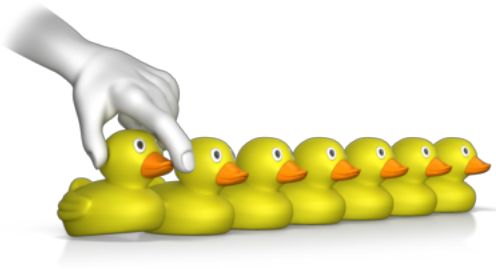
Version 9.4

Release Orchestration

Speed, relevance and quality



This document is a companion article to the presentation given to Agile Testing Days.

Potsdam, Germany 16 November 2018

Introduction

The explicit promise of all the disruptive initiatives of the past twenty years is to make possible the deployment of products with **speed, relevance** and **quality**. If not already, that should be the goal of any team or company developing software today. IT specialists involved in software development want to learn how best to orchestrate product releases to attain those three objectives. This paper attempts to explain how this can be achieved by focusing attention and effort on release orchestration.

In the context of release orchestration, the first of the three objectives, speed, means release frequency. How often are new features released into production. Relevance is the applicability of features to stakeholder requirements. Relevance is answering yes to the question, are you releasing features that the market wants? Defining quality is not easy. It means many different things to many different people. It is also a good way to start a serious argument or a bar fight. But for the purposes of this discussion I simply mean, does the deployed product do what all stakeholders expect it to do in the way they want it?

Release orchestration is sometimes considered to be little more than an evolution of automated release management. Release management used to be the work of that semi-mythical creature, the release manager who rose from the depths of the development team a couple of days prior to each big release, to piece together the disparate and often incompatible threads of the project. Only this hero knew the code well enough to lead, tirelessly from the Thursday afternoon code freeze, through the nights (isolated, alone and often too dangerous to approach – like a werewolf during a full moon), over the weekend to assure that everyone could start fixing bugs in production bright and early Monday morning.

That may sound like ancient history to some. For others it might be this coming weekend. Whatever release management may have been in the past, we should consider carefully how and how fast that responsibility integrates into modern development and operations *processes.*[1] To move from a process based on a manual release to one where releases are continuous and automated, is difficult for reasons far beyond the technical challenge. The change represents a paradigm shift and that is problematic for anyone: organisations, teams and even people do not change so dramatically from one day to the next. The speed and magnitude of a transformation to DevOps is not only disruptive and difficult, even more frightening, it is not something that unfolds neatly. As with most paradigm shifts, it is anything but a clean transition from *before* to *after*. As a matter of fact, the process usually tends to be rather messy.

The reality in the industry is that companies are at many different levels of DevOps maturity and delivery setups are fragmented across industries. Further, there exists different and often conflicting budgets and contractual obligations across internal and external teams who themselves are as often at odds with classical development teams. Add to this the many monoliths with dependencies and you have what appears to be a fragmented and disoriented release universe. One would be forgiven for believing that we are headed toward a state of maximum entropy. Almost. We should remember that we are in the middle of a paradigm shift where the rules of the road are still in flux. That is not to say, however, that there is nothing to be

[1] The rubric of development and operations processes includes all elements of agile development and CI/CD practices.

said about the direction of release practices. For this article, I focus on a release process represented by archetypal DevOps practices while highlighting elements of release orchestration. In this way, I hope to isolate the three, aforementioned primary elements of speed, relevance and quality to see what needs our attention most.

Today a release is much more than automated release management. It expands to include infrastructure and release automation which, though both are important, are often relegated secondary concerns in a comprehensive release orchestration process. To be clear, by release I mean the deployment of a product or service to a production-like environment. Release orchestration, on the other hand, is several components which, taken together, are a culmination of the promise of Continuous, Integrated Delivery. The components in the release corpus are:

- **Tools and Best Practices**. The release toolchain has matured to where now we can automate almost all of what the release werewolf did over release weekends at **speed**.
- **Integrated Workflows**. Important and now common and understood. Tools are properly integrated with one another and we understand get information on which to base our next set of features. These feedback loops from customer activity along with CI monitoring allow for product **relevance**.
- **Skills and the mindset**. Development and operations staff have adapted to the requirements of shared responsibility for the end result. **Quality**.

All this combines to make a better, faster SDLC which is not relegated to development teams working in isolation but working together with operations to get well-tested, desired features to customers as rapidly as possible: Quality, relevance, and speed.

Release toolchain

Until the *processes* associated with a release are automated, many companies continue to rely on tools to, in the best case manage or, in the worst case simply document, the steps. Excel or the equivalent, are static tools which document existing practices, and which are still in widespread use. Excel, though static is probably still better than say a pencil for managing deployments. Though maybe not by much. If you do have only manual processes a spreadsheet is as good a record of truth as any. If, however, you want to run microservices, a pre-requisite is automation and orchestration. If you need help getting from manual to automated releases there are many tools which claim to help migrate from static, manual process to automated steps though these alone are usually insufficient. What is critical is to get tooling aligned with ways of working: integrated and iterative

Again, *the choice of tools will, and should usually be, secondary*. Choose tools that suit the work being done. Don't decide how to work based on the tools. Otherwise, it akin to pounding nails and drilling holes with a screwdriver because that is what is what is at hand.

There is the assumption, still widely held, that the tools a team or company uses is the principle stimulus to adopting new ways of working. We've all heard it: "oh, once you adopt *fill-in-the-blank*" the argument goes, "your developers will improve their output, quality will improve, time-to-market-will decrease, the sun will shine 24/7, whatever". There is just enough truth in the assumption to make it seem valid: tools do impact how you work far more important is the culture of a company. With a hammer you can certainly pound a nail into a wall better and faster than if

you use a screwdriver. But, if you *are* using a screwdriver, the problem is not the screwdriver. The root cause of the problem person who or process which decided you should try to use a screwdriver in the first place! And, a hammer is not always the right choice. A nail gun might be an even better choice if you are staring at 4000 nails.

In any case, he corporate decision-making process that came up with the screwdriver method is company culture. It is far more invasive and influential to ways of working i.e. *corporate processes* than tools alone will ever be. And by culture, what we really mean is the ingrained mentality which is unwilling or unable to adapt to change. The famous "culture eats strategy for breakfast" (often, though falsely, attributed to Peter Drucker) is the usual retort to such claims. The warning signs for the software industry, however, are clear. The 2018 DevOps Report stated the problem with unusual clarity:

> *Organizations often focus on buying solutions in the form of tools, vendors, and methodologies.* ***However what's important is the capabilities these solutions enable, not the solutions themselves.***[2]

Despite the symbiotic relationship, the culture in which the tools are used is the critical component, not the tools themselves. So, when we talk about DevOps, Release Orchestration or SRE, the tools we employ are important, but ultimately secondary. Once we have achieved full or nearly full release automation, we are able to push features out at speed. For all that though, we still cannot assure the quality of the product and by quality, we include the usual metrics as well as relevance and customer satisfaction.

Releasing Relevant Stuff

The speed and levels of integration available to release products today far exceed what was possible in the recent past. The workflow can be a system where all components can (or at least should) interact via common interfaces and produces results that can be understood or acted upon automatically. It is not simply *automation* that makes this model different, it is *integration*. That integration is supported and refined with transparent feedback loops. The following is an example of a generic CI/CD flow. Note the orange lines which represent feedback loops.

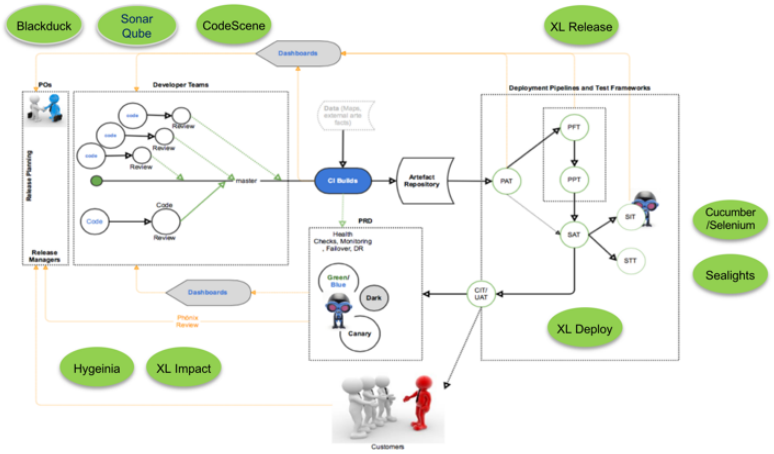


Figure 1 An example of an integrated workflow

[2] "State of DevOps 2018", DORA, 2018.

The workflow in figure 1 above shows an idealised pipeline in which each tool and process is integrated.

- Tools work together seamlessly providing consistent information over compatible interfaces.
- Customer feedback is integrated into the lifecycle. This is one of the missing steps in many pipelines. If practiced honestly, puts product at the centre.
- Continuous feedback from all sources – ideally in real-time - allows continuous analysis of results. When acted upon, this is information enables continuous improvement of the SDLC.

This next-generation, automated SDLC begins to resemble something more appropriate for continuous delivery: short release cycles, limited delays, strong communication loops. This new development "pipeline" more accurately represents reality – visualised here as an infinity loop.

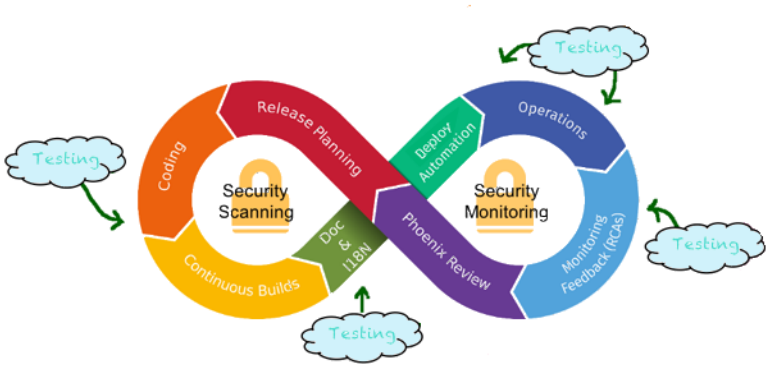


Figure 2 A modern SDLC "pipeline"

Not only is wasted effort in the release cycle easier to identify, but security and testing modules are inserted at any stage in the process. In this model the functional product is of central importance. To paraphrase Martin Fowler, we organise around the product.[3] There is no manual change approval process! And, continuous delivery becomes are reality.[4]

This is indeed a revolutionary step. *It is changing the paradigm from a pipeline to an infinity loop* (or circle). Implemented in this way, the process does not pass through any speed bump-like "gates" and, more importantly, the cycle is not a one off: Instead of passing through the pipeline a single time, the cycle repeats as many times as required.

Compared to traditional or waterfall development patterns, the ideas and practices associated with, and enabled by Lean, Agile, CI/CD, and DevOps practices[5] provide a new model in which there is no defined start or definite end. Therefore, there is not no "release" *per se* as in the old paradigm. Instead there is a series of small improvements pushed into production rapidly and regularly. Metaphorically, these "releases" are waves.

---

[3] "The State of Agile Software in 2018", Keynote to Agile Australia, 25 August, 2018.

[4] Purposely, I have also left out manual testing which though it is outside the scope of this paper, still have a role in software development even today.

[5] Throughout this article I use the terms "DevOps" and "DevOps transformation" as an umbrella phrases or proxies for both the disruptive initiatives (Agile, Lean and DevOps) as well as a model SDLC: continuous development, delivery and improvement. I have taken this liberty because in this article I am primarily interested not in the mechanics of release orchestration but in its transformation as an idea and of course its shortcomings.

Are the methodology wars over then? One might be tempted to claim victory: stand on the aircraft carrier with the banner “mission accomplished” in the background. It is tempting to look upon the tools and methodologies at our disposal and be lulled into thinking that the victory is complete. After all, look at a sample of disruptive initiatives that are available today:

- DevOps / SRE skill sets
- Infrastructure as a Service
- Lean and Agile Practices
- Test- and Behaviour-Driven Development
- Extreme & Pair Programming
- Continuous Integration / Continuous Delivery
- Continuous Deployment
- Production Testing

The DevOps transformation appears to have a pretty good handle on speed. Challenges still remain, however, with quality and stability. To make some sense of what remains, let’s take a step back to look at this in a larger, historical context.

A quarter century ago, there were several voices rising above the rest asking questions about the quality of software being produced. One of those with which most people are familiar (or should be) is Frederick Brooks. In his famous Mythical Man-Month,[6] he challenged developers in 1975 to consider efficiency seriously. He did so again even more forcefully in 1995, in his twentieth anniversary edition he admonished the industry for ignoring quality. Contemporaneous with and complementary to Brooks’ work, Capers Jones, a second voice, identified quality problems plaguing the industry.[7]

1 Lack of a practical working definition of what "quality” means for software
2 Inadequate defect prevention
3 Inadequate use of reviews and inspections
4 Insufficient or careless testing
5 Lack of quality measurements
6 Lack of understanding by senior and project management that quality is on the central path
7 Excessive scheduling pressure leading to unwise attempts to short-cut quality control techniques
8 Unstable and ambiguous user requirements

None of the above should come as much of a surprise to people actively involved in software development and what is most striking about these is that they all could have been written in 2018! But they were written in 1993. Over the course of the past 25 years, the industry has made great advances in software languages, hardware as well as development and delivery practices. Yet for all the innovation and optimisations, a significant number of foundational problems persist.

[6] Brooks, Frederick P. Mythical Man-Month: Essay on Software Engineering, Addison-Wesley, 1995.
[7] Jones, Capers, Assessment and Control of Software Risks, Prentiss Hall, 1993.

A fundamental problem that Release Orchestration has yet to solve fully is the how to improve both time-to-market AND user satisfaction simultaneously. In other words, how to align the three elements of speed, relevance *and* quality. In some ways, it is Release Orchestration's equivalent of the three-body problem.[8] But unlike the mathematical problem, ours is solvable.

Things have changed in the past 25 years

The IT and development world have not been static since 1993. Far from it. So why do we still find similar, or in many cases, identical issues especially in software quality? It is not difficult to conjecture what accounts for this persistence. Even leaving aside for the moment that much of the software in the mid 90s was "physical" i.e. delivered with a code name, version number, on a predetermined release date, and as often as not on a physical medium, the similarities with how software is developed today outnumber the differences. In fact, development practices persistently almost obstinately continue to mirror manufacturing processes.

If development stubbornly maintains traditional way of working, then the results will only vary in style not substance. By that I mean, changing the ways of working to increase speed of development and deployment has tangible effects on release cycles and frequency but it does not have significant impact on stability and quality. Consider what has changed since 1993.

- Processes and software management have evolved. This is the primary and foundational change leading to increased release frequencies.
- Organisations are changing. Organisations, like society tend to adapt slowly but change erratically.
- The nature of valuable assets has changed. From operational expense to intangible asset, both IT and Development assumes a new position on accounting reports.

By doing this, we isolate the problem and can identify where our focus should be. Let's look at each in turn.

Processes

Using Value Stream Mapping adopted from Lean, we can inspect (by duration) each step in our development and release process - from planning phase to Phoenix Review (retrospective). There are, however, two important flaws in the map below: it omits product teams and it is flat. More accurately, it resembles a flat pipeline.

[8] The **three-body problem** is the problem of taking the initial positions and velocities of three point masses and solving for their subsequent motion. It is far from being completely understood. For more information on this fascinating topic, visit Scholarpedia.

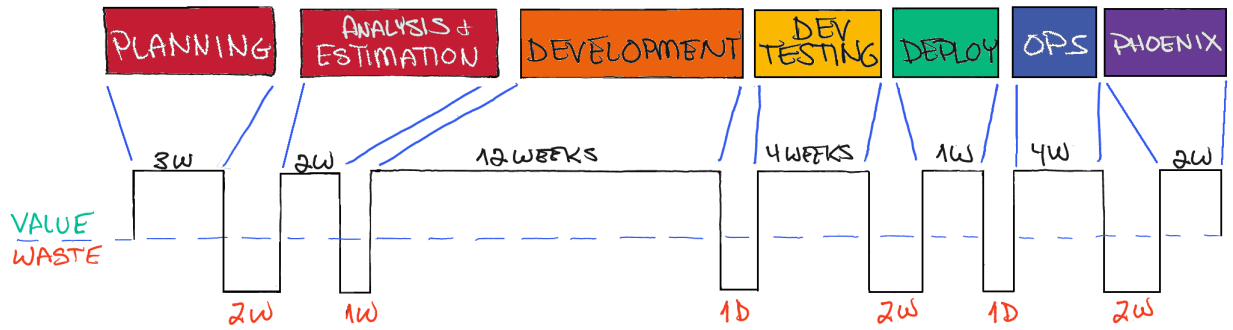


Figure 3 Value Stream Mapping Current State (estimation/analysis)

The following map (from a different project) addresses the first problem by adding in a step for "feature request."

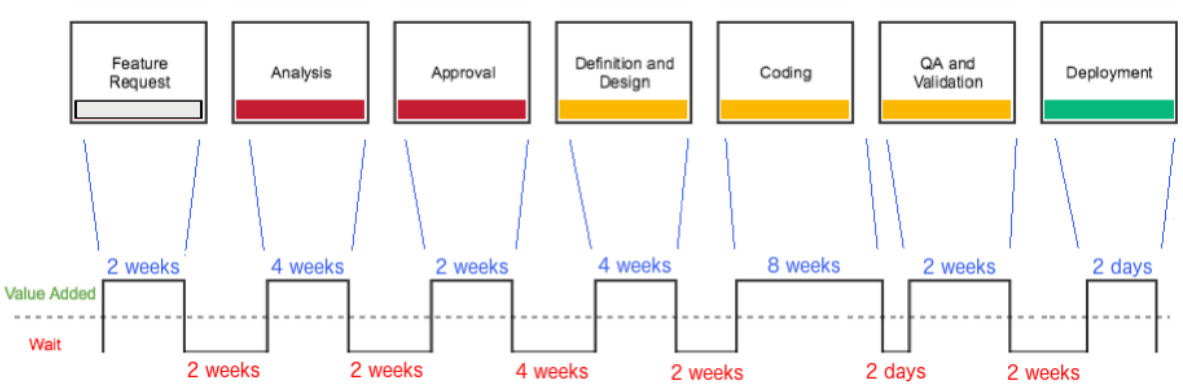


Figure 4 A product-centric view of the first Value Stream Map with Feature Request (note missing Phoenix Review)

The difference is superficial and alone, it is not entirely sufficient. There remains the question of the feedback loop: the connection between customers and product decisions. How long, for example, does the process or step take? How does it work? That is where the flat, pipeline image we are so used to fails to convey an accurate picture.

Next generation processes demand better collaboration and communication channels between customer, operations and program or project teams and leaders. The second shortcoming of standard VSMs, flatness, is better visualized in a circular map as shown in *Figure 5.*

Figure 5 Next Generation Value Stream Mapping

A circular Value Stream Map has the advantage of making visible the gap between *Phoenix Review* and Planning. This gap - represented here by the wedge "PRODUCT OWNER" - is really a handoff to another group, in another department. This delay is critical and often either missed or ignored. Part of the reason for this is the persistence of silos within our organisations. Usually the Product Team is isolated from the development teams either partially as a separate team, or completely as part of another organisation entirely. Other gaps too often identify handoffs to other departments. The gap between "ESTIMATION" and "DEVELOPMENT" also represents a delay for finance or senior management to calculate and approve the cost of developing a feature or paying down technical debt. There is also a gap between "DEV TESTING" and "DEPLOY" (in many projects a manager or end customer insists on the infamous "stabilisation" or "quality control" phase). In these cases, the value stream map is distorted further.

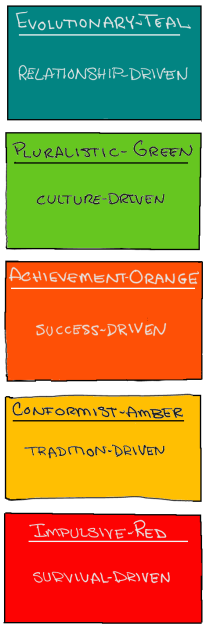
Figure 6 Extended Value Stream Map with external stakeholders

Figure 6 shows a value stream map in which stakeholders outside the traditional DevOps lines of reporting to other departments. Finance may be involved in the decision to continue with a feature or refactoring. The customer, an internal QA team, the security team, or even an external testing company is often inserted between a completed product and a deployment to production ("QA TEST"). This is often seen in industries with regulatory or compliance constraints but is just as often a holdover from the "test-it-before-it-goes-out-the-door" school of Quality Control. Lastly, senior leaders or the leadership team ("SR. LT") are often involved in the evaluation of products after they are in production. This may come in the form of escalations from customers, sales requests, etc. These are examples of out of channel communications which are, unsurprisingly, a regular feature in many organisations. This are, however, a wasteful process and the existence of such a path is arguably an anti-pattern symptomatic of other, usually larger organisational deficiencies.

Organisations

The second area of change is organisational structures and how they interface with transformation initiatives. Obviously, the full story of the relationship of organisation to transformation is well outside the scope of this article. It does, however, have a significant part to play so we need to describe the relationship, if only in broad strokes. As suggested above, organisational change is not a smooth process, it usually occurs in fits and starts. This is as true for evolution, as it is for society. And a commercial organisation does not operate according to rules incompatible with the society in which it exists.

The 2014 book, Reinventing Organizations, by Frederic Laloux outlines transformational steps through which societies move. These range from societies defined by command authority, those with formal roles and processes, accountable meritocracies, value-driven cultures, and beyond. Drawing from research in related scholarship, he categorises organisational development steps along similar lines, assigning each a representational colour: red, amber, orange, green and teal.



The modern, corporate structure of the late twentieth century (orange) is achievement-oriented and success driven. Its goals are to beat the competition and to achieve profit and growth. Innovation is the key to staying ahead. It remains, however, hierarchical and practices management by objectives (**command and control on what**; **freedom on the how**). This model has been around since at least the 1950s.

A culture-driven, pluralistic corporate model (green), on the other hand, assumes that all perspectives deserve equal respect. It seeks fairness, equality, harmony. community. cooperation, and consensus. Green organizations often strive to inspire their employees to great things, leading them to outperform more traditional command-and-control organizations. Laloux identifies several companies (Southwest Airlines, Ben & Jerry's, and The Container Store) as examples of organizations whose founders have championed Green organizational practices. While one can debate which companies can or should be included in the list, it does not take away from the importance of understanding the concept that there are other ways in which some companies are defining core values and corporate models.

Moving from one stage to another does not *replace* the principles of the previous one but incorporates them: it is a revision, not a rejection of the previous model.

Like evolution, moving from one stage to the next is usually a difficult feat. It takes time, is rarely uniform, and the transformation is never complete.

Though controversial in some business circles, these kinds of categorisations are not particularly new elsewhere and in academia. Dr Ron Westrum did something similar for the health industry in 2004. He categorised orgs as Pathological, Bureaucratic and Generative.[9] As far back as 1992, the differences in corporate culture were understood to have a tangible impact on performance. In the book Corporate Culture and Performance, research suggested that "value-driven organisations outperform their peers by wide-margins."[10]

Laloux argues convincingly that organisational change mirrors change in human consciousness and that transitions are evolutionary stages.

> *"Human consciousness evolves in successive stages; there is no wishing away the massive amount of evidence that backs this reality. The problem is not with the reality of the stages; it is with how we view the staircase."*[11]

DevOps is an evolutionary step forward. It is a mindset as much as it is a methodology: meaning that there is no "checklist", no right way to do it. Indeed, it is the *how* as much as the what that really makes for a successful transformation. Because the disruptive initiatives not exist in a vacuum in order to realise a functional, efficient DevOps-based organisation, the transition must be nurtured by a corporate culture that accepts innovation, risk and change at the same time that it helps mould that very culture. Laloux's proposition is that organisations are evolving, and we already know that the disruptive initiatives first emerged from the bottom-up within software organisations. Therefore, we will want to evaluate our DevOps Transformation activities through that lens in order to fully appreciate the implications of our actions and guide future plans.

A July 2017 article from McKinsey argued that "shortcomings in organizational culture are one of the main barriers to company success in the digital age." Success in the digital age is explicitly what DevOps hopes to achieve. The article continues by proposing that there are "…three digital-culture deficiencies: functional and departmental **silos**, a fear of **taking risks**, and difficulty forming and **acting on a single view of the customer**." [12] It can be argued that the first is simply an organisational challenge although that too is dependent on corporate leadership and therefore indirectly cultural. The second and third, (risk taking and customer-orientation) however, are directly results of an organisation's attitude and are fundamental to understanding the rise of the disruptive initiatives. Agile, Lean, DevOps, CI/CD, TDD, BDD, etc are not the stimuli of corporate change. They are disruptive initiatives enabled by change happening at other levels entirely.

---

[9] Ron Westrum. "A Topology of organization culture." 2004.

[10] Laloux, Frederic. Reinventing Organizations: A Guide to Creating Organizations Inspired by the Next Stage of Human Consciousness, Nelson Parker, Brussels, 2014, p. 33. See also notes p. 335. Kotter and Heskett established that companies with strong business cultures and empowered managers/employees outperformed other companies on revenue growth (by a factor of four), stock price increase (by a factor of eight), and increase in net income (by a factor of more than 700) during the eleven years considered in the research.

[11] Laloux, p. 37.

[12] Goran, Julie, et al, "Culture for a Digital Age", McKinsey, July 2017.

Historians have long appreciated that the catalyst for change does not come primarily from technology but from organisational and social change. Again, to adopt an historical perspective, a recent article in the New York Times sums up this idea concisely.

> *„The history of labour shows that technology does not usually drive social change. On the contrary, social change is typically driven by decisions we make about how to organize our world. Only later does technology swoop in, accelerating and consolidating those changes. "* [13]

Leaving aside cause and effect for the moment, there can be little doubt that a synergetic relationship has emerged. Agile and DevOps changes the way leaders can and do operate while the options made available to corporate leaders provides an environment in which DevOps can thrive. Ideally, everyone wins.

But that future is not prearranged: any of the disruptive initiatives can still lose momentum. It may not become the release-operations paradigm of the future. At some point, whether from fear or ignorance, corporate leaders might abandon the innovators. Assaults on inefficiencies and waterfall may just as quickly and easily be turned back by the forces of tradition. Though recent studies show that Agile reversal rates are dropping[14], the jury is still out on other disruptive initiatives like DevOps. In any case, history does not assure continuity and we should avoid complacency or misplaced optimism.

Development and operations initiatives need to consider, to a greater degree than most people have done up to this point, the dynamics of a changing corporate world. Until now, all the disruptive initiatives have been interesting, moderately successful though mostly technical and ultimately peripheral phenomena in the corporate world.

A Modern Company and its Assets

One way to measure what is valuable to a company, is to look where it chooses to invest. How important DevOps and the disruptive initiatives are in the hierarchy of corporate importance is directly proportional to investment in both money and time the company is willing to commit.

The roots of modern corporations can be traced back well into the past: Managerial capitalism in the last century, joint stock companies of the 17th and 18th centuries, chartered companies of the 16th.[15] As organisations changed, adapted and grew, the way they viewed themselves and their value also changed. And it continues to do so. One way in which companies have begun to view things differently is in how they are valued. In their 2017 book, Capitalism without Capital, Jonathan Haskel and Stian Westlake trace the rise of corporate valuation alongside the increases in investment in *intangible* assets. They demonstrate that intangible assets - brands, supply chains, software and processes, for example – assume an increasingly important role in assessing not only the value of a company but its focus and direction. An asset is defined as "an economic resource that is expected to provide a benefit over a period of time."[16] In this definition, how an

[13] Hyman, Louis. „It's Not Technology That's Disrupting Our Jobs, "New York Times, 18.08.2018.
[14] Reifer, Donald J., "Quantitative Analysis of Agile Methods Study (2017): Twelve Major Findings", Online whitepaper, posted 10 August 2017.
[15] For an interesting and informative historical summary of companies, see John Mickelthwaite and Adrian Woolridge, The Company: A Short History of a Revolutionary Idea, 2003.
[16] Haskel, Jonathan and Stian Westlake. Capitalism without Capital: The Rise of the Intangible Economy, 2017.

organisation looks on its developers and operation people – the "resources" responsible for some of these newly important intangible assets – also changes.

Broad category	Type of investment	Type of legal property that might be created	Treated as investment in National Accounts?
Computerized information	Software development	Patent, copyright, design IPR, trademark, other	Yes, since early 2000s
	Database development	Copyright, other	Recommended in SNA 1993, but OECD suggests uneven implementation
Innovative Property	R&D	Patents, design IPR	Yes, recommended in SNA 2008, introduced gradually since then
Economic Competencies	Training	Other	No
	Market research and branding	Copyright, trademark	No
	Business process re-engineering	Patent, copyright, other	No

Note: R&D should be thought of, in line with official definitions, as scientific-oriented spending as distinct from, say, artistic or design endeavors. "Other" in column 3 refers to things like trade secrets, contracts, etc. Column 3 refers to formal intellectual property: we would expect all intangible investment to produce tacit knowledge as well.

Source: Columns 1 and 2 from Corrado, Hulten, and Sichel 2005, column 3 based on Corrado 2010, column 4 from Corrado et al. 2013.

Figure 7 Haskel and Westlake, types of intangible investments [17]

The disruptive initiatives, like other internal product or process, are economic competencies which become valuable assets when practiced and honed to high efficiency. They may even achieve a significant level of market differentiation. By this measure, engineers, perfecting release orchestration processes are actively producing value for the company. *Disruptive initiatives like DevOps create value*: investment in production assets. They have become producers of value where once they were simply an operating expense. This is significant in several ways. Because the nature of corporate investment has been shifting toward intangible assets, products and processes become one of the greatest assets of a company engaged in developing software and operating services.[18] And processes like Release Orchestration have gone from being operational costs to valuable assets. We can see from *Figure 7* that in the US this transition occurring in the mid-1990s.

[17] Ibid. p. 56.
[18] Ibid.

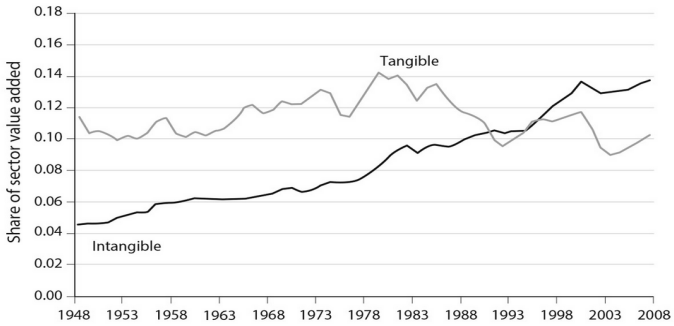


Figure 8 Corporate investments - United States [19]

In Europe, the transition took longer but now intangible investments exceed tangible in most economies. The degree of investment is a good proxy for how a company judges what is important.

Finally, let's double back to the intersection of release orchestration and quality, which is really the challenge that remains. Major innovations and movements have pushed many industries in the right direction. The disruptive initiatives - Agile, Lean, SRE/DevOps, etc. - have all contributed to improving our three primary elements: speed, relevance and quality. But, we are still faced with problems like:

- Insufficient or careless testing
- Lack of quality measurements
- Excessive scheduling pressure
- Unstable and ambiguous user requirements

Each of these represent a gap in Release Orchestration that needs to be filled.

To address these, we need to extend our vision to include those groups not generally included under the DevOps innovation umbrella: Product and Program, middle-level management, C-Level, even HR. In many ways, the modern corporate structure is still closely modelled on the design of the joint stock company of the nineteenth century. This needs to change.

Conclusion

The full realisation of the promise of Continuous Delivery and the evolution of processes, organisations and our understanding of assets. Remember, we started out by defining release orchestration as a release with SPEED, RELEVANCE and QUALITY.

There still remains what practical steps are required to achieve the promise of efficient development and deployment and all that implies: speed, relevance, quality. What KPIs are available to measure release efficiency?

Not only is release orchestration part of every step in the process, it has emerged as a valuable asset to the company and should be treated as such: an asset deserving attention and investment.

Finally, and perhaps the most crucial improvement which still needs to occur with the disruptive initiatives is the inclusion and integration of product. The role of product or program management

[19] Ibid. p. 36.

in the Release Orchestration / DevOps process needs to be strengthened and improved. And it cannot be a stand-alone or independent element within the release orchestration process.

The DORA State of DevOps 2018 identifies 3 keys all of which reinforce the idea presented both in this document and in the presentation on which it is based:

1. The extent to which teams segment products and features into small batches that can be completed in less than a week and released frequently, including the use of minimum viable products (MVPs)
2. Whether organizations actively and regularly seek customer feedback and incorporate this feedback into the design of their products
3. Whether development teams have the authority to create and change specifications as part of the development process without requiring approval

The first two correspond to speed and relevance. The third is our quality problem and how we can fix it. DevOps advocates have earned a seat at the table – and rightly so. They become one of the keys to success. But not all leaders understand this yet. Laloux has argued that "an organisation cannot evolve beyond its leadership's stage of development." Experience has demonstrated that fully successful transformations cannot succeed without a clear vision and the full and vocal support of the LT. The so-called *middle management* must commit to a culture and way of working which emphasises security and quality in all phases of the development cycle and they need to bring senior leaders onboard when possible.

This is the current phase of the DevOps revolution. To continue its forward momentum, the advocates of the DevOps movement need to overcome the cultural resistance which stifles innovation – especially where it has been internalised. As practitioners, we need to stop reinventing square wheels and focus on improving what works. And we need to improve software management capabilities by integrating all stakeholders into the software development and delivery lifecycle.

Peter Caron
Berlin, 2018

Broad category	Type of investment	Type of legal property that might be created
Computerized information	Software development	Patent, copyright, design IPR, trademark, other
	Database development	Copyright, other
Innovative Property	R&D	Patents, design IPR
	Mineral exploration	Patents, other
	Creating entertainment and artistic originals	Copyright, design IPR
	Design and other product development costs	Copyright, design IPR, trademark
Economic Competencies	Training	Other
	Market research and branding	Copyright, trademark
	Business process re-engineering	Patent, copyright, other