Microsoft's offerings in DRM are typically known as MSDRM. After several cycles of fortifying DRM processes, seeing them hacked,

and fortifying them again, the MSDRM team decided to try and learn more about, and to thwart the people that were circumventing the DRM protections. There were efforts on several fronts, including the removal of sharing web sites, and the prosecution of those caught illegally sharing. One of these efforts involved collecting enormous amounts of data on hackers, hacker groups, websites, and networks. Much of this data collection was farmed out to specialized security firms. The result was hundreds of megabytes of data, each month. I was contracted to develop a method to receive, store, and provide analysis tools for the data received each month
The system I arrived at was based on SQL Azure, using Azure Analytics (not available at the time, but scheduled to be ready when we were ready for them). The solution I proposed was relatively inexpensive. All data storage was in house, as was development of the initial prototypes. The storage, using Microsoft Azure was safe, secure, and infinitely expandable. Access was global. And processing power, storage and bandwidth were easily expandable to meet any future need. In my opinion, this could have been one of the largest, most comprehensive data repositories of people behaving badly with computers, outside of the NSA.

Information from this system could be used to stop copyright infringement, prevent malware and virus attacks, reduce spam attacks, and a host of other benefits.
In the end, the MSDRM team elected to go with a flat text file and 'findstr'.

Dates of Employment
Microsoft 6/2006 – 7/2009

<u>Senior Technical Program Manager - .NET Micro Framework</u>

The .Net Micro Framework (NETMF) was built from extending technology originally developed for the 'Spot' portable devices and MSN Direct, .NETMF was beautifully positioned to become the de-Facto operating system for small and mobile devices. NETMF was .NET, with world class development environments and tools, for devices with less than 2MB of flash memory. NETMF was the perfect platform for everything from very small smart devices to low end mobile phones and tablets. It was the perfect low end anchor in Microsoft's suite of computing platforms that included WinCE, Windows Embedded, the desktop OS's, the Server business, and the emerging cloud technologies (Azure). With the addition of the NETMF, Microsoft had an OS solution for any computing requirement, no matter how large or small.

I was a Senior Program Manager for the DTM when I first learned of the .NET Micro Framework. Because of my background in embedded devices, and my interest in the growing mobile markets, I was very interested in the platform. After learning more about the platform, and talking to the team, I was offered, and accepted a job as a Senior Program Manager on the .NETMF team.

When I joined the team, the .NETMF didn't yet have any networking capabilities. Resources were somewhat limited, so any in-house development to add networking would take too long. I began looking for external resources that could be used to give the .NETMF a networking capability. After evaluating several offerings, I selected EBSNet to provide the TCPIP v4 network stack, and Exceptional Software to develop a Web Services stack using Device Profile for Web Services (DPWS). Adding a fully functional Networking and Web Services stack to the .NETMF took less than 6 months.

NETMF's downfall was largely due to Microsoft senior management's lack of interest in emerging mobile and small platform opportunities, and a singular focus on cloud based scenarios. This narrow focus and lack of interest resulted in the cancellation of the NETMF effort as a product, and the disbanding of the NETMF team.
It is of interest to note that Google did not share Microsoft's lack of vision, and forged ahead with the closest competitor the .NETMF had, the Android platform.

Dates of Employment
Microsoft 3/2002 -6/2006

Senior Technical Program Manager –Windows

In the early to mid 1990's, Microsoft was being criticized by press and competitors for the frailty of its operating systems. Some of this criticism was due to the frailty of the hodge-podge OS called Windows 95 (formerly Windows 3.x), some of the criticism was un-deserved (think Apple), but the lion's share of the frailty (and hence, the criticism) was due to third party device drivers. Drivers are a core component of the operating system, yet are developed outside of Microsoft. Drivers are that small bit of OS code that connects the Microsoft Operating System to somebody's piece of hardware. Unfortunately (for Microsoft) if a driver crashes, more often than not, the OS crashes. The user sees the Blue Screen of Death (BSOD), and automatically blames Microsoft.

In my beginning days at Microsoft, I was part of the driver development support team, helping hardware vendors write their drivers for the Microsoft platforms. Originally, there were two people responsible for developing and releasing the Driver Development Kit (DDK), and a handful of Support Engineers to support the DDK. I was a key part of a successful Support push to convince Microsoft senior management (Gates and Ballmer) to expand the DDK effort. In 2002 I was recruited to join the expanded team (now called the Windows Driver Kit team, or WDK). By 2002, the team designing the DDK and certification tools had grown from 2 to over 50.

My contribution to the WDK team was to drive the development of tools used by 3rd parties to test and certify their drivers for the Microsoft OS's (Windows Hardware Quality Labs, or WHQL).

Dates of Employment
Microsoft 4/1997 – 3/2002

Technical Program Manager - NDIS

My first job at Microsoft was helping 3rd party developers write network drivers for their network interface cards (NICs). In Microsoft operating systems, NIC's interface with the host OS through a software layer called NDIS (Network Driver Interface Specification). It was easy to see, from my time in Support, that there was room for improvement in NDIS, and in the tools used to develop NDIS drivers. This was at a time when networking was making this huge migration out of corporate offices, government institutions, and universities, and into the living room. These were the early days of WiFi, BluTooth, and PLC. This was the adolescences of the Internet we know today. This was my opportunity to make a difference on a global scale, and I eagerly grabbed the brass ring. As NDIS PM, I drove the evolution of NDIS through 5 years of the fastest and most comprehensive growth possible. I made sure NDIS supported new technologies, often before those technologies were released. I helped make sure NDIS was safe and secure in this new, 'wild west' environment called the Internet, and I made sure hardware vendors, and independent developers had the tools and resources necessary to develop quality drivers for their products.

Microsoft

NDIS Support – Support Engineer, 1995-1997

My first job at Microsoft, I signed on to help external developers write NDIS drivers for their hardware. Part of my job was to make sure these external developers had what they needed to do their job. Early on, I identified several holes in the Microsoft offerings to these developers, and worked hard to plug those holes. One such hole was the inability for third party developers to write an intermediate driver (or shim) that fit between a NIC with its NDIS driver) and the protocols layered above. This was initially viewed as a security risk by the networking team. Because of the number of companies that were writing antivirus drivers, or sniffers, or filters, or a host of other uses, I lobbied (successfully) to change the networking team's mind, and support an intermediate driver model. I also developed and released an NDIS Intermediate driver sample (IMSAMP) to fill the gap between its release and the next OS release officially supporting intermediate drivers.

Another serious problem I addressed, while supporting NDIS, was the serious lack of resources allocated to developing a device driver kit (DDK) and tools third party developers needed. As the lead of a team formed to address this problem, we were able to convince senior management that this was a problem that needed fixed. Today, the WDK team responsible for the tools, kits, and certification processes numbers well over 50.

Space Industries and Telecommunications (SITEC) 1993-1995

I was an Embedded Development Engineer responsible to the Temperature Measurement Unit on the Wake Shield.

The Wake Shield was a joint NASA/University of Houston project. The idea was to launch a 16' concave aluminum dish into low earth orbit (LEO, the domain of the Space Shuttle). While travelling at 17.5kMPH, this dish would produce a vacuum in its wake, purer than could be economically produced on earth. In this wake (on the back of the wake shield) gallium arsenide guns would shoot at silicon wafers, producing much finer substrates for the development of integrated circuits (IC). The wafers would then be harvested, returned to earth, and turned into IC's. This growing process required strict temperature control, which was handled by an embedded controller of my design, running my software, to control the cycling of on-board heaters.

The Wake Shield was a great project, and a lot of fun to work on. Its demise was due to a number of factors, such as the demands of the shuttle program, and the invention of more efficient, terrestrially based vacuum technologies.

Communication and Data Systems Associates (CADSA) 1986 - 1993

A small telecommunications company in Webster Texas, we developed distance learning tools. The idea was that small rural schools would often have 1 or 2 really exceptional students, but could not afford full time instructors for the advanced classes that would benefit these few exceptional students. Our technology, along with an affordable subscription, would for example, put a calculus instructor in front of a camera, to be broadcast to these gifted students in many locations. We also provided the technology (pre-internet days) that allowed the students to participate in the class by responding to questions and quizzes. My role was developing the small, embedded response devices, the head end systems, and the data paths that connected the two. These data paths used a combination of Telephone and custom satellite data paths developed by CADSA.