Not quite what you are looking for? You may want to try:

- Active Directory Webservices for DevOps
- The Road to DevOps ROI

highlights off

12,347,297 members (74,681 online) Sign in

CODE PROJECT® For those who code

articles Q&A forums lounge DevOps

“Agile Development” – The Software Industry’s New Meme

Steve Naidamast, 8 Feb 2016 CPOL

5.00 (20 votes) Rate this:

“Agile Development” – The Software Industry’s New Meme

“Agile” as A Meme

Since the introduction of Agile Development in the early 2000s, it has increasingly spread throughout the software industry in the same manner a popular cult is formed; based on supposition, mythology, and rumor for which little real statistical proof has ever been provided.

A more accurate description of the growing popularity of this development paradigm is that of a sociological “meme”. Wikipedia defines the word “meme” using basically the same definition as found in the online version of the Merriam-Webster Dictionary, a standard in English studies, as the following…

*“A **meme** is “an idea, behavior, or style that spreads from person to person within a culture”. A meme acts as a unit for carrying cultural ideas, symbols, or practices that can be transmitted from one mind to another through writing, speech, gestures, rituals, or other imitable phenomena with a mimicked theme. Supporters of the concept regard memes as cultural analogues to genes in that they self-replicate, mutate, and respond to selective pressures.”*

The Wikipedia description goes on to elaborate on the transfer of memes in the same way that is found in biological evolution.

“Proponents theorize that memes are a viral phenomenon that may evolve by natural selection in a manner analogous to that of biological evolution. Memes do this through the processes of variation, mutation, competition, and inheritance, each of which influences a meme’s reproductive success. Memes spread through the behavior that they generate in their hosts. Memes that propagate less prolifically may become extinct, while others may survive, spread, and (for better or for worse) mutate. Memes that replicate most effectively enjoy more success, and some may replicate effectively even when they prove to be detrimental to the welfare of their hosts.”

In other words, memes are transferred in a culture the same way that natural selection works in biology; except for the fact that with memes, which are transfers of information, they become successful not as a result of intelligent research and understanding by the Human transfer points but in somewhat the same manner that fads are often found to grow in popularity, through simple popularity of an accepted strain of thought.

Birth Pains

When “Extreme Programming (XP)” was developed prior to the Agile movement, it was defined in the fourth year of a five-year projected development timeline for the Chrysler C3 Payroll system, a system that was to be developed to replace the existing payroll processes. The people that developed the XP development paradigm *thought* they had found a “silver bullet” to software development with their dilution of planning and analytical design coupled with the new idea of “paired programming” mixed with a “Code First” strategy. One would think that after working with such development concepts over the four years of their project, they would have been actually on to something. Not waiting for the final results in the form of a completed and delivered project into production, the “Extreme Programming” creators went out on tour around the US to popularize what they believed they had developed as the answer for speeding up software development in a time when technical management was coming under increasing pressures to get their deliverables in to production faster. These promoters elicited a lot of interest and the basis for a new software development paradigm was born.

Unfortunately, in the 5th and final year of the project, the entire thing collapsed under the weight of these newly derived concepts proving that they were erroneous from the beginning.

Nonetheless, a new software industry “meme” had been born as a result of the popularization efforts by the creators of XP. This new meme would have found its way into extinction had it not been for the efforts of a new set of developers (with probably the same people who created XP) who came up with the “Agile Manifesto” promoting a more refined methodology where the best tenets (as they believed them to be) of XP development could be incorporated, while adding alternative and\or refined concepts to the “Agile” paradigm, which were seen as corrections to the original XP model. So with “Agile”, paired programming was an option and not a requirement but nonetheless still promoted. The “Code First” strategy was still followed while planning and design were diluted with the idea of collaborative development with the user community; another concept that has proven to have mixed results since many users really don’t care how their systems get built.

In fairness to the developers of both the XP and “Agile” techniques, there were two major, external forces that lent their influence to these developments.

First was the excited frenzy that surrounded the rise of the Internet as a commercial medium for transactional processing, which in turn gave rise to the “DotCom Bubble, where clueless venture capitalists and to some extent organized crime invested millions if not billions in an outrageous number of new start-up companies all organized around bringing new forms of consumerism to the Internet.

Such start-ups were under heavy pressures to develop new applications that would allow everyone to buy literally anything on the Internet from credit to physical goods. The “meme” born from XP made its way into these small firms who were all under the impression that they had found something completely new and unique with this new form of rapid application development paradigm. Combined with the inexperience of the new youth oriented culture of these new start-ups, the development techniques they employed against frivolous plans and goals that in many cases made little sense, all came together in the first software development business “bubble”, which eventually blew up in their faces pushing the newly minted “DotCom Era” into a sudden free-fall it never recovered from.

However, the idea of using the Internet for commercial processes was not as farfetched as the failure of so many companies made it appear to be. The Internet was certainly a viable medium to work with so everyone went back to the drawing boards to begin again.

Saved From Extinction

At the same time, the outsourcing of US technical jobs was becoming as much a frenzy in the early to mid-2000s as corporations saw the Internet as a communication medium that could be used for connecting technical departments to foreign counterparts in countries such as India and China that could provide similar technical resources for far lower costs than what could be found in the United States; or so it was thought as no one gave any thought to the costs in administration, management, quality control, and project control to name a few ignored areas in this rush to cut the bottom line.

This rising inequity within the US corporate sphere gave new impetus to US technical personal to adopt techniques that rushed out deliverables at greater speeds trying to prove that US developers were as every bit as good as their lowly-paid foreign counterparts. The result was that instead of maintaining the original quality of the US software community it instead slowly adopted the “Agile” paradigm allowing corporations in turn (though they were doing it anyway) to reduce or eliminate vital components to the development of quality software such as business analysts, system analysts, software designers, architects, quality control centers and the like pushing increased responsibilities and pressures onto software developers and engineers all with the ridiculous idea that software development teams could do it all.

Under such circumstances the “meme” that was born with XP found a new nesting place in the developing Agile Community of proponents. The result was the growing contention that software developers could do everything while also adopting in-depth knowledge of the businesses they were working in. Article after article appeared in the field’s technical journals proposing as much and technical managers looking for ways to reduce costs and speed up the delivery of their products “drank the Cool-Aid” giving this nascent meme a new lease on life instead of the extinction it should have experienced.

Today, hardly a technical site ignores this as one article after the other is published proclaiming the benefits of Agile, Team-Agile for enterprises and the new paradigm of **DevOps**, which appears to increase responsibility by including operations personnel as part of the software development team as well as their responsibilities. The industry's propaganda engines have been on full throttle for quite a while and the deterioration within the software development industry has demonstrated this. Even Microsoft has fallen prey to this infectious disease of high-speed development without proper structural planning and analysts have recently made note of the deterioration in their product quality after switching to the "Agile" paradigm.

Nonetheless, in all the years that this meme has infested the software profession, very little hard-core analysis has been performed in the same fashion that long-term analysis of earlier software development efforts and the many impediments to quality product delivery had been analyzed. There are some articles pointing to the successes and failures of the Agile methodology, but for the most part they are few and far in-between with little to no actual in-depth discussion has to how the projects were actually implemented. People who have written such articles that accurately describe Agile as a failure in their own experiences are merely told by proponents that it wasn't implemented correctly or some facets of the paradigm were not in included.

What Did We All Do Before "Agile"?

However, prior to even XP, the "bible" of software development ("Rapid Development"), was published in 1996 by Stephen McConnell, which outlined every major aspect on how to develop good, durable software products without all of the hype of the paradigms today. This book is still so popular it is still available on Amazon.com in its original 1st edition publication. And prior to its publications, there were many examples documented of how good software projects were structured. In one such case, an entire book that included all of the framework source-code for a highly successful project for the state of South Dakota demonstrated how Visual Basic 6.0 was used for both the project's desktop and web-based applications. The concurrent usage of this system was over 1500 users! "Agile" was not included just standard, well-defined N-tiered development processes.

Instead of looking deeply into the past for what actually worked, creators of XP and Agile only took a short glance and found its only target being that of the "Waterfall Approach" to software development demonizing it as the worst possible way to create new software while promoting the new concepts as the saviors of the software industry. And if one were to apply the "Waterfall Approach" to every type of software development project, that accusation would be quite correct. However, the way Agile is being promoted is in fact the same way that the "Waterfall Approach" is being demonized since Agile promoters see it as the panacea for all types of projects, though they have never offered any substantial proof in order to make such a justification.

The reality is that the "Waterfall Approach" is merely one type of development paradigm, which has been quite successful for the large, complex projects that it was originally designed to be used against. However, there are approximately 10 different types of software development paradigms that have been in use for years that the current crop of software professionals appear to have either ignored or never bothered to research. If they had, their theories on "Agile" would fall apart. To demonstrate this, here are the standard development models that have all been used to date in terms of software engineering analysis…

- Pure Waterfall
- Code & Fix (what many organizations still use to their detriment for all projects)
- Spiral
- Modified Waterfall
- Evolutionary prototyping
- Staged Delivery
- Evolutionary Delivery
- Design to Schedule
- Design to Tools
- Off The Shelf Software Implementation

Each of the development models above have their strengths and weaknesses and each has to be researched for its appropriate use for a new project under development.

Surprisingly however, the "Code & Fix" model is what Agile today mostly mirrors, which is fine for simple maintenance tasks but useless for complex maintenance and complex project development. In this former vein, Agile has proven somewhat successful but has found its limitations when applied to these other larger projects.

Nonetheless, like "Code & Fix", Agile is being used for many different types of projects as a result of the effectiveness of the belief that "Agile" can be applied just about anywhere in the software development spectrum (an underlying foundation to the meme contention). And being as embedded in the software industry's culture as it is, "Agile Development" has become the model for all such development making it not only a "cultural meme" in the profession but a dangerous one similar to what happens with cults; unquestioned belief.

Questioning the Dogma

There are those who have questioned the promotion of the "Agile" model and have done so quite well, though such people are ignored in today's conformist society since for most it is now too difficult to do any real critical thinking on matters within a culture that promotes sound-bite thinking.

Capers Jones, a highly respected software engineering analyst, has an excellent paper that demolishes the concept of paired-programming, which can be found here.

In his paper, Jones demonstrates the faulty economics that paired-programming forces on technical organizations as well as demonstrates statistical analysis showing the failures in such a development technique.

In another well written paper by Lajos Moczar and published in CIO magazine, he demonstrates that "Agile" simply lacks any common sense in its approach to software development. The paper is short but concise in its presentation and can be found here.

Finally, in a third paper, written by Alex Papadimoulis, "Agile development" is compared to building a pyramid sideways. The comments are as interesting to read as the article as you will find both supporters of Alex's contention as well as detractors. Admittedly, the concept is a little far-fetched, but the author makes some very good points.

And not surprisingly, such papers and subsequent analysis today is rather difficult to come by as it took quite a while to dig these up in research. Instead, there should be a lot more out there with similar research and points of view but there isn't as "Agile" thinking has literally taken over the psychology in the business application development corner of the software profession. And anyone who tends to disagree with this is basically told they don't know what they are talking about.

This process has also shown the same deterioration in critical thinking in software development that has been shown in many aspects of US national sociology, especially education, as demonstrated in well documented, sociological studies. With mobile-computing taking center stage in most popularized software development today, one needs no longer the critical skills in complex system development that were once the hallmark of professional US software developers and engineers. Instead, mobile computing requires a rather limited of skills in order to build non-complex applications considering that only so much can fit on a smart-phone with only a proportional increase in complexity for tablet-based software. And yet, what is actually being developed for these devices?

In fact, if one were to take a look at the Windows App Store and its breakdown of stored applications (which has probably the same amount as the Android Store has), practically 90% of such applications are devoted to games and entertainment leaving one to wonder exactly how necessary such "junk consumerism" is to our national, psychological health.

"Agile", which has done a rather good job in reducing software development processes to nothing more than sound-bites in terms of any real discussion that surrounds it has in fact seen a number of what appear to be well designed documents discussing various aspects of "Agile". However, the graphics used in these documents all of which have shown nothing more than photographed scratches on paper for diagrams of the intended topic appears to demonstrate the rather casual approach to the serious subject of software development that this paradigm supports. One would think that the authors, if they were serious about their subject matter, would have created better graphics that supported their contentions. It is these images that tended to stand out as if testifying to the idea that "Agile" is as casual as the images appear to be.

The "Agile" meme, which can now be found in just about every aspect of modern, business, software development may in fact be quite supportive of the fluff that passes today for real software projects as is shown by its success for simple maintenance tasks. However, until we see real, critical analysis on a far, wider scale regarding the "Agile" development methodology we can assume that its realities are being ignored (or hidden) while the entire business development aspect of the profession is seduced into believing that the "magic silver bullet" for software development has finally been found.

It is a telling tale of why wisdom is wasted on the old and youth is wasted on the young...

License

Share

@ EMAIL

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

The Path From COBOL to Mobile

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

Unconstructive rant (the article, not this comment... well maybe)
destynova 11-Feb-16 2:09

As someone who has worked on different projects, at different companies and times, each at a different point on some arbitrary scale of "Agile", I saw benefits and drawbacks with every method of designing products and building software that we tried.

This article was very disappointing to me because it didn't really discuss those benefits and drawbacks in a meaningful, constructive way. Instead it could be summed up as "Agile is hipster nonsense, and these small number of smart people agree with me..." followed by an irrelevant and almost incoherent rant about "junk-consumerism", mobile apps and "our national, psychological health" which presupposes that all readers will be American for some reason.

Instead of wasting the reader's time talking about how unwise young people are drinking the Kool-Aid of the Agile cult, you could have drilled down into exactly which parts of "Agile" (and there isn't just one "Agile", there are quite a few approaches and techniques) work and which do not. You pointed to Capers Jones' paper which rated the effectiveness of a whole bunch of things, which I found last year and was surprised to see that pair programming was rated poorly, with static analysis tools appearing to contribute more directly to success. But why? Was it due to the way pair programming was applied, or the types of problem being worked on, or is pair programming just bad in general, and if so why is it bad?

My experiences of pair programming (almost 10 years ago, on a small team doing a subset of XP stuff) were overwhelmingly positive, even if only because people wouldn't get "stuck" through frustrating problems or lack of motivation. The downside was probably getting into lengthy sidetrack discussions about unimportant things like code style, whether an exception should be thrown here or null returned, etc.

Clearly you have a long experience and keen insight here -- I just wish you'd used it to present a balanced evaluation of the useful (rapid iteration and stakeholder feedback, small, frequent deliverables, collective responsibility for the entire codebase etc) and less useful parts (still need to work hard to get meaningful and concise specifications from stakeholders, and planning and estimation are difficult when you're not specifying everything up front) of Agile. Instead you've just put together an angry hatchet job denouncing the entire movement, which is such a waste.

modified 11-Feb-16 8:21am.

Sign In · View Thread · Permalink

Agile is a fashion

AndyKEnZ 11-Feb-16 0:45

O this'll put the cat among'st the pigeons 😊

I wish I could find an article I was reading about fashion and how it is created obsolescence.

I've an inkling it was an essay by Bertrand Russell (or was it Orwell?), anyway I think agile will eventually become old-fashioned, so be careful all you fashionistas.

Sign In · View Thread · Permalink

I think perhaps you cannot see the forest for the trees

Enigmaticatious 7-Feb-16 15:51

First I have to say, that your article seemed incredibly Amero-centric, and not something which applied in many other countries, which would contradict your reasoning behind "why" this became popular.

I also think that perhaps you have ignored the human factor here and the general history of traditional software development.

As a veteran of 20+ years, I have worked under the majority of paradigms, for large companies and small, for outsourced companies and in house and my experience is drastically different to what you discuss here and there are no indicators which would suggest my experience is any different to most other people.

How many projects wasted millions of dollars and hours to fail with nothing to show for it but the death of a thousand trees in a detailed design document and nothing else? How many projects never got passed user acceptance because the product put in front of the user was NOTHING like they wanted?

No, you have completely missed the point of Agile development. Agile development is ALL about iteration, and yet this word did not appear ONCE in your article. It is about putting something in front of a user long before a million dollars have been spent and getting their feedback and improving it with subsequent cycles. It has absolutely nothing to do with being able to not hire architects or business analysts... and I have NEVER seen an agile project NOT include these roles, why on earth you think it did is beyond me.

Managers picked up the "buzz word" because they knew their clients were burnt too many timed with wasted rigid software development where nothing of value is shown until well after much money was spent. They needed to find some way to convince their clients not to abandon them, so when a term came along that allowed them to say "its not like what you have experienced before" they jumped on it for this reason alone.

It freed them from the shackles of having to go through several tiers of design, review, approval, re-review, update, re-submission and re-approval and made that time PRODUCTIVE by pushing something out the customers could actually get their hands on and FEEL like progress was being made.

The REASON why so many people don't understand what agile truly means is because they only want to use it in name only, they see no benefit in the model, nor even care about it. To them it is just there to keep the powers that be off their back.

The interesting thing, which you also completely ignored is that despite this new paradigm, businesses are still STUCK in the old world, they still want to know exactly how long something will cost up front so they can budget and still fail to understand that estimation with accuracy is only achieved by understanding the problem space in greater and greater detail. Try telling them that software will be done when it is done and will take as long as it takes and they simply refuse to accept it.

The problem has always been PEOPLE, and their inability to understand software development. They all think that it shouldn't cost as much as it does without any point of reference as to how much it really should cost. It is the reason why ALL salesman will always undercut the truth of their prices because they know it isn't the most accurate price that wins, it is the lowest price.

All you are seeing is just another "buzz" word, misused for their own purposes and never truly understood... as always, it is the abusers of it that obfuscate the truth of it.

At the end of the day, a piece of software will take as long as it takes, and no matter how much or how little you design it, no method can make it faster, only bad methods can make it slower. You design where design is needed, you dont do things just for the sake of it, you document where documentation is needed and you most DEFINITELY need an architect for any major project regardless of what methodology you use... just because it is agile doesn't mean you act like a blind moron and ignore the logical and fundamentally simple concept of "understand your problem space"

Sign In · View Thread · Permalink 5.00/5 (3 votes)

Re: I think perhaps you cannot see the forest for the trees

Slacker007 8-Feb-16 2:23

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

Re: I think perhaps you cannot see the forest for the trees

rzvdaniel 10-Feb-16 22:35

I agree with you.

It looks like the author was exposed to "teams who were doing Flaccid Scrum and made the mistake that that's all there was to Agile (Uncle Bob)".
<a href="https://blog.8thlight.com/uncle-bob/2015/10/16/agile-and-waterfall.html">Agile is not now, nor was it ever, Waterfall.</a>

Martin Fowler about Flaccid Scrum:
"I always like to point out that it isn't methodologies that succeed or fail, it's teams that succeed or fail. Taking on a process can help a team raise its game, but in the end it's the team that matters and carries the responsibility to do what works for them. I'm sure that the many Flaccid Scrum projects being run will harm Scrum's reputation, and probably the broader agile reputation as well".
<a href="http://martinfowler.com/bliki/FlaccidScrum.html">FlaccidScrum</a>

-- modified 11-Feb-16 5:49am.

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

Image

Nelek 7-Feb-16 0:20

Have you written allowance of copyright owners to use that image? If not... please consider to delete it here in CP

M.D.V.

If something has a solution... Why do we have to worry about?. If it has no solution... For what reason do we have to worry about?
Help me to understand what I'm saying, and I'll explain it better to you
Rating helpful answers is nice, but saying thanks can be even nicer.

Sign In · View Thread · Permalink

Re: Image

Steve Naidamast 7-Feb-16 2:53

The image used for the article was found in the public domain. It was in fact taken off of Twitter by Google.

I did try to find the original owner of the image but could find no information as to who it belonged to.

In addition, if an image has copyright rules against it, it is either water-marked or noted that it cannot be used for commercial use.

In any event, the way the image is being used, it does not provide any form of profit.

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

Sign In · View Thread · Permalink

Re: Image

Nelek 7-Feb-16 22:05

> **Steve Naidamast wrote:**
> The image used for the article was found in the public domain.

As everything in the internet in public domain is legal...

> **Steve Naidamast wrote:**
> I did try to find the original owner of the image but could find no information as to who it belonged to.

Good to know, thanks for it.

> **Steve Naidamast wrote:**
> In addition, if an image has copyright rules against it, it is either water-marked or noted that it cannot be used for commercial use.

Not always, not necessarily and not imperative. This is part of a film and the film is younger than 50 years, ergo it has copyright perse.

> **Steve Naidamast wrote:**
> In any event, the way the image is being used, it does not provide any form of profit

If a moron find it, yes it will give profit, for him demanding you or CP

As side note:
1) I don't find that image is giving any value to your technical contribution (which I find good)

2) I am not commanding you, I am not CP staff only volunteer. Internet is still a free land, do what you want. I just like the site and want to avoid unnecessary problems with something that could be easily solved. If you want leave it in your personal blog, but please edit it here in CP

M.D.V.

If something has a solution... Why do we have to worry about?. If it has no solution... For what reason do we have to worry about?
Help me to understand what I'm saying, and I'll explain it better to you
Rating helpful answers is nice, but saying thanks can be even nicer.

Sign In · View Thread · Permalink

It's not all bad

Will J Miller 6-Feb-16 22:48

Steve, I enjoyed your article, and I plan to read the references. I'm a long time veteran of the industry, and I have shared many of the same reactions to the Agile movement. I even authored a blog for a time where I was critical of the process.

Then the wave hit home and I had to embrace the methodology, and while I'm no convert, it has its place. When I attended university a few decades ago, we accepted the principle that 80% of the effort was spent maintaining the code. As it turned out, it was true. I've probably spent more than 80% of my time devoted to maintaining old applications than writing new ones from scratch.

The old school thinking was to invest a great deal in the architecture, design principles, and code quality so the software would be easy to maintain. This was hard work, and projects were often late and over budget. And unfortunately many failed in spite of these best practices. Sometimes, it's just a bad idea; sometimes, the team is ineffective; Either reason, the development team gets the blame.

What agile has done is optimize the maintenance phase where most application development teams spend 80% of their time by lightening up the process while the old schoolers aimed to optimize the phase by building a rock solid foundation.

Unfortunately, too many teams for reasons having **nothing** to do with the SDP find themselves maintaining systems built on a house of cards. So maybe focusing on the SDP in this phase isn't such a terrible idea.

In the hands of a skilled manager, one who really understands the essence of software development and how to organize people around a common task, it can be quite appropriate and successful. Project teams will deliver predictably in short intervals, and the application improves over time.

Where the software community has had challenges is with finding the right balance between all the competing demands. And depending on what you're building and for what reason, the SDP should change to support it. We never get all the ideal inputs for success. The best approach is often an improvised one where the team picks the best methodologies to meet the challenges of the project's unique competing demands.

It's tempting to throw the baby out with the dirty bathwater. There are aspects to embrace and failings of the old methodologies to acknowledge even when they are applied to their proper domains.

Nice Article!

Sign In · View Thread · Permalink

Re: It's not all bad

Steve Naidamast 7-Feb-16 3:05

@Willl J Miller

Thank you for excellent, in-depth comments.

I believe I stated in the article that "Agile" has had success in the maintenance arena. However, recent documents have begun to show that it is completely unfeasible for complex and enterprise software development.

"Agile" has also been shown to increase developer burn-out over time. And since I worked in what was considered an "Agile" environment on a large-scale project I experienced that first-hand.

The reason why so many projects failed with earlier methods had very little to do with software developers and engineers but technical managers refusing to adhere to quality practices as well as proper estimation metrics. This mixed with business users that had no interest in accurate timelines and thus used initial, inaccurate time estimates for project completion simply initiated projects that were doomed to failure. Those that did succeed did so through sheer luck or as a result of proper planning and implementation.

The issue has never been with development practices but management. And this has been demonstrated time and time again in many analytical reports on the matter. For an excellent reference read McConnell's "Rapid Development" (1996).

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

Sign In · View Thread · Permalink

Good Article

Santhakumar M 4-Feb-16 20:21

Good Article

Sign In · View Thread · Permalink

The Fable of the Storks

sdmcnitt 4-Feb-16 6:13

I get the idea from your essay that you do not like Agile. 😊 I can relate. I have been around in IT long enough to have seen it all I think. A short story about flocking to the latest thing:

The Fable of the Storks

Once upon a time there was a community of storks. The pride and joy of the stork was the tail feathers.

One day hunters found the community and started chasing the storks to get their tail feathers.

A group of storks yelled to all "let's all run really fast." Next time the hunters came in Jeeps and plucked them.

Then a group of storks yelled to all "let's kick them really hard". The hunters then hunted in packs of two, one plucking from behind while the stork attacked the other.

Then a group of storks yelled to all "lets stick our heads in the sand". The hunters just took their tail feathers all the more easily.

The moral of the story: sometimes no matter what you do or who you follow - you're plucked.

I will check back in a few days and see the religious war raging in the comments. I hope the trolls stay nice.

I hope I die in my sleep like my grandpa Bart, not screaming and kicking like the passengers of his cab.

Sign In · View Thread · Permalink

Re: The Fable of the Storks

Steve Naidamast 4-Feb-16 6:59

You are correct in your observation. However, like you I have been in the profession a very long time and see "Agile" as an unnecessary paradigm when similar and far more successful engineering paradigms have already been used in producing high quality software.

I think your "Fable of The Storks" is quite appropriate for much of our part of the profession. However, I see the growth of "Agile" and now its lackey, "DevOps", more a result of society that is deteriorating in terms of critical thinking skills and which now seems to absorb information in terms of sound-bites making most younger professionals in a severe sense rather lazy.

Instead of doing what actually needs to be done to produce quality software they instead come up with a lot of nonsense to get around doing what many developers don't like doing such as planning, design, and documenting.

And to do this they use all sort of excuses such as business is changing in the 21st century, businesses demand faster turnaround in software development, we have to communicate better with our business partners, all of which our younger counterparts believe to be all new issues when they are either irrelevant or long term issues that will never be solved.

Business has not in any way changed in how it runs its various institutions. All businesses operate under the same criteria even if they become more democratically run such as with employee owned cooperatives. Such organizations have run basically in the same fashion since they were first designed in the modern incarnation in the 17th century. The fact that the millennium changed is meaningless.

Faster turnaround? Obviously these young technicians have never read any studies on Human multi-tasking, which has been scientifically proven to be nothing more than an illusion. However, numerous articles have been written in our tech journals on how we all do it and how to handle it giving rise to the corroboration of your Storks fable. All these youngsters believe this crap and other such ideas so they come up with how they can speed up deliverables.

It simply won't happen since each project, in which each is unique, in order to deliver a quality product requires its own determined, minimal amount of time to implement.

Oh no they say we can do "continuous integration". We can do it faster.

Well people can only work so fast and it has been reported that "Agile" environments tend to experience higher levels of burnout since personnel cannot keep up with the pace. And "continuous integration" is just another mirage they have set up for themselves.

Exactly, what are these people delivering in such a rapid delivery environments? I doubt it is complex code\features considering that such development takes time. And not surprisingly, few reports if any have stated exactly what it is they are delivering in 35 releases a day (as one company proudly touted a few months back)? As a result, it appears then that what is being delivered is minor corrections to the applications as well as defect corrections that have been allowed to enter production as a result of poor reviewing processes.

But that's ok with the new style of development! And I have seen this stated as such. "We don't have to develop perfect software!" No you don't but having so many releases in a single day indicates something is terribly wrong with your quality.

Everyone is in such a bloody rush today to get everything done faster and faster. Well, good software takes time and good, complex software takes even longer.

Eventually, the entire edifice will collapse under it's own weight just like the original "Extreme Programming" project did. And then these kids will suddenly find a brand new paradigm and they will call it "Waterfall"...

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

modified 4-Feb-16 13:21pm.

Sign In · View Thread · Permalink

The Agilist

DumpsterJuice 1-Feb-16 10:20

I have experienced several Agile Projects, all of which ultimately feel apart and became code death marches.

Its not that I think Agile is completely without merit, its just that it puts so much nonsense in front of getting things done, that it becomes its own reason for being.

Do I have a solution? no no really, I think all of them are flawed in some way, but you cant argue that Agile is less flawed, and get me excited about it. I believe in work breakdowns, into smaller chunks of very small teams.

It will be interesting to see how Agile untangles itself from the enterprise, as I know one day it must.

The thing that worries me - is what its replacement might look like.

Where there's smoke, there's a Blue Screen of death.

Sign In · View Thread · Permalink

Re: The Agilist

Steve Naidamast 1-Feb-16 11:33

@DumpsterJuicee

From reading through the technical publications I review what will come after "Agile" will probably be much worse if the current trends continue.

In particular, the trend towards development speed is to encourage ever faster turn-around times, which in lieu of the profession we are in is completely illogical since quality development output requires a minimal amount of specified time for each type of project. There is simply no getting around this axiom as it is more or less similar to a natural law in physics.

People can only work so fast and get things done properly. However, here the trend appears to be to accept defects in production deliverables. The question is why?

Are business users becoming so erratic with so little time that they can no longer understand that all projects whether they are software oriented or not require reasonable periods of time to develop and implement? I really do not believe that this is being pushed by all business users but instead by many technical professionals who want to impress them and actually believe that software development can be produced in similar style to "speed dating", which in of itself never made any sense at all.

It seems that in this "modern" world people can no longer simply take a breath to reflect on what they are doing and how to do it well; its just shovel the crap out as fast as possible without thinking about it.

"Agile" never had to replace anything since all of the mature practices had already been shown to work on a variety of different projects. However, no one looks at this material since it is "old" or what Microsoft likes to call "legacy".

And look at Microsoft and how they have propagated this sociological disruption. If .NET 4.6 is now out, they look at 4.5 applications as "legacy" applications. ASP.NET WebForms is a "legacy" development model next to ASP.NET MVC. And yet WebForms still works as well as anything else.

If you review the many technical journals for our field you will find that it appears that no business representative has ever demanded that software development speed up to such unheard of rates. No one has shown that any company in particular has demanded that such rapid development be the norm. It is all within the technical profession itself.

Yes, there are ways to make development more efficient when comparing an organization's development practices to mature, best practices that may not be followed there. However, our profession just seems to scream "faster" and no one is asking why.

No company has ever gone out of business because it took the time to implement quality software products. However, companies have gone out of business or reached crises points for not implementing software properly. A classic example for the latter case was the Oxford Health Care institution, one of the largest in the US at the time, that rushed in a brand new administrative system. It blew up in their faces and nearly destroyed the entire company. It never recovered from this foolishness...

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

Sign In · View Thread · Permalink

Excellent Write-Up

Member 8196534 30-Jan-16 8:02

I have (unfortunately!) been around this industry too long and have seen the rise, and subsequent fall, of many attempts to change development and project methodologies.

It is a sad indictment on our industry that this is the case. Too many projects still fail and so a new methodology is proposed that will solve all the problems of the previous ones... until that one is shown to have its flaws too.

Agile has its place - but it's no panacea, no one-size-fits-all solution and completely inappropriate for many projects. Unfortunately its vehement supporters tend not to appreciate the complexity and different nature of various software developments and simply bang the drum louder.

The thing that really gets me about Agile? Before Agile, if you wrote down all the things that developers don't like doing... and hey presto, you don't need to do those if you use Agile! Documentation? Nah, forget about any need for the software to still be in use five years down the line since you'll have moved on to other things. Proven, working code? Nah, you'll just fix bugs for the next build. Specifications? Nah, if the client doesn't like it, you'll just change it once they've explained what they want (and hence try doing Agile on fixed-price development work assuming you also need a coherent system at the end of it).

Sign In · View Thread · Permalink

Re: Excellent Write-Up

Steve Naidamast 30-Jan-16 10:16

Thank you for your positive comments regarding the outlook of my essay.

It is amazing how our profession has just turned a blind eye to techniques and tools that have proven their capabilities for new paradigms and tools that seem to either have no real benefit or if they do, they never have a chance to mature because the next "big thing" is right behind it.

With software development it appears that "Agile" has become popular by it's sheer inclination to eliminate or reduce functionality that developers just don't like doing; like you mentioned, proper documentation.

Hopefully, aircraft manufacturers will never adopt this silliness into their own software development endeavors along with other industries that deal with life and death situations or the Human species will be doomed...

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

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

Re: Excellent Write-Up

Enigmaticatious 7-Feb-16 15:56

I find it interesting that in your response you are victim-blaming here.

Everything is the fault of the methodology, not the fault of the people practising it, or the people who control the fate of the project in making unintelligent decisions because of it.

The problem isn't the method, the problem as always are the managers, the business managers, the accountants and the cowboys. A methodology is only as good as the people who use it, and a project doesn't fail because of the methodology, it fails because people have poor expectations of how much things cost, they have their own personal agendas which are directly in conflict with the project, developers (cowboys) are lazy, managers want to look good, business owners don't want to pay money.

I think the appropriate saying here is "Hate the PLAYER not the GAME"

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

Re: Excellent Write-Up

Member 10065956 9-Feb-16 7:16

If "a project doesn't fail because of the methodology", then why was there ever a need to create a new methodology?

Sign In · View Thread · Permalink

Re: Excellent Write-Up

Steve Naidamast 9-Feb-16 7:41

You are exactly correct in your observation.

However, there have been major, sociological forces that led to the development of "Agile" and not because anything that came before was inadequate. Instead, technical managers and then many software developers all were co-opted by business pressures to do more faster with less resources.

At the management level there was a lot of incompetence to begin with but at the developer level it appears to be a result of sheer panic as a reaction to being able to maintain one's job. When the younger generations began entering the field in increasing numbers, panic appears to have morphed into a cultural meme as the chaotic environments were seen by the new developers as the norm. They in turn then went on to refine what already existed.

Prior to all of this, there were many successful ways using software engineering practices to produce quality software in reasonable time periods but it took work and understanding; something many technical managers simply did not want to be bothered with. And I have seen this consistently throughout my career.

The result is we now have "formalized chaos"...

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

Re: Excellent Write-Up
Member 10065956 9-Feb-16 8:21

I couldn't agree more. I, too, have been developing (for profit) for a long time - 30+ years and still gainfully employed doing so. Unfortunately, ours is an industry that doesn't truly value experience. I'm sure I'll draw flak for this comment, but developers with 5 - 10 years experience are often deemed "senior", yet this simply isn't enough experience to put this into perspective. From my vantage point, you've done an excellent job of summarizing how we've gotten to this point — and, indeed, it's frightening.

Re: Excellent Write-Up
Steve Naidamast 9-Feb-16 8:46

Thank you for your kind comments.

As to the so called "seniors" you mention, I have always viewed anyone under 15 years in the field as one who hasn't started shaving yet. A senior to me is a person in their 50s.

If you liked this article, you will surely like the next one as it demonstrates with a real world example another professional field that also saw the implementation of the same type of short-sighted ideologies and how they disastrously affected the professional cadre that make that field up.

The article will be out later this week...

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

Re: Excellent Write-Up
Enigmaticatious 10-Feb-16 15:05

Because methodologies are the best way to try and condense the way in which people work into good practices.

I think you are making the mistake of thinking that new methodologies are only created because old ones failed, rather than old methodologies not being suitable for certain types of work.

If someone comes to you with a quick project, your not about to tell them you will need a month of design time up front, to which you will produce a functional specification, then a technical specification and then after spending thousands of dollars you will be ready to actually start.

Is that a failure of the method? NO. It is a failure to understand the problem space and apply an appropriate methodology that suits that problem space.

So I repeat... "A project doesn't fail because of the methodology", it fails because of people, their own personal agendas, their lack of ability and inability to apply the correct methods in the right circumstances.

Do you blame a hammer for not solving the problem when you are screwing in a light bulb? Nope, you blame the moron who chose the hammer to accomplish this task in the first place

Sign In · View Thread · Permalink

The Church of Agile

netizenk 29-Jan-16 5:57

Imposed on developers by people who never wrote any code because it's "what everyone's doing" with a sole purpose of wasting our time and energy on daily meetings useful only to take you out of "the zone" and many hours wasted weekly on endless babbling wrapped in some fancy names someone incompetent came up with.
From my observations, it changes absolutely nothing, other than wasting more time in useless meetings, as good developers continue to write good code and deliver good solutions while bad developers continue to write bad code and deliver bad solutions.
It might make some sense for large teams building commercial software but it's a complete waste of time and energy for the wast majority of developers.
And we must accept it all on blind faith without any proof that it works or adds any value to the process so it's a religion, that's what it is...

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

Re: The Church of Agile

Steve Naidamast 29-Jan-16 7:27

Thank you for your input regarding my latest article.

Your comments makes me believe that you are aware of Human sociology relatively well as what you describe is basically what has happened to our profession in near totality in the business application sector.

I can only disagree on one point you made. "Agile" has been found to work successfully on very small projects or non-complex maintenance tasks. When applied to large-scale, enterprise projects analysts have begun to notice that "Agile" more or less completely fails as it is not designed to handle the amount of complexity involved in such projects. Thus, they have come up with a new format for using "Agile" in such projects, labeled "Enterprise Team Agile", which at first glance appears to be nothing more than adding multiple teams to work within the existing paradigm with different levels of milestones and meetings.

I had found a very interesting document on this subject, which I read through quickly but did not keep it for later reference.

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

Sign In · View Thread · Permalink

A good synopsis of the history of the movement

Member 11682335 29-Jan-16 5:31

In my own history, after working on teams that have tried all approaches, I have found that the most important paradigm in any team environment is *communication*. And this is where Agile has made great strides, by requiring teams to actively communicate on a very regular basis. Those of us who work with and around computers tend to be a fairly reclusive lot, and it's hard enough to get two or three sentences out of some people. But when you are working with complex interactive software (especially with today's web and SOA platforms), it is vital to have team members collaborating and not holing up in their cubes perfecting their Silos of Excellence.

Waterfall tends to encourage these people to do just that, and then report in when they are ready to check in code that meets the set of requirements that were current a month ago. Agile forces team members to interact on a daily basis, and then to interact with product owners at least every two weeks at the end of each sprint. And this is where Continual Delivery shines best, because the product owner gets a deeper insight into the features that are getting completed even if the entire product is not done yet, whereas with Waterfall, the product owner doesn't see anything until the product is complete unless they poke real hard for status updates.

Personally, I'm not a huge fan of Scrum, and tend to lean towards DSDM since that is the model where I have seen the most success, though I think Scrum might work if you have the right team make-up.

Sign In · View Thread · Permalink

Re: A good synopsis of the history of the movement

Steve Naidamast 29-Jan-16 7:20

Thank you for your valuable comments on my latest article.

Though I am no fan of the "Agile" movement as can be seen from my writing I can see that like any other development paradigm it can be refined to be more inclusive of the vital functionality that it has up to now more or less ignored (ie: requirements gathering). And it would also be a positive move if the people who work on it's definition would look to devise different forms of "Agile" that could be applied to different types of projects. I believe that Stephen McConnell's, Construx Software, is already doing this so he may be a very good model to use as a guide.

Nonetheless, the overriding problem I have noticed in the current software development industry is that there is too much reliance on development paradigms and new tools instead of good common sense. And with technology undergoing the many traumatic changes that has been fostered on it in the past few years, this is not a good sign for in-depth knowledge because as soon as the profession latches on to the latest craze, the knowledge of the tool or paradigm that was being used tends to go out the window leaving few paradigms and tools to be able to develop into mature products. A noted analyst, Hadi Hariri in England, made a very good case for this point in a recent lecture he gave at the jaxConference. You can view the YouTube video at this link...The Silver Bullet Syndrome by Hadi Hariri - YouTube

---

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

Sign In · View Thread · Permalink

Re: A good synopsis of the history of the movement

Member 11682335 8-Feb-16 6:38

I've never really been one to run away to the latest craze, and tend to not be much of a trend-follower, much less a trend-setter. In the end, it's all about what is effective for your team at the time. And for some teams, Agile works. For others (especially where you have to conform to specific standards) you have to approach with other techniques. But to lock into a particular paradigm just because it's trending is doing a disservice to the multitudes of intelligent people who came before you. At the same time, we must be willing to look at change and follow what makes sense, because failure to do so would eventually lead to dinosaur-dom (i.e., extinction).

In today's development climate, with so much of development becoming web-centric, we are hearing new buzzwords every month. Rapid application development, responsive design, single page application, MVC, service oriented architecture...and we watch some development models fall to the wayside, and whole platforms become obsolete quite literally overnight. And you can bet that the current framework which you are leading development into will be either obsolete or completely gutted and redeveloped within the next 18 months. Or someone will come out with yet another new web framework that makes the current method look like yesterday's jam. So why not embrace a more Agile development method? Yes, we could spend months collecting requirements, and put the developers to work writing documentation that completely exposes the beautifully designed architecture that they have written. And then throw it all away in 18 months when the product owner favors a new technology because it has snazzy widgets.

To be honest, have I ended up with code that I was embarrassed to pass along to the customer? Yes indeed.

Everyone has if they have been in the business long enough. But I do not usually have the luxury of bringing everything to a bright polish and must instead be satisfied with the fact that the customer has a usable product that fulfilled the requirements as expressed.

Sign In · View Thread · Permalink

Re: A good synopsis of the history of the movement

Steve Naidamast 8-Feb-16 7:24

I can't really disagree with your basic premises. However, I will say that terms like "rapid application development" and "Service oriented architecture" are actually rather old terms considering that the first made its debut in the mid-1990s and the latter in the late 1990s.

That being said, I will say the following...

Standard software engineering practices as they are now have been refined over many years of research and project analysis as well as experimentation. Such practices also provide enough variation in terms of life-cycle methodologies that there is literally something for any type of project you can think of. And no software engineer would ever say that all the assets of each of these paradigms have to be used to their fullest or even made part of the project. With each life-cycle model you are free to choose what is needed for your environment but simply follow the basic tenets of each.

"Agile" basically eliminates much of the basic tenets of good software design and implementation using a lot marketing hype to cover up its inadequacies. "DevOps" merely adds to this mess by attempting to dilute operational infrastructure for the needs of software development instead of having operations setting their own required standards, which are necessary.

Like manufacturing, software has to follow a certain set of processes to produce quality output. In manufacturing, the processes have been refined by process analysts who were leaders in their respective fields. However, none of the processes if any were eliminated just refined.

Software is fast becoming a diluted mess as even your own reply indicates. However, in my next article, which should be out later this week I demonstrate using a real-world example that mirrors the current ideological trends in software development today how "Agile" and it's offshoots will eventually lead to disaster...

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

Sign In · View Thread · Permalink

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin