Agile, **DevOps**, & US Fighter Pilots

Steve Naidamast, 15 Feb 2016 CPOL

3.67 (3 votes) Rate this:

Russian PAK-50 – Now Considered the World's Greatest Fighter Jet "Agile" & "**DevOps**"; Two Sides of The Same type of Coin In spite of the promotion of the "Agile" development lifecycle, it is a highly flawed technique, which often succumbs to being nothing more than what is known as "controlled chao

Editorial Note

This article appears in the Third Party Products and Tools section. Articles in this section are for the members only and must not be used to promote or advertise products in any way, shape or form. Please report any spam or advertising.

Russian PAK-50 – Now Considered the World's Greatest Fighter Jet

"Agile" & "DevOps"; Two Sides of The Same type of Coin

In spite of the promotion of the "Agile" development lifecycle, it is a highly flawed technique, which often succumbs to being nothing more than what is known as "controlled chaos". This is exactly the term that naval aviators use to describe landings on an aircraft carrier; a technique filled with so many dangers and variables that if but a single issue occurs the pilot and craft could be both done. Modern computerized systems aboard such ships have lessened the dangers to some extent by incorporating remote control flight to the planes. However, again if anything goes wrong with the software or the hardware, the pilot is left to land the craft him or herself making the reliance on such software an added danger to pilots in such circumstances. Using such software regularly, the pilots lose their hard learned skills in landing such aircraft, if modern training has taught them such hands-on skills in the first place.

"Agile" has no features or advantages that mature software engineering practices have not already devised and demonstrated

successfully across a breadth of different types of software projects. It is merely a methodology that allows developers to escape the necessities of good project implementation and the fact that so many “Agile” projects still incur some level of failure in their endeavors is a testament to this contention.

One of the biggest failings of “Agile” however, is the idea that a developer can do it all following on the trends in the early 2000s that began with businesses eliminating crucial departments that supported the software development process (ie: Quality Control). Between the outsourcing and the reductions in staff, IT organizations were unfairly left to themselves to devise methods to keep the organizations afloat despite dwindling resources. This was the catalyst for such concepts as “Agile”.

However, it was already well known by this time that developers had more than enough on their plates than to be able to start handling increasing technical responsibilities.

In the 1990s before the onset of outsourcing was in its full throws and the Internet had become the accepted medium for enterprise development, it was believed that developers could not only assume the role of developer but that of the Local Area Network (LAN) specialist. This implementation of such a job-mix was attempted and short-lived due to the massive knowledge bases that LAN specialists had to contend with on their side of the fence.

Senior developers wrote about such foolishness and confirmed that no developer could adequately acquire and handle two different but enormous knowledge bases. As mentioned, this attempt was short-lived.

However businesses were not done in their attempt to reduce costs by trimming down IT organizations and the next idea to be floated was that developers “should know the business”. The question is why? Developers are technical personnel and they are not there to “know the business” but instead to implement what the analysts, who are supposed to know the business, provide them. By now such a concept may seem alien to younger readers of this piece. But really, what is “knowing the business” going to do for a quality developer except be able to act as a business or systems analyst? However, that was gist of being able to cut those levels in technical staffing.

Nonetheless, this idea took hold and allowed businesses to trim their business analysts departments while eliminating system analysts leaving IT organizations to resolve these losses on their own.

The result was a bunch of emerging technical managers demanding that developers now have some in-depth knowledge about business processes. And the game was afoot allowing “Agile” promoters to incorporate this delusion into their own proposals. What they seemed to have ignored is that all businesses operate along the same paradigms. It doesn’t matter if they sell insurance or toys. True, the cultures may be different but the underlying processes are all the same.

In terms of technical refinement, “Agile” promoters love to tout the idea that it is a replacement for the “Waterfall” approach in software development. However, as many senior engineers have written, “Waterfall” was but one approach to such endeavors. However, “Agile” promoters couldn’t get past this idea and promoted it to the point that many now believe it. And given this emphasis, one could easily understand “Agile” as a singular replacement for the “Waterfall” approach but then the question is, “What is the replacement for all the other life-cycle models that have been used over the years?”

Now that “Agile” was now overwhelming many IT organizations with it’s adoption, such technical evangelists began thinking of a new way to merge the rest of the IT organization into the software development environment. What resulted was a concept that was believed to be an answer to better “communication” between the different personnel as well as increase the speed at which applications could be updated or released to production. This new paradigm is now known “**DevOps**” and the same marketing hype that surrounded “Agile” is now making it’s rounds for “**DevOps**”; the combination of the two as a way to solve all of the issues that IT organizations have been known for. This has generated the chimera of the “full stack” developer, which simply means that now a developer is expected to do it all. What else could it mean?

One of the main concepts that both “Agile” and “**DevOps**” share is that of closer collaboration through communication. Communication used to mean for software developers, developing project plans and quality requirements gathering. So what is this supposed to mean since “Agile” doesn’t really encourage or emphasize these two critical aspects of software development?

“**DevOps**” appears to mean that you are now mixing such operational processes such as Quality Control and “Release Control” (among the various other types of processes that operations sections handle) into the software development environment. This can only add to the pressures and stresses on developers that they are already under from the constant delusion from technical management of having to speed up everything ever faster as each year goes by.

To begin with, there are very sound reasons for the compartmentalization of various functions in the software development and implementation processes.

On the one hand, a company that is publicly traded must undergo auditing inspections on a regular basis and unless someone has changed the SEC statutes, combining operations within software development organizations simply won’t get a company through such an audit. Instead, it would most likely get such a company heavily fined for “comingling“ critical processes. The reason why such audits have these requirements is to formalize impartiality towards the production environment.

And for all the blather from “**DevOps**” promoters about developers having better access to production implementation processes, there are also laws in public companies that restrict this type of access for developers for the same reasons just noted.

These restrictions of course do not apply to privately held companies, which startups and small business usually are. However, no matter the company type there is a good reason why development staff should be kept separate from any form of mixing with operations. And this is because it makes such environments more conducive to having developers take on additional responsibilities allowing management to see an additional opportunity to reduce staff. This happens all the time.

With the software profession rapidly adopting the "Agile" ideology adding this new ideology of "**DevOps**" is sure to make things even worse in development organizations that are already suffering under pressures from the general chaos that "Agile" promotes.

However, younger technical professionals seem not to understand this since they have no other experiences in environments that worked successfully without "Agile" to compare the current avalanche of nonsense. Instead they see better access to their process needs a seemingly good thing without understanding the drawbacks.

"**DevOps**" promoters, like their "Agile" counterparts, all tout the idea that one of the major underlying foundations of this new paradigm is "communication" between the different types of specialist in such environments. So let's then understand this; with email, land-lines and a raft of smart-devices all available to developers in today's technical environments they now need to be in close proximity to those people who actually do the software implementations into production.

The other side of this coin is that "**DevOps**" with it's encouraging operations personnel to be closely aligned with the software development organization, developers will supposedly find it even easier to implement their code into production with this additional access to these people. The question is how?

No matter how you slice it, operations personnel are still responsible for implementing completed software into production; at least until management figures out a way to get rid of these people as well. But here is the kicker as we are not just talking about software implementers but Quality Control personnel, "Production Control", "Desktop Support" and the management of the applications that are used to support all these processes as well.

If you read between the lines of a lot of this hype, there is an underlying suggestion that software developers can do their own Quality Control, which has already been scientifically proven to be impossible.

Over the many years that our profession has existed a lot of research has gone into the capacity for software developers to eliminate their own defects. The combined results of such studies have clearly shown that developers can at a maximum find approximately 60% of all defects that may have entered their code. 60%! That's it. And the simple reason for this is because developers are too close to their own work. It is not a criticism but a simple fact.

The new ideologies also appear to hype the idea that with better tools developers will be able to eliminate a greater amount of defects on their own. I wouldn't count on that. Just learning the complexities of all these new tools will introduce their own issues into the mix.

It seems that such ideological evangelists appear to believe that tools and products are the way to make software development faster, more accurate, and more efficient. And yet can they be entirely blamed for this when all of US society in particular has become enamored with technology as the answer for everything? Have a requirement; we have an app for that. Using one's brain for even sedate functions such as reading a map is becoming a lost skill.

Nowhere is it ever mentioned that software developers have to do what they do best, which is technical design of applications and systems as well as code software.

So let's see how these type of ideologies worked out in the real world using an entirely different profession that implemented the same type of short-sighted ideologies for this profession as "Agile" and "**DevOps**" both promote for software development.

US Fighter Pilots

The real-world example is the United States Fighter Pilot training and mission capability across the Air Force, Navy, and US Marines. The results from similar ideologies that infested these areas of our Armed Forces have been horrendous.

Being a fighter pilot is a specialty unto itself just like software engineering is. Though the professions have nothing in common with each other, the skills that both types of professionals require are at the same level of complexity in that both fields have enormous knowledge bases to contend with and both must deliver quality to either survive in air combat or reduce the costs of ownership to business organizations while also maintaining their computerized processes efficiently.

Though the delivered results are of course different for each profession, the underlying factor here is the quality of what is being delivered, which either promotes the longevity of a pilot or the longevity of a business organization.

The US Army Air Service, which became the US Army Air Corp in WWII and the US Air Force in 1947 has never produced a good share of top combat pilots while both Allied and enemy Air Forces have always demonstrated an ability to produce far more highly capable pilots during all periods of engagement. Similar results have been shown for US Naval and Marine Corp aviation.

For the United States Air Force in particular there are two specific reasons for this failing. Starting with WWI, the interest in US

combat aviation capability was always frowned upon by the US military bureaucracy. Prior to WWI there was very little interest in investing in air power technologies. Once WWI began and the US became in engaged in 1917, US Army Cavalry officers became the foremost detractors of developing combat aviation out of fear that their own roles would be diminished consider that up to that time Calvary was always considered an elite unit.

This led to the US not being able to field a single plane developed in the US for combat duty. The result was that the US developed only the JN-Jenny two seat trainers, which later became the common plane in use for barnstorming fame and the US Air Mail Service in the 1920s. The US did receive a license to build the two-seat bomber, "DH4" from the British firm of Airco during the war for which these variants did see combat duty in Europe. For fighter aircraft, the US was initially provided with Nieuport 28s, a rather ineffective fighter when placed against equals or better equipment until later when the Americans were provided with British SE5.a aircraft and French Spads.

Thus for all of the innovation done with air power during WWI by other nations, the US remained far behind and acquired little knowledge from it's own allies to develop its own combat aviation capabilities.

The ignoring of the needs of the fledgling Air Service after the war was maintained by US Cavalry Officer Corp as well as the drawdown of funding for the US military in general since it was US tradition to disband the extended units created for servicing a conflict. Both of these situations remained in place up through the beginning of WWII when it became evident that the US would have to develop its air power capabilities and recruit pilots to develop an aviation arm.

When the US entered the war it's top of the line aircraft were the Army's P-40 Warhawk of Flying Tiger fame and the US Navy Wildcat fighter, both capable planes in the hands of skilled pilots who knew how to use their best traits. The P-40 was durable that it remained in production throughout the war. Lesser known and derided by pilots for it's difficult handling was the emerging long-range fighter, the P-38 Lightening.

Again however, the overall drag on US air power development remained evident even after the initiation of hostilities with US. It was further demonstrated when General Curtis Lemay, the commander of the US Air Force in WWII, allowed unarmed US bombers to enter into combat in Europe without proper escort fighters, though one was available; the P-38 (The P-38 had the capability and the endurance to support such bomber missions but it was tough plane to fly so pilots who weren't provided in-depth training in it avoided flying it leaving few pilots to do so). The result was the dramatic loss of US bomber aircraft and air crews up through 1943 until the famed P-51 finally entered service.

Again, few American fighter pilots emerged as part of the top echelon of pilots during the war when compared to Britain, Russia and Germany. Though the US did produce quite a number of fighter squadrons with some excellent aircraft their ace-making capabilities remained below average in terms of excellence due to the ideologies that had been fomented by the earlier US Army Calvary, which encouraged the view that the Air Corp was there to service the Army.

Though this type of ideological issue could be found in all of the Air Forces that engaged in both world wars, it was only the US which maintained this short-sightedness while the other Air Forces more quickly shed the limitations placed on them by such conservative thinkers, which allowed for better training of pilots.

Even Canada, the US' northern neighbor, consistently produced better pilots in general and more aces.

Whatever bright moments the US Air Force had in WWII it would be once again be contained when General George Marshall shortly after the war ended imposed an "Agile" like ideology on all of the US air arms, which culminated in the "up & out" culture throughout the services that is still pervasive today whereby pilots are forced to consider their careers by fulfilling the requirements to move up instead of what they are supposed to be doing, which is flying fighters. Not doing this meant that pilots and others such specialists were forced out of their military careers altogether.

In the case of fighter pilots, Marshall wanted his pilots to assume a plethora of military skills to prepare them for any and all eventualities that may arise during the Cold War with Russia that was just beginning to heat up.

What did this edict do? Well, fighter pilots are there to train for and fly in combat or as they get older use their experience in air operational positions keeping their huge knowledge bases and competencies as well as the best of the best within the military aviation communities. Marshal's plans put fighter pilots everywhere else except where they were supposed to be. Fighter Squadrons would simply have to do with less capable resources.

Similarly, the results of US military aviation in the later Korean War turns out to have been fairly dubious at best when all of the statistical research has been evaluated, since US pilots could not compete on an equal level with their counterparts in Allied Air Forces or their opponents. And subsequent fighter exercises with other nations after the war demonstrated that US pilots simply did not have the fighter skills their foreign colleagues were provided with.

By the late 1960s Navy air operations officers were seeing the detrimental trends that had been placed on their pilots as a result and decided to create the Navy Fighter Weapons School or what has become more popularly known as "Top Gun". The school's program was designed around the same techniques that the US Air Force had come up with for their own fighter school in the 1950s.

The "Top Gun" pilots however, were selected from the top one percent of naval aviators disallowing other good pilots from attending thereby preventing a wider knowledge and competence base within the naval aviation community. In addition, "Top Gun"

pilots, like their Air Force counterparts, came under a new "ideology"; that of missile-only fighter training since senior aviation commanders, weapons manufacturers and analysts had come to the conclusion that the era of the "dog fight", the very thing that fighter pilots were supposed to learn, was over as a result of the more modern missile technologies now being implemented. Well, no other Air Force on the planet forgot any of the valuable lessons of that "era" and maintained in-depth training for such types of fighting and used them consistently in conflict operations.

The results of US military aviation in the Vietnam conflict were actually not much better than that of Korea despite the new top training of the "Top Gun" pilots in the Navy and similar training in the Air Force. Soviet fighter pilots have been actually found to be much better than their American opponents due to the increased in-depth training provided by the Soviet Air Force.

Since that time US fighter pilots, no matter the advanced aircraft they were flying, have never won an international competitive exercise against any foreign competitor simply because they were denied the ability to maintain and extend the valuable lessons of experience obtained in actual combat simply as a result of ideologies.

With Marshall's original edict still in play, US fighter pilots serve an average of four years before they must rotate out to a different type of position to fulfill their career goals. This leaves a paltry knowledgebase for new fighter pilots coming into the services while diluting their overall capabilities to a bare minimum.

"DevOps"

Has anyone noticed the parallels here with the current state of the US software profession? Marshall's "ideology" is in essence the same as that of "Agile" where developers must assume different roles in order to remain functional in such environments. Like their counterparts in US military aviation fighter squadrons, software developers today are asked to take on undue burdens that do not let them concentrate on what they are actually there to do, which is design and develop quality software. This was initiated with the business "ideology" of outsourcing, which began in the mid-1990s and led companies to expect more out of their remaining developers, while first the "Extreme Programming" adherents and then later those of "Agile" fostered this trend within the software profession itself.

Instead of resisting such pressures through the use of proper negotiation techniques with management that could have developed better outcomes for scheduling development timelines while maintaining necessary and vital functionality across technical departments, the people who proposed "Agile" went along with their business counterparts and fostered the breakdown of software development paradigms to fit their own narrowly focused agendas allowing the trend to become the prevalent ideology it is today.

"**DevOps**" in retrospect is similar to the late addition of the "missile only" ideology in the Vietnam War, which caused many tragic deaths unnecessarily as well as the loss of expensive equipment. Here, however, "**DevOps**" hopes to "streamline" the production implementation process while quietly undoing compartmentalized operational procedures as software developers along the way are encouraged to take on these roles as well. US Fighter pilots have been literally "streamlined" out of their effectiveness.

The result is a growing similarity to what has happened in US combat aviation where software developers are involved in so many different aspects of creating software they are no longer doing what they really should be doing on a regular basis.

"**DevOps**" won't really streamline anything. Instead it will foster additional chaos within software development organizations as they all try to cope with this new "ideology" that promises them better implementations of production products. But will it?

With manufacturing, leading process flow analysts refined "existing" processes to develop better results in manufacturing processes, which found its way into the US manufacturing sectors. The US subsequently threw that all away with outsourcing.

"**DevOps**" will simply encourage companies to eliminate operations departments that house quite a number of areas all necessary to proper production implementation. And we have all seen this with the results of outsourcing and the hyping of and acceptance of the "Agile" paradigm. There is no reason to suspect that a similar process will not occur with "**DevOps**".

"Quality Control" is the leading area that documents have mentioned as something that "can" be merged with software organizations. However, there is also "Production Implementation" and "Production Control", the latter which is often tasked for verifying the accuracy of the output of various systems. There are also the hardware and network teams, which often come under the purview of operations. All this is now expected to come under the guise of "**DevOps**". And exactly how are these various areas to be held accountable to other areas in the IT organization when now they could just as easily lose their ticketing systems that control reporting of issues to software organizations as well as other such systems that provide necessary statistical information for accountability. They could also lose "Quality Control's" ability to do quality initial and regression testing for production implementations, while "Production Control" can also rely on software developers to verify the accuracy of their own results. Why bother with any of this when software developers can take on new responsibilities for testing and new updates can be rushed into production very quickly?

In both ideologies, "better communication" between users and software developers and software developers and operations is touted as a primary motivation for all of this. But is there really a problem here? Do operations departments have to be somehow melded with software for "better communications". This doesn't seem plausible with all the tools that vendors tout as the answer for everything so why meld two types of departments into one to streamline existing processes? Can't these processes be streamlined

without disrupting existing structures in the IT organization?

Part of the answer to this that "**DevOps**" promoters offer is the concept of "continuous integration", the process by which software developers can more easily get their out6put into production environments. It is a nice idea but in reality does it really work? Microsoft offered a software aspect of this when they upgraded IIS to allow for real-time module updates with .NET web applications, which included the dynamically interpreted web-site component to the .NET infrastructure. This allowed operations personnel to merely copy over a new C# or VB.NET source-code module to an IIS web-site without having to take the server down.

This new enhancement, which appeared sometime around 2005/2006 had its drawbacks however. For one, source code is not as secure as MSIL pseudo-code (though that has proven to be not so secure either but it can be obfuscated), which .net compiled applications produce and second, modules that had not been JIT compiled into the cache needed to be compiled or recompiled again if removed. This did not present overly serious issues but it did make web-sites somewhat sloppier in smaller organizations where accountability was often overlooked; except when necessary.

The long-term outgrowth of this improvement is that now entire web-based applications can be easily modified on a regular basis with hyper-defined schedules that allowed for many releases of such modifications on daily basis. What has this given rise to; more carelessness on the part of software developers and\or their technical managers since if a defect was found it could be quickly modified and implemented. This in turn lowered the scale of quality in such organizations as software personnel are now promoting the idea that it doesn't matter if defects enter into production any longer since they can be so quickly corrected.

That is all well and good as long as a critical defect doesn't get promoted into production and left there like one did in the 1980s at a highly reputable hospital, which seriously affected life and death analysis. At this particular hospital a new diagnostic-results package was implemented through a third-party vendor that did such development. The vendor had not done their quality control properly within the area of data entry for primary and secondary blood test results that were to be entered into the system. These two blood tests were standard procedure in all hospitals since the second test was used to confirm the first test's results to ensure a proper diagnosis or dispute it. The life-and-death nature of this issue was that when the second blood test's results were entered and successfully saved, the system did not record these tests as the secondary test results but merely copied over the results of the first blood test to that of the second set of results and displayed them to the doctors when they did their patient analysis. This not only presented the viewing of such information inaccurately but eliminated the data of the vital results of the second set of tests. Thus, the two blood tests ALWAYS confirmed the original diagnosis. This very serious defect was never caught by anyone; not the vendor, not the implementers of the software on site, nor the nurses or the doctors since they were probably looking at these tests from different aspects when doing their research. And so that defect was implemented into production and stayed there for 6 months being processed on a daily basis until the hospital's employed senior analyst in charge of the data caught the issue quite by accident and was able to contact the software vendor to have it quickly corrected.

Luckily, very luckily, no one died as a result as this hospital would have been brought to its knees had such an issue ever been exposed publicly. Again, luckily, it did not.

This is what happens when even proper IT structures were in place. Think of what can happen if they aren't as "**DevOps**" promoters propose. Does anyone really believe that the **DevOps** infrastructure would have saved the day if businesses see a new opportunity to cut IT staff further and software developers are under increasing pressures to ensure that their development can be free of defects way beyond the 60% level already noted? It is not very likely.

The final fly in this ointment is the actual "brain drain" as a result of both "Agile" and "**DevOps**" that has become the norm in the professional software field with the near glee that younger professionals appear to have concerning the pushing out of more experienced senior professionals.

If we go back to the fighter pilot example we find that currently, military pilots often have to move out of their positions after around 4 years of active flying or they lose their ability to maintain their military careers. While they are on active duty, they receive far less than adequate training and less flying hours than their foreign counterparts due to the constant re-allocation of military funds for political adventurism in foreign conflicts that has literally sapped the entire US economy and the consistent purchasing of highly expensive weapon systems that are now being proven to be nothing more than syphons into the military weapon contractors' pockets since none of these new systems work as required. There are so many documents on this situation in the US on the new F-35 for example that a book could written from them.

In the software profession, vendors and software managers along with their business counterparts are increasingly looking for younger and younger professionals who often have no desire to work with senior professionals. The result is much younger IT organizations with less and less mature experience to catch many of the issues that are in the process of being developed allowing them to subtly undermine these new cultures in the work-place by becoming just another aspect of them.

To validate the truth behind age discrimination in the current software development industry simply go to http://www.infoworld.com and search on the words, "age discrimination" and a bevy of recent documents will come up describing the seriousness of the issue. The result of all this is that when a field allows for a "brain drain" to occur as it has with the software profession, knowledge bases contract substantially, younger professionals lose the ability to absorb mature disciplines and experiences of the older professionals, and quality seriously deteriorates. This has happened to US manufacturing, US military weapons development, US fighter pilots, and many other professions in the United States now including the software profession, where very little is produced of quality any longer.

When you start stripping away basic assets and resources for the purposes of cost containment (merely a nice phrase for making business executives wealthy) and "streamlining" operations, it affects everything in business organizations like a cancer. What happens then is that you get things like "Agile" and "**DevOps**" both of which are attempting to support existing sociological deterioration instead of fighting it.

Both these paradigms are continuing to promote the deterioration in quality software output and the younger professionals are not entirely to blame since they do not have access to the experience that can show them anything different. And when the few older professionals left have tried, they are shunned and pushed out of their positions due to the youth oriented arrogance.

Though there are fewer and fewer senior professionals left in the software profession, it may behoove the younger crowd to begin supporting their retention in order to be able to qualify what the younger crowd is proposing with their development paradigms. Who knows, maybe "Agile" and "**DevOps**" could be refined to the point where they actually make sense for their implementation while using the best of both software engineering and the newer concepts being promoted. However, what is happening now, it appears the possibility of this scenario occurring is not very likely...

References:

What Is **DevOps**?
http://theagileadmin.com/what-is-**devops**/

Is **devops** killing the developer?
http://www.javaworld.com/article/2152543/java-app-dev/is-**devops**-killing-the-developer.html

How '**DevOps**' is Killing the Developer
https://www.jeffknupp.com/blog/2014/04/15/how-**devops**-is-killing-the-developer/

Reforming America's Overhyped Airpower
http://www.pogo.org/our-work/straus-military-reform-project/military-reform/2013/reforming-americas-overhyped-airpower.html

License

Share

About the Author

Steve Naidamast

Software Developer (Senior) Black Falcon Software, Inc.
United States

No Biography provided

You may also be interested in...

- Active Directory Webservices for DevOps
- DevOps in 2015 – Beyond Basic Metrics
- The Road to DevOps ROI
- An Introduction to DevOps
- DevOps needs DevSec
- SAPrefs - Netscape-like Preferences Dialog

Comments and Discussions

You must Sign In to use this message board.

Search Comments Go

First Prev Next

Applicable to so much...

Member 10179960 16-Feb-16 7:28

Thanks for the beautiful article. I don't have much time at work to read deeply (most is skimming to find "diamonds"), but this article tied me up and pushed my other work to the side.

If you look closely at the advance of many technologies over the past 100 years, this increasing lack of attention to fundamental details and experience, in favor of mechanization and automation of process, has truly increased the volume

of output but has decreased its intrinsic quality.

The quality difference is usually either unnoticeable to the uninitiated or it is called "negligible".

I can name several fields where the quality standards of 2016 are so different from those of 1916 that one could seriously ask whether there has been a complete abandonment of standards. I'll name three simply because I am familiar with them: bringing books to press, computation, and penology. I suspect that my observations apply across the board to ALL civilized activity.

In not one of these fields has there been an increase in quality of output except in technical metrics... factual, grammatical, and spelling errors abound in modern publishing; very few people can use pencil and paper to figure a simple division problem; and there is a greater tolerance for disrespect of societal mores and norms.

All this in spite of a vast increase in publication (if we add in electronic publication, certainly), computations, and percentages of jailed people. Faster and more, but shoddier.

One might suspect that this is simply a "North American" problem, but that is simplistic. It is parallel with the worldwide adoption of the idolatry of efficiency and power rather than respect for intrinsic worth.

I'll leave off here because this would probably become a rant and I am writing on my employer's time.

Thank you for a mentally stimulating article.

Davarino

Sign In · View Thread · Permalink 5.00/5 (1 vote)

Re: Applicable to so much...

Steve Naidamast 16-Feb-16 8:00

@Davarino

Thank you for your positive comments regarding my latest article. They are very much appreciated.

I completely agree with you that modern culture has taken a turn for the worse regarding the quality of products being developed. And you are correct to note that this deterioration is global than just relegated to the United States. However, from my own experiences I have found the US to be the worst case.

My wife and I traveled to Austria for many years regularly starting in 1993. Back then Austria still had a wonderful, traditional culture where whatever was being produced in the public domain whether it was art or the food had an exceptional quality. The public courtesy was beyond reproach, though the Viennese in particular are a very hard people to befriend. Nevertheless I accepted their personalities given that I was there to enjoy it all no matter the negatives.

Today, Austria, especially Vienna where we spent most of our time, has been terribly Americanized as a result of the neoconservative business model being taken up by many in Europe.

Though the quality of the products still appeared to remain at a good level we did begin to notice the deterioration that comes with the reduction of traditional ties to the public domain.

Austria, for us, was no longer the special place we found in 1993.

What is interesting to note for our profession was that even through the 1990s, professional US developers still saw their field as a craft first and then a science. Now it is simply a clinical mess.

Maybe we should all watch Rod Serling's last television film, "Requiem for A Country Doctor". It has very prescient meaning for today's sound-bite culture...

Steve Naidamast
Sr. Software Engineer
Black Falcon Software, Inc.
blackfalconsoftware@outlook.com

Sign In · View Thread · Permalink

Re: Applicable to so much...
Member 10179960 16-Feb-16 9:56

Oh, I certainly did not intend to imply that the U.S. situation was not worse than most. Whether it is THE worst I leave to more travelled people than I to decide.

Material culture is rampant.

Let's face it: material culture can pander to every weakness known.

It's hard to resist the siren song of pleasure and opulence, especially when the services of science, mental activity, and measurement are employed to accompany the music of that song.

"You, too, can have the shadow of the good life at WalMart! You must simply sacrifice the livelihood and dignity of your neighbor."

Sign In · View Thread · Permalink

Re: Applicable to so much...
Steve Naidamast 16-Feb-16 10:02

I didn't see anything in your comments where you implied anything in that regard.

I was just saying that with my experiences, the US in this regard appears to be the worst case.

To be fair, I should have mentioned that people have had similar experiences with the UK, which has certainly drunk the neoconservative "Kool Aid". And France as well is seeing it's traditions go down the drain as it is following the same path that the US has taken in becoming a "police state"...

Steve Naidamast
Sr. Software Engineer
Black Falcon Software, Inc.
blackfalconsoftware@outlook.com

Sign In · View Thread · Permalink

My vote of 1
LarryC11 16-Feb-16 7:20

This timely article correctly calls attention to a very real and serious problem faced by software developers. However, it needs a bit of editorializing to get the points across to all readers. The comparison between DevOps and modern fighter role is a good idea but the main point was lost in the explanations. It needs to make the convincing argument to managements everywhere that overworking the software developers is making them less productive. Also, there were distracting misspellings such as Calvary vs. Cavalry that need to be cleaned up.

Sign In · View Thread · Permalink

Re: My vote of 1
Steve Naidamast 16-Feb-16 7:34

Thank you for your comments.

The article was written to target software developers and the sociological processes that are seriously affecting the profession.

Unfortunately, both business and technical management have deaf ears when proposing anything near reality to them. I have tried many different approaches towards management regarding quality development processes but they all failed, including one in which I demonstrated the success of a project using pure software engineering standard practices.

The problem is really with the US business culture, which sees mediocrity as a successful level to achieve. Software professionals are simply not part of this equation and is the reason that so many software projects continue to fail at different levels.

Steve Naidamast
Sr. Software Engineer
Black Falcon Software, Inc.
blackfalconsoftware@outlook.com

It is nice topic

Dr. Song Li 15-Feb-16 15:28

Hi Steve,

It is a nice and important topic.

I did not give my vote because I have not been able to read it carefully and I have not got your points. But I have to agree that you gave us a good and important topic. One suggestion is that if you can use even more plain English it will be even greater, considering that the native language of a large portion of your readers (Code project is accessible through the world) is not English.

In any case, think you for writing it up.

Thanks,
Song

Re: It is nice topic

Steve Naidamast 16-Feb-16 2:01

@Drr. Song Li

Thank you for being able to appreciate the piece I wrote.

I understand that many people who do not have English as their primary language may have some difficulty with my pieces. Unfortunately, my style is one that I have developed over many years of writing. And given the sociological complexity of this topic I have no idea as to how I would have written it clearer than I already have.

If you have any questions or misunderstandings that you would like to have me answer or clear up for you I would be happy to respond to them if you email me at my email-address, which I provide with all my replies to comments on this forum.

Steve Naidamast
Sr. Software Engineer
Black Falcon Software, Inc.
blackfalconsoftware@outlook.com

Re: It is nice topic

Dr. Song Li 16-Feb-16 2:10

Understand it Steve. No worry. The complexity of this subject is so much beyond "Object Orientation".

By the way, you may be interested in this link: Agile Is Dead (Long Live Agility) - PragDave

Thanks,
Song

Sign In · View Thread · Permalink

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin