SAND2021-6250R

The DevOps: A Concise Understanding to the DevOps Philosophy and Science

Brandon T. Klein
Sandia National Laboratories
Albuquerque, New Mexico
Email: btklein@sandia.gov

Abstract—**This writing examines the DevOps practice from a new perspective, one of understanding its philosophical and scientific nature. DevOps has fundamentally changed the landscape for research and development based on guiding philosophical and scientific principles. Advanced computational technologies and domains have adopted DevOps to enable advanced solution engineering for efficient, quality-assured output. The author provides a concise account of how the philosophy and science of DevOps synergistically defines its essential disposition.**

1 INTRODUCTION

DevOps—A successful paradigm to some and a mythical, esoteric practice to others. The internet community has generalized the definition of DevOps as a set of practices that combines software development (Dev) and Information-Technology (IT) operations (Ops) with the intention to shorten systems development lifecycle (SDLC) and provide continuous integration and continuous deployment of high-quality software (DevOps, 2020). This writing proposes a concise understanding of DevOps by describing its underpinning philosophy and science and how these domains guide its success.

For the scope of this writing, the term DevOps encompasses the industry term DevSecOps ([Dev]elopment + [Sec]urity + [Op]eration[s]). Security should be implicit and omnipresent to the DevOps paradigm and practice. Security throughout DevOps is connected to quality assurance to provide efficient, quality assured solutions through methodologies such as SDLC (Laskowski, 2011). Security is everyone's responsibility through SDLC and has been championed through the technology industry (Vogels, 2017). DevOps originated in the software development and IT community as a compilation of practices performed within a larger practice; however, the paradigm and principles of DevOps are reusable for general technology solutions engineering and beyond (Sajjad, 2020). The following sections provide a concise understanding to the DevOps philosophy and science, which synergistically define its disposition.

2 TAXONOMY

The following proposed taxonomy for DevOps foundationally encompasses the domains of Philosophy and Science. Figure 1 illustrates the proposed DevOps taxonomy for Philosophy and Science along with their respective branches, subdomains, and instantiations. Under the domain of Philosophy, the branches of Epistemology and Ontology will be examined through additional classification of Belief, Knowledge, and Truth. Accommodating instantiations of these subdomains will be further examined through concepts and practices such as: Agile development, Quality Assurance, and Lean production. Under the domain of Science, the subdomains of Social and Applied will be examined through additional decomposition of Sociology, Psychology, Economics, and Engineering. Accommodating instantiations of those subdomains will be further examined through concepts and practices such as: Culture Transformation, Faster-Time-to-Market, and Automation.

3 PHILOSOPHY

The use of "philosophy" for this writing, will be lexically attributed to the Cambridge Dictionary definition, "a particular system of beliefs, values, and principles" (Philosophy, n.d.). The DevOps philosophy proposed here is continual improvement and delivery of efficient, quality assured solution engineering through industry recognized methodologies and practices. The following section is not intended to be exhaustive; but is rather a

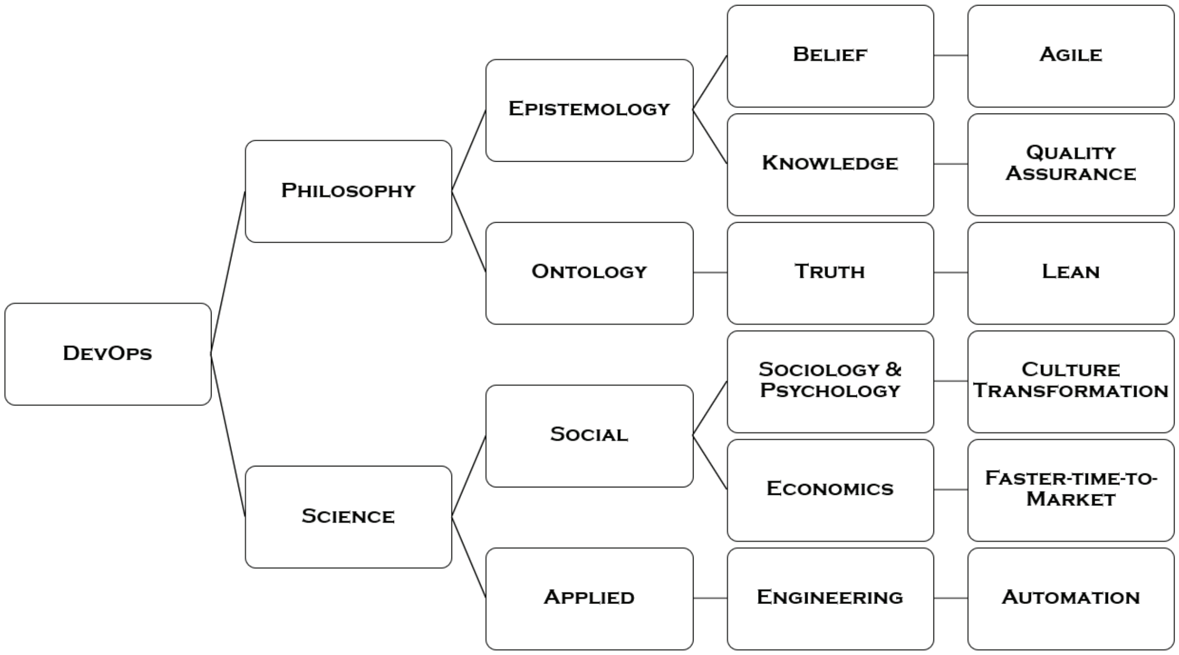

Figure 1 DevOps Philosophy and Science Taxonomy

scholastic recognition of the DevOps philosophy and core philosophical principles through examination of the associative branches: Epistemology and Ontology.

3.1 Epistemology

According to Oxford University Press, the definition of Epistemology is “the theory of knowledge, especially with regard to its methods, validity, and scope”; the definition further states, “Epistemology is the investigation of what distinguishes justified belief from opinion” (Epistemology, n.d.). The following sub-section explores academic and industry sources from which DevOps draws knowledge, such as: concepts, frameworks, methodologies, and practices. The epistemological understanding of DevOps is further explored through supporting sub-domains: Belief and Knowledge.

3.1.1 Belief

Cambridge Dictionary defines belief as “the feeling of being certain that something exists or is true” (Belief, n.d.). While DevOps may imply a variety of beliefs, the fundamental belief in DevOps explored here is Agile. At its core, DevOps holds the essential belief that there should be a reduction in time (speed/agility) of development to operations for a quality assured solution (output). It strives to reduce and shorten the feedback loop within ideation, development, production, and output. Ergo, the belief to employ a rapid, efficiently responsive methodology is essential to DevOps and is thus, a shared affinity with the Agile methodology.

Agile methodology, or Agile, is a belief held by the project management community in the efficacy to plan, execute, develop, and produce an output based on quick, iterative development and collaboration cycles, known as Sprints, amongst varying teams. Businesses, federal and academic institutions, along with professional bodies developed and adopted various approaches to manage output (e.g. products and services); from Traditional Waterfall to an evolved Adaptive approach of Agile (Satzinger et al., 2009). Agile originated from a collection of software professionals who desired to improve the software development process. The collective software professionals scribed the famous Agile Manifesto, the foundation of which stemmed from the basis of 12 principles that collectively established the belief structure of Agile (Beck et al., 2001) and later for DevOps.

Businesses as well as federal institutions demand accelerated solution engineering for time-sensitive and mission critical work. The need for speed in solution engineering of mission critical work is so pressing, one format redesigned the distinguished Sprint process of Agile into a complementary format known as—the Design Sprint. The purpose of the Design Sprint distills problem solving of mission critical needs into five days: Day One—strategize the target; Day Two—ideate solutions; Day Three—select a solution; Day Four—

develop a prototype; Day Five—test the prototype and capture lessons learned to decide how to progress next (Knapp et al., 2016). The Agile philosophy permeates DevOps and establishes agility within the practice. The concepts and practice of DevOps rely on speed to quickly engineer efficient, quality assured solutions, and through integration and development to continually improve and deliver desired solutions.

One method DevOps applies the Agile belief is through CI/CD, a tech-industry term associated with Continuous Integration and Continuous Delivery. This understanding, however, should be augmented to include: CI as Continuous Improvement and CD as Continuous Development. This augmentation to CI/CD suits the need to continuously develop from the continuous integration. Additionally, the purpose for continual integration is to improve the previous iteration and to continuously deliver said improvement as soon as possible. Thus, the augmented CI/CD model is an incessant CI/CD (Continuous Improvement and Continuous Delivery) cycle with aggregated sub-cycles of single and/or multiple CI/CD (Continuous Integration and Continuous Development) pipelines. Implementation of CI/CD is tactically executed through software engineered pipelines and can range from simplistic (one pipeline) to complex (many pipelines). Implemented pipelines may be interconnected or mutually exclusive; architecture of CI/CD pipelines is tailored according to need.

The CI/CD model is applied to software development and IT operations to enable accelerated delivery of solution output. Any integration should be an incremental improvement to the output. Thus, any development should be driven by the need to integrate decided improvements that are driven by the feedback loop cycle of continuous improvement in order to provide continuous delivery of improved output. The applied Agile belief through CI/CD in DevOps allows output to fail-fast, learn/improve quickly, and shortens the feedback loop cycle regarding knowledge.

3.1.2 Knowledge

Knowledge, according to Oxford Learner's Dictionary, is "the information, understanding and skills that you gain through education or experience" (Knowledge, n.d.). Vast technical understanding and skillsets are paramount to efficiently practice and execute DevOps. The knowledge inputs for practicing DevOps are numerous and multifaceted as illustrated through current hiring qualifications within the leading technology industry. The term "T-Shaped Professional" describes DevOps professionals who require broad technical knowledge in conjunction with profound specialized technical knowledge (DevOps Institute, 2019).

In addition to the knowledge needed by the DevOps practitioner, knowledge surrounding the DevOps process encompasses an array of abstract and concrete information spanning multiple technical and non-technical disciplines and domains. Yet, there is a fundamental confluence of the various disciplines and domains regarding desired knowledge—continually improved output through efficient, quality assured methodologies.

Recognition regarding the importance of knowledge capture, expression, evaluation, and retention integrated in a feedback loop cycle has been accounted for through formal concepts such as: Shewhart's Cycle for quality control—specification, production, inspection (Shewhart, 1939); and Deming's rendition for quality control and improvement—Plan-Do-Study-Act (PDSA) (Deming, 2018); and the Japanese rendition of Deming's work into Plan-Do-Check-Act (PDCA) (Imai, 1986). For example, when a business develops a strategy (Plan) to create a new widget, the release of the widget to the market (Do) allows the business to collect market information regarding the widget for analysis (Study/ Check). With DevOps, the analyzed knowledge can be quickly integrated (Act) for the next development cycle to improve the widget and its delivery. Thus, in DevOps, time to knowledge is shortened and continuously integrated into development for continuous improvement of the delivered output.

Formal concepts like Shewhart's Cycle, PDSA, and PDCA provide the structure and methodology for problem solving/decision making within DevOps. The ability to solve problems or make decisions sooner becomes a competitive advantage for the business or institution employing DevOps. Knowledge becomes a commodity for continual improvement and the ability to capture, retain, share, or expend it dictates significant business decisions, strategies, and future outcomes. DevOps ensures knowledge is appropriately cycled to deliver efficient, quality assured solutions by following a phased approach, based on Shewhart's Cycle, PDSA, and PDCA.

In order to benchmark the knowledge captured through processes like Shewhart's Cycle, PDSA, and PDCA regarding quality improvement, a data driven approach is applied through Six Sigma. Six Sigma was founded by Motorola to combat quality failures and reduce error rates through the combination of Statistical Process Control and Total Quality Management (Tennant, 2001). Six Sigma methodology provides DevOps the knowledge for benchmarking current state of quality in order to strive toward necessary incremental improvement of assured output.

Define, Measure, Analyze, Improve, and Control—otherwise known as DMAIC (pronounced "Duh-MAY-ick") are the five phases of Six Sigma improvement (George et al., 2005). The five phases of Six Sigma improvement guide DevOps to provide quality assured solutions as follows: 1) setting an initial quality level for desired output (Define); 2) concretize initial output (Measure); 3) review knowledge from quality and error levels (Analyze); 4) integrate and develop corrections to known issues for new output (Improve); finally, 5) ensure subsequent delivery cycle includes only improved output (Control). This data driven improvement cycle continuously repeats for DevOps through knowledge capture and feedback loop cycles. The knowledge collected from employing Six Sigma in DevOps provides the necessary process to continually improve the quality level and reduce/eliminate known errors on the next continuous delivery cycle of efficient, quality assured solutions. In order to continually reduce errors or other issues causing constraints, DevOps continually seeks truth regarding existing value-add and non-value-add items for the continuous improvement and continuous delivery of efficient, quality assured solutions.

3.2 Ontology

The lexical representation of ontology from Merriam Webster's definition follows, "a branch of metaphysics concerned with the nature and relations of being" (Ontology, n.d.). DevOps has existed in name since 2009, thanks to Patrick Debois (Mezak, 2018). However, the concept/practice of DevOps has a notable familiarity due to its evolution from the harmonious collection of vetted methodologies, practices, frameworks, and concepts. The next section explores the ontological nature of DevOps through the sub-domain: Truth.

3.2.1 Truth

The preceding section concerning Epistemology and its subcomponents, Belief and Knowledge, lays the groundwork for the ontological Truth of DevOps. The following discussion of Truth is not meant as an exhaustive list of all truths within DevOps, but is a discussion of what, at the highest level, embodies Truth for DevOps. Truth is "the body of real things, events, and facts: actuality" according to Merriam Webster's dictionary (Truth, n.d.).

A genuine truth regarding DevOps is—Lean. The "Lean" in DevOps is respectfully analogous to the "lean production" methodology coined by John Krafcik to reflect the Toyota Production System (TPS) of Toyota Motor Corporation (Toyota); which, intentionally seeks to continuously reduce or eliminate extraneous items and inefficiencies from a production structure that is flexible and less oppressive compared to its predecessor, mass production (Womack et al., 2007).

TPS derived inspiration from both the Ford Production System, a mass production model championed by Henry Ford and workflows of American supermarkets. From this beginning, TPS became a continually improving, just-in-time, Kanban pull system under Taiichi Ōhno of Toyota (Ōhno, 1988). Additional enhancement to TPS came from adoption of Shigeo Shingo's Single-Minute Exchange of Die, also known as SMED (Shingo, 1985).

Lean production has been tested and validated as a truth through Toyota's success and pervasive adoption of TPS across varying business domains. The lean production mindset became associated as a way of thinking—lean thinking. Lean thinking is truth: either an activity is lean or still contains non-essential waste, known as *muda* in Japanese, inhibiting the true value to be realized (Womack & Jones, 1996). The notion to make something "Lean" could be considered as rendering it to be precisely composed of only the exact items needed for it to be efficiently optimal; any additional components are considered excessive, non-value-add and cause value to diminish or become suboptimal.

The output produced from employing DevOps starts with lean thinking for frugal feature standardization through the Minimum-Viable-Product (MVP) concept (Ries, 2011). Frugality should not be corelated with shoddiness but should instead be understood in the context of being sensible, prudent, and economical with regard to resources as well as expectations. A sacrifice to quality should not occur to the MVP in DevOps. Instead, a phased approach to deliver efficient, quality assured features of the MVP should take place through the lifecycle of CI/CD (Continuous Improvement and Continuous Delivery via

Continuous Integration and Development). Additionally, DevOps incrementally improves the initial MVP through continuous knowledge capture and plowback of said knowledge into the next MVP development cycle. DevOps repeats the improvement cycle toward "perfection" or "near-perfection" as the ultimate end goal. Note however, that what constitutes perfection will be determined by the needs of the business/institution.

Efficiency is a vital mantra for DevOps and its success and stems from thinking lean for all aspects of the practice. Thinking lean and making solutions lean, while employing DevOps, aligns well with the familiar adage attributed to Einstein (2019) "Everything should be made as simple as possible, but not simpler" (p. 475). From lean programming code, to lean CI/CD pipelines, the ecosystem of DevOps should consist only of what is exactly needed and nothing more. Additional *muda* will bloat and overload the DevOps practice causing cumbersome maintenance and operations, which is not ideal. Lean thinking for DevOps is a truth and many features that encompass the practice, such as software engineering and computing, employ lean thinking.

Lean thinking is natural to software engineering and computing as both come from the field of mathematics. In mathematics, algorithms should be reduced (reduction) down to the simplest form (Boyer & Merzbach, 1991). Any additional items not pertaining to desired functionality of the algorithm are out of scope and need not be present. Software engineering lean code is recognizable from the desire to develop clean code—simple, direct code (Martin, 2008); additionally, to quote Hoare (1969), "Computer programming is an exact science" (p.576). Lean computing environments are desirable, as is recognized in the collective Unix philosophy of making each program do one thing well by eliminating or not adding unnecessary clutter and complication (McIlroy et al., 1978). Additional lean computing environments include minimalistic operating systems that contain only specific elements essential to desired functionality with capability to modify elements just-in-time if ever needed (Geer, 2009).

Lean is truth for DevOps and enables the practice to continually seek improvement and optimization through reduction of *muda*. Lean thinking is vitally important to the efficiency and success of DevOps. DevOps fundamentally believes an efficient, quality assured solution should be lean in order to reduce/shorten the feedback loop within ideation, development, production, and output. Combining Lean, Agile, and Quality Assurance provides a concise view of principles guiding DevOps philosophy for efficient, quality assured solutions.

4 SCIENCE

According to Oxford University Press, the lexical definition of science is, "the intellectual and practical activity encompassing the systematic study of the structure and behavior of the physical and natural world through observation and experiment" (Science, n.d.). The following section is a brief scholastic account of the scientific nature innate to DevOps through examination of associative branches within the Science domain: Social and Applied.

4.1 Social

Social Science is "the scientific study of human society and social relationships" according to Oxford (Social Science, n.d.). The following section conveys three significant disciplines of Social Science that contribute to the scientific nature of DevOps: Sociology, Psychology, and Economics.

4.1.1 Sociology & Psychology

According to Giddings (1923), "Sociology is an attempt to account for the origin, growth, structure, and activities of society by the operation of physical, vital, and physical causes, working together in process of evolution" (p.8); and according to Arnoult (1976), "Psychology is the study of behavior; that is, the study of observable actions of humans and other animals" (p.7). Although Sociology and Psychology are distinctive fields, they will be examined together to better understand the adoption of DevOps by an organization. The following section explores DevOps culture and transformation, specifically as applied to business organizations that adopt the practice. Many of the points analyzed, however, are applicable for other types of organizations such as federal and academic institutions.

Businesses are social systems-of-systems (SoS) organizations that endeavor to effectively motivate autonomous, free-willed humans to achieve common behavior and organizational goals (Hersey & Blanchard, 1977). For the varying systems in a business to embrace DevOps, a significant commitment and effort is required by the leaders and workforce. Both the individual worker and collective workforce need to culturally adopt such

DevOps principles as continuous improvement, which compels a continuous learning environment.

DevOps adoption transforms the culture of a business into thinking and behaving within a new paradigm, both sociologically and psychologically. When a business embraces a new paradigm, new capabilities emerge and current capabilities need to be assessed for continuance. Capabilities under the previous business paradigm may not align with DevOps and therefore may need to be eliminated or transformed into new capabilities to prevent them from becoming rigidities (Leonard, 1992).

Businesses should commit to adapting or reinventing as needed to meet requirements for a successful transformation to DevOps. Executive leadership should provide consistent, proactive communication to the workforce regarding its DevOps transformation progress in a transparent manner to eliminate barriers and ensure success. Reinventing the business philosophy, learning new competencies, and eliminating transformation barriers enable adoption of DevOps principles and practices. The commitment to transform the business must be steadfast from leadership and effectively executed for the workforce or it will fail (Deming, 2000).

When transitioning to DevOps, new capabilities most assuredly require new skills. The business must commit to maturing the workforce with necessary skills for DevOps, like the T-Shape Professional, to provide significant psychological value to the individual worker and sociological value to the workforce. The ability to provide value to the individual worker strengthens the organizational commitment between the individual and the business (Margelytė-Pleskienė & Veinhardt, 2018), This, in turn, strengthens a successful cultural transformation to DevOps. If new skills are not fostered, individuals may become disenfranchised and seek alternative businesses to continue career development.

Businesses using technology must recognize the need to adopt DevOps or be left with organizational obsolescence (Le Mens et al., 2015) and organizational cynicism (Wanous et al., 1994). Additionally, businesses must guard against resistance to DevOps due to hubristic leadership (Sadler-Smith et al., 2016), collective hubris (Sullivan & Hollway, 2014), and success syndrome (Tushman & O'Reilly, 1996), all of which could stifle or prevent adoption and transformation to DevOps.

Transformation for a business is not trivial. If a business can initialize and communicate the value of the transformation to the workforce, barriers to transformation become easier to remove. A reference model for successful cultural transformation is found from Toyota's "Toyota Way." Toyota's culture was transformed into "Toyota Way" by adopting TPS and positively embracing its workforce to foster sociological and psychological value (Liker & Franz, 2011). Toyota describes "Toyota Way" as rooted in the spirit of making things (*monozukuri* in Japanese) (Toyota Motor Corporation, n.d.) as well as two principles: continual improvement (*kaizen* in Japanese) and respect for people (Liker, 2004). The principles of "Toyota Way" transformed Toyota's culture into a positive, learning environment, which strengthened organizational commitment from its workforce (Allen & Meyer, 1990) and significantly contributed to longstanding industry success (Sobek II & Smalley, 2008).

The DevOps culture is analogous to "Toyota Way" regarding positive, impactful business and culture value. DevOps provides a unique organizational culture that practices continual delivery of efficient, quality assured solutions through continual improvement methodologies based on science. DevOps encourages a scientific environment for the collective and individual to organize, research, develop, and assess solutions based on repeatable information (Wallace, 1971).

4.1.2 Economics

According to Marshall (1890), "Economics is on one side a study of wealth and on the other a branch of the study of man" (p.1). Economics has two main branches: macroeconomics and microeconomics. Macroeconomics studies the global economy while microeconomics studies the economies of individual persons and businesses. For scope and brevity, this section describes how DevOps enhances the wealth of individual businesses through an important feature of reducing the time required for an efficient, quality assured output (e.g. product/service) to enter the market—coined "Faster-Time-to-Market." Although this section focuses on DevOps for businesses, it would be an oversite not to recognize the value Faster-Time-to-Market has for research institutes such as universities and federal laboratories, although, applied as such, the phrase warrants slight modification to the more suitable—Faster-Time-to-Science (NASA, 2013).

Functional application of DevOps provides business operations increased acceleration of quality assured output to the market. Increased acceleration of production from DevOps could be viewed as faster time to production, which enables a Faster-Time-to-Market. The concept of Faster-Time-to-Market originates from "Time to Market" in correlation to the reduction of product lead time for an output to reach the consumer market (Charney, 1991).

DevOps provides significant impact to businesses universally through its general employment of key concepts such as Agile, Lean, Quality Assurance, Cultural Transformation, and Automation. Incorporation of said concepts provides value-add to managing and improving business processes with the ultimate aim of reducing costs while maintaining, or even increasing, customer satisfaction. Incorporation of DevOps for businesses improves value chains through reduced costs and differentiation, which could provide competitive advantage within a tightly vying industry (Porter, 1985). Additionally, DevOps enhances the ability of an organization to provide quality assured output at an accelerated frequency for the market (Faster-Time-to-Market), which thereby increases opportunity to satisfy customers. The ability for a company to provide customer satisfaction sooner and more frequently than the competitor is a competitive advantage and bastion against competitive forces within its industry (Porter, 1997).

Increasing the frequency of customer satisfaction with delivered output is a critical success factor for any business due to inbound cashflow from consumer spending and consumer retention. Therefore, businesses may spend significant resources to obtain customer satisfaction and retention through quality of conformance, which focuses on the identification and elimination of defects before market release to ensure output meets, or exceeds, intended design specifications (Garrison et al., 2017). DevOps enhances the ability for a business to obtain quality of conformance through its efficient, quality assured methodologies regarding continual improvement and delivery. The sooner quality assured output can reach the consumer market ahead of competitors, the sooner an opportunity is presented to make a sale from consumer spending (i.e. inbound cashflow). If a business has known inbound cashflow, that monetary resource will be worth more at present time than in the future due to potential uncertainties and failing inadequacies: internally with people, processes, and technologies; or externally to macroeconomic and geo-political uncertainties (BIS, 2011). Also, time is money; the Time-Value of Money principle regards a unit of money today as worth more than the same unit of money in the future (Kimmel et al., 2018). Ergo, realization and purchasing power of timelier inbound cashflow allows the business to make monetary decisions sooner; a critical ability in competitive markets. If a business obtains competitive and comparative advantage over competitors, those advantages become invaluable to the business and are exploited to lead the industry.

While macroeconomics and geo-political risks are beyond the control of businesses, operational risk is not. Operational risk includes technology systems and technology risk, both significant to the practice and continuation of DevOps. Since the purchase and use of technology may not provide value-add, businesses meticulously strategize and cost manage how technology can provide value-add through operational efficiency improvements in the form of economies of scale or economies of scope (Saunders & Cornett, 2017). DevOps provides value-add through increased speed of output (Faster-Time-to-Market) to obtain economies of scale and additionally enhances ability to meet output need, fulfilling economies of scope. DevOps looks to help provide a methodology for businesses to get-to-market sooner (Faster-Time-to-Market), which in turn provides a valid return-on-investment regarding the technology required to engineer and implement DevOps.

Thus, DevOps can complement and further enhance a paradigm of business success already understood through the synergy of total quality management and operational risk management (Luburić, 2012). DevOps provides continual improvement and continual delivery of solutions (output) while benchmarking the current state of quality to strive toward incremental improvement for efficient, quality assured solutions.

4.2 Applied

According to Smith (1967), "[A]pplied science can be regarded as a body of experimental methods, experimentally determined constants and theory" … "that are likely to prove helpful to engineers and others who have to solve socially determined problems by the use of any available methods and concepts regardless of their intrinsic intellectual interest or lack of it" (p.57). The following section conveys the significant discipline of Applied Science that contributes to the scientific nature of DevOps: Engineering.

4.2.1 Engineering

Engineering is, according to Engineering World (1919), "the practice of safe and economic application of the scientific laws governing the forces and materials of nature by means of organization, design and construction, for the general benefit of mankind" (p.34).

As mentioned earlier in this writing, DevOps originated from software engineering, likewise significant nomenclature associated throughout the practice originates from the respective engineering field. Software engineering encompasses knowledge in areas such as: computer programing, architecture (e.g. data, software, system), solution engineering; as well as various disciplines, such as: computer science, mathematics, quality management, systems engineering (Bourque et al., 2014). DevOps engineering methodologies have greatly enabled efficiency and success by holding to sound mathematical and engineering principles: Abstraction (Russell, 1903), Modularity (Gauthier & Pont, 1970), Polymorphism (Cardelli & Wegner, 1985), Portability (Newey et al., 1972), Encapsulation (Parnas, 1972), Reusability (Rich & Waters, 1983).

Engineered automation is an essential aspect of DevOps and is a key component to its success. Additionally, automation is essential to modern and future computing technologies. In 2001, IBM proclaimed autonomic computing a necessity and provided principles on the ability to self-manage through four concepts: self-configuration, self-optimization, self-healing, and self-protection (Kephart & Chess, 2003). Then in 2005, IBM provided an architectural blueprint for autonomic computing (IBM, 2005). John von Neumann set precedence in autonomic computing with formulated lectures on automata computing machines compared to the human nervous system (von Neumann, 1951) and later the notion of self-reproducing automata (von Neumann, 1966). A significant contribution to the success of DevOps resides in the practice of engineering CI/CD pipelines to automate configuration, optimization, healing, and securing of efficient, quality assured, engineered solutions.

Implementation of DevOps CI/CD pipelines leverage sophisticated technologies to enhance and promote automation of disparate systems, which require nontrivial, customization for interoperability, however developed software has eased implementation (Klein et al., 2020). Software and system operation teams, or full-stack engineering teams, coordinate interfaces to develop, manage, and operate systems used to establish desired pipelines. Nonetheless, integration of DevOps CI/CD pipelines need to be engineered to enable expressive automation, thus engineering automation to enable automation engineering (i.e. build the machine that builds the machines).

The engineered automation in DevOps, such as, but not limited to CI/CD pipelines, are composed of synergistic, reusable, polymorphic building blocks: Images, Scripts, Application Programming Interfaces (APIs) (Klein & Miner, 2018). The engineered building blocks in DevOps are analogous to abstract data types in computer science and Functionals (higher-order function) in mathematics. The building blocks are untyped in the utmost abstract form when designing generic, reusable architecture patterns and solutions, then typed when instantiated to develop efficient, quality assured solutions. The combination of the engineering building blocks in DevOps provides increased expressiveness to architect and engineer various reusable patterns and solutions, ranging from trivial to complex. Since the engineering building blocks can be software defined, advanced development with automation and computing has taken place. Entire systems, and SoS, have evolved to become integrated with software defined solutions (Juve & Deelman, 2011). Fully software defined systems and SoS have been deployed unto various physical and digital environments for various production workload purposes (Gannon et al., 2017). Even atypical propositions (Miner & Klein, 2018) for physical modern systems and SoS to become software defined have initialized (Miner et al., 2018).

Advanced efficient, quality assured computing solutions exist in DevOps due to expressive interplay between the engineering building blocks. These engineering building blocks enhance the use of entirely developed software defined systems, and/or SoS, solutions, which provide a gamut of advantageous and tactical architectural patterns to quickly employ such as: save-point, ephemeralness, immutability, and idempotency. Similar in notion to a save-point for video games or databases, software defined solutions can expressively save-point entire ecosystems (as images, scripts, APIs). Once the ecosystem is saved, CI/CD pipelines can redeploy solutions to resurrect or roll-back to desired state; and/or continue existence in a different ecosystem for survival or compliance (Hadgu et al., 2016). Ephemeral solutions live only for the moment in time they are specifically needed, then shut down until needed again or terminate from existence through CI/CD pipelines. Ephemeral, software defined computing environments have increased in use through computing (Cotta et al., 2016). Immutable solutions are designed to have nonchanging and persistent state. However, if an undesired permutation occurs in the solution, then the permutated solution is destroyed and re-instantiated to the originally intended desired state through CI/CD pipelines. Such implementation of immutability into entire solutions includes immutable infrastructure (Mikkelsen et al., 2019). Idempotent solutions are designed to consistently provide nonchanging, persistent existence and behavior regardless of the number of times deployed or applied through CI/CD pipelines. Such idempotent software defined computing solutions include distributed cloud environments (Ramalingam & Vaswani, 2013).

The recognition and significance of DevOps automation is pervasive in modern technology, solution architecting, and engineering in advanced computing fields such as Cloud Computing, High-Performance Computing, Big Data, Machine Learning/Artificial Intelligence, Internet of Things, Robotics, and Extended Reality. When the automation of development and deployment of software-based solutions are combined, increased expressiveness and DevOps automation success are realized. Through the combination of DevOps CI/CD pipelines and engineering building blocks, the ability to develop efficient, quality assured solutions through engineered automation becomes achievable and maintainable.

5 CONCLUSION

DevOps is a multifaceted, eclectic concept and practice with origins in academic and industry recognized methodologies and practices. The philosophy and science behind DevOps are the reasons for its fast adoption and success. This writing covered DevOps with regard to the domain of Philosophy through the branches of Epistemology and Ontology, under the classifications of Beliefs, Knowledge, and Truths. Additionally, the discussion explicated DevOps in light of the domain of Science, as subdomains of Social and Applied were examined through the fields: Sociology, Psychology, Economics, and Engineering. Accommodating instantiations of the subdomains for both Philosophy and Science were further examined through key DevOps concepts and practices and applied contextually to a business framework.

6 ACKNOWLEDGEMENTS

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2021-6250 R

7 REFERENCES

Allen, N. J., & Meyer, J. P. (1990). “Organizational socialization tactics: a longitudinal analysis of links to newcomers’ commitment and role orientation”, *The Academy of Management Journal*, 33(4), 847-858.

Arnoult, M. D. (1976). *Fundamentals of Scientific Method in Psychology*. Dubuque, Iowa. Wm. C. Brown Company Publishers.

Bank for International Settlements – BIS. (2011). *Principles for the Sound Management of Operational Risk. Basel Committee on Banking Supervision*. June 2011. Basel: Bank for International Settlements.

Beck, K. M., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *The Agile Manifesto*. http://agilemanifesto.org/

Belief. (n.d.). Cambridge Dictionary. In Cambridge.com dictionary. Retrieved from https://dictionary.cambridge.org/us/dictionary/english/belief

Bourque, P., Fairley, R. E., & IEEE Computer Society (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK®)* (3rd Edition). IEEE Computer Society Press, Washington, DC, USA.

Boyer, C. B., & Merzbach, U. C. (1991). *A history of mathematics*. New York, N.Y: J. Wiley and Sons. 229.

Cardelli, L., & Wegner, P. (1985). On Understanding Types, Data Abstractions, and Polymorphism. *Computing Surveys*, 17(4), 471-522.

Charney, C. (1991). *Time to Market: Reducing Product Lead Time*. Dearborn, MI: Society of Manufacturing Engineers.

Cotta, C., Fernández-Leiva, A., Vega, F., Chávez, F., Merelo Guervós, J., Castillo, P., Camacho, D., R-Moreno, M. (2016). *Application Areas of Ephemeral Computing: A Survey*. 10.1007/978-3-662-53525-7_9.

Deming, W. E. (2000). *Out of the Crisis*. United Kingdom: Massachusetts Institute of Technology, Center for Advanced Engineering Study.

Deming, W. E. (2018). *The New Economics for Industry, Government, Education*. United States: MIT Press.

DevOps. (2020, February 4). In *Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/wiki/DevOps

DevOps Institute. (2019). *The DevOps Engineer Craze– Who You Should Really Hire*. Retrieved from https://devopsinstitute.com/2019/05/30/the-devops-engineer-craze-who-you-should-really-hire/

Einstein, A., & Dyson, F. (2019). *The Ultimate Quotable Einstein*. United States: Princeton University Press.

Engineering World. (1919). *Engineering World: A Journal of Engineering and Construction*. United States: International Trade Press, Incorporated. 14-15.

Epistemology. (n.d.). *Oxford University Press*. In Lexico.com dictionary. Retrieved from https://www.lexico.com/en/definition/epistemology

Gannon, D., Barga, R., & Sundaresan, N. (2017). Cloud-Native Applications. *IEEE Cloud Computing*, 4, 16-21.

Garrison, R. H., Noreen, E. W., & Brewer, P. C. (2017). *Managerial Accounting*. Boston: McGraw-Hill/Irwin.

Gauthier, R., & Pont, S. (1970). *Designing Systems Programs*. Prentice-Hall, Englewood Cliffs, N.J., 1970.

Geer, D. (2009). The OS Faces a Brave New World. *Computer*, 42(10), 15-17. 10.1109/MC.2009.333.

George, M. L., Rowlands, D., Price, M., & Maxey, J. (2005). *The Lean Six Sigma Pocket Toolbook: A Quick Reference Guide to Nearly 100 Tools for Improving Process Quality, Speed, and Complexity*. New York, NY: McGraw-Hill.

Giddings, F. H. (1923). *The Principles of Sociology: An Analysis of the Phenomena of Association and of Social Organization*. United Kingdom: Macmillan.

Hadgu, T., Klein, B. T., Appel, G. J., & Miner, J. G. (2017). *Cloud Computing for Complex Performance Codes*. United States. 10.2172/1343253.

Hersey, P., & Blanchard, K. H. (1977). *Management of organizational behavior: utilizing human resources* (3rd Edition). Englewood Cliffs, N.J.: Prentice-Hall.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communication of the ACM*, 12(10), 576-580, 583.

IBM Corporation. (2005). An Architectural Blueprint for Autonomic Computing. *Autonomic Computing White Paper*.

Imai, M. (1986). Kaizen: *The Key to Japan's Competitive Success*. McGraw-Hill Education, New York.

Juve, G., & Deelman, E. (2011). Automating Application Deployment in Infrastructure Clouds. *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, Athens, 658-665.

Kephart, J., & Chess, D. (2003). *The Vision of Autonomic Computing*. IEEE.

Kimmel, P. D., Weygandt, J. J., & Kieso, D. E. (2018). *Financial Accounting: tools for business decision making* (9th Edition). New York: Wiley.

Klein, B. T., & Miner, J. G. (2018). *DevOps: Images Scripts APIs Oh my!*. United States. https://www.osti.gov/servlets/purl/1512856

Klein, B. T., Giese, G., Lane, J., Miner, J. G., Jones, J. J., & Venezuela, O. (2020). *An Approach to DevOps and Microservices.* United States. 10.2172/1635752.

Knapp, J., Zeratsky, J., & Kowitz, B. (2016). *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. New York, NY: Simon & Schuster.

Laskowski, J. (2011). *Why software quality assurance and IT security need to work together*. IBM Corporation. Retrieved from https://www.ibm.com/developerworks/rational/library/software-quality-assurance-IT-security/software-quality-assurance-IT-security-pdf.pdf

Le Mens, G., Hannan, M., & Pólos, L. (2015). Organizational Obsolescence, Drifting Tastes, and Age Dependence in Organizational Life Chances. *Organization Science*, 26, 550-570.

Leonard, D. (1992). Core Capability and Core Rigidities: A Paradox in Managing New Product Development. *Strategic Management Journal.* 13. 111-125.

Liker, J. K. (2004). *The Toyota way: 14 management principles from the world's greatest manufacturer*. New York: McGraw-Hill.

Liker, J. K., & Franz, J. K. (2011). *The Toyota Way to Continuous Improvement*. New York: Mc-Graw Hill.

Luburić, R. (2012). Synergistic effects of Total Quality Management and Operational Risk Management in central banks. International Journal for Quality Research 6(4), 381-388.

Margelytė-Pleskienė, A., & Vveinhardt, J. (2018). The Quintessence of Organizational Commitment and Organizational Cynicism. *Management of Organizations: Systematic Research,* 80, 67-88.

Marshall, A. (1890). *Principles of Economics*. Book 1. Macmillan and Company.

Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. United States: Pearson Education.

McIlroy, M., Pinson, E. N., & Tague, B. A. (1978). *UNIX time-sharing system: Foreword*. The Bell System Technical Journal, 57, 1899-1904.

Mezak, S. (2018). *The Origins of DevOps: What's in a Name?*. DevOps. https://devops.com/the-origins-of-devops-whats-in-a-name/

Mikkelsen, A., Grønli, T., & Kazman, R. (2019). *Immutable Infrastructure Calls for Immutable Architecture*. 10.24251/HICSS.2019.846.

Miner, J. G., & Klein, B. T. (2018). *FFRDC-in-a-Bottle.* Sandia National Laboratories. United States. https://www.osti.gov/servlets/purl/1513087

Miner, J. G., Klein, B. T., Venezuela, O., & Comen, A. (2018). *Federally Funded Research and Development Centers-as-Code* (Version 1). [Computer software]. https://www.osti.gov//servlets/purl/1439578.

NASA. (2013, November 13). *NASA brings Earth Science 'big data' to the cloud with Amazon web services.* Phys. https://phys.org/news/2013-11-nasa-earth-science-big-cloud.amp

Newey, M. C., Poole, P. C., & Waite, W. M. (1972). Abstract machine modeling to produce portable software—a review and evaluation. *Software: Practice and Experience*, 2(2)*,* 107-136.

Ōhno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. United Kingdom: Taylor & Francis.

Ontology. (n.d.). *Merriam-Webster*. In Merriam-Webster.com dictionary. Retrieved from https://www.merriam-webster.com/dictionary/ontology

Parnas, D. L. (1972). On the Criteria to Be Used in Decomposing Systems into Modules. *Communication of the ACM*, 15(12), 1053-1058.

Philosophy. (n.d.). *Cambridge Dictionary*. In Cambridge Dictionary.com dictionary. Retrieved from https://dictionary.cambridge.org/us/dictionary/english/philosophy

Porter, M. E. (1985). *Competitive advantage: Creating and sustaining superior performance*. New York: Free Press.

Porter, M. E. (1997). How competitive forces shape strategy (HBS reprint 79208). *Competitive Strategy*. 1-10.

Ramalingam, G., & Vaswani, K. (2013). Fault Tolerance via Idempotence. *ACM SIGPLAN Notices*, 48, 249-262. 10.1145/2480359.2429100.

Rich, C., & Waters, R. (1983). *Formalizing Reusable Software Components*. The Artificial Intelligence Laboratory Massachusetts Institute of Technology.

Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. United Kingdom: Crown Business.

Russell, B. (1903). *The Principles of Mathematics* (Volume 1). Cambridge: University Press.

Sadler-Smith, E., Akstinaite, V., Robinson, G., & Wray, T. (2016). Hubristic leadership: A review. *Leadership*, 13. 10.1177/1742715016680666.

Sajjad, F. (2020). *Implementing DevOps Goes Beyond Technology*. DevOps. https://devops.com/implementing-devops-goes-beyond-technology/

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2009). *Systems Analysis and Design in a Changing World*. Boston, MA: Thomson Course Technology.

Saunders, A., & Cornett, M. M. (2017). *Financial Institutions Management: A Risk Management Approach*. 9th ed. New York: McGraw-Hill.

Science. (n.d.). *Oxford University Press*. In Lexico.com dictionary. Retrieved from https://www.lexico.com/en/definition/science

Shewhart, W. A. (1939). *Statistical Method from the Viewpoint of Quality Control*. Department of Agriculture. Dover, 1986.

Shingo, S. (1985). *A revolution in manufacturing: the SMED system*. Japan: Taylor & Francis.

Smith, C. (1967). A Historical View of One Area of Applied Science—Metallurgy. *National Research Council Panel on Applied Science Technological Progress. Applied Science and Technology Progress: A report to the Committee on Science and Astronautics, U.S. House of Representatives*. Washington, D.C.: National Academy of Sciences. 57-71.

Sobek II, D. K., & Smalley, A. (2008). *Understanding A3 Thinking: A Critical Component of Toyota's PDCA Management System*. Productivity Press: Boca Raton, FL.

Social Science. (n.d.). *Oxford University Press*. In Lexico.com dictionary. Retrieved from https://www.lexico.com/en/definition/social_science

Sullivan, G., & Hollway, J. (2014). Collective pride and collective hubris in organizations. In, *Understanding Collective Pride and Group Identity: New Directions in Emotion Theory, Research and Practice*. United Kingdom: Taylor & Francis. 10.4324/9781315767680-7.

Tennant, G. (2001). *Six Sigma: SPC and TQM in manufacturing and services*. United Kingdom: Gower.

Toyota Motor Corporation. (n.d.). Toyota Production System. Toyota. https://global.toyota/en/company/vision-and-philosophy/production-system/

Truth. (n.d.). *Merriam-Webster*. In Merriam-Webster.com dictionary. Retrieved from https://www.merriam-webster.com/dictionary/truth

Tushman, M., & O'Reilly, C. A. (1996). Ambidextrous Organizations: Managing Evolutionary and Revolutionary Change. *California Management Review*, 38(4), 8-30.

Vogels, W. (2017). *Amazon Web Services 2017 re:Invent Keynote Presentation*. [Video]. YouTube. https://www.youtube.com/watch?v=nFKVzEAm-ts

von Neumann, J. (1951). The general and logical theory of automata. In L.A. Jeffress (Ed.), Cerebral mechanisms in behavior; the Hixon Symposium, 1-41. Wiley.

von Neumann, J., In Burks, A. W., & Goldstine, H. H. (1966). *Theory of self-reproducing automata*. Urbana: University of Illinois Press.

Wallace, W. L. (1971). *The Logic of Science in Sociology*. Chicago, IL: Aldine Atherton, Inc.

Wanous, J. P., Reichers, A. E., & Austin, J. T. (1994). *Organizational cynicism: An initial study*. Columbus, Ohio: Max M. Fisher College of Business, Ohio State University.

Womack, J. P., & Jones, D. T. (1996). *Lean thinking: Banish waste and create wealth in your corporation*. New York: Free Press.

Womack, J. P., Jones, D. T., & Roos, D. (2007). *The Machine that Changed the World*. United Kingdom: Simon & Schuster.