Agile Big Data Analytics

AnalyticsOps for Data Science

Nancy W. Grady, Jason A. Payne,
Advanced Analytics
SAIC
McLean, VA, USA
nancy.w.grady@saic.com, jason.a.payne@saic.com

Huntley Parker
Software Integration
SAIC
McLean, VA, USA
alson.h.parker.iv@saic.com

Abstract—**Big data analytic (BDA) systems leverage data distribution and parallel processing across a cluster of resources. This introduces a number of new challenges specifically for analytics. The analytics portion of the complete lifecycle has typically followed a waterfall process – completing one step before beginning the next. While efforts have been made to map different types of analytics to an agile methodology, the steps are often described as breaking activities into smaller tasks while the overall process is still consistent with step-by-step waterfall. BDA changes a number of the activities in the analytics lifecycle, as well as their ordering. The goal of agile analytics—to reach a point of optimality between generating value from data and the time spent getting there. This paper discusses the implications of an agile process for BDA in cleansing, transformation, and analytics.**

Keywords— advanced analytics; agile development; analytics lifecycle; AnalyticsOps; big data analytics; data science; data science process models; Deep Learning; DevOps; Knowledge Discovery in Data Science, machine learning

I. Introduction

A general model for data science is a lifecycle of data collection, curation, analysis, and action to achieve mission goals. Depending on the data under observation, the desired goal, and metrics for success, the amount of time allocated across these steps will vary. Advanced analytics and machine learning are fast-growing areas of practice and research, so there is a need to develop a modernized, standard process that addresses current challenges in machine learning application development and deployment cycles – and encompasses changing techniques and technologies.

Much of the thinking about data science has been driven by large internet companies, which are fundamentally data-driven. The focus has been to “fail fast”, that is, to learn quickly if an analytics direction has promise. The goal is a business outcome rather than improving a measure of accuracy on a specific analytic model. In some cases, a surrogate hypothesis is developed that can lead to the same business outcome by inference from an entirely different hypothesis. This process is typically ad hoc for a data science team and strongly depends on the skills of the leading data scientist. While the specifics of the analytics are discussed, the process is not standardized to allow repeatability across the broader community.

Typically, data science is discussed only in the context of the pre-analytics cleansing and transformations and in the analytics step – relying on a traditional data analytics lifecycle. Operational Big Data Analytic (BDA) systems, introduce a number of complexities due to the distribution of the data, the novel data storage techniques, and the involvement of multiple organizations when for example using cloud services.

Software development lifecycles (SDLC) have changed dramatically with the introduction of agile methodologies, to reduce development times, to provide earlier value, and to lower the risk of failure. The data science lifecycle for BDA systems is currently in the same situation as SDLC-driven development prior to the introduction of agile methods.

We present in this paper our approach to a modern process model for BDA that aligns with new technological changes and implements agility in the lifecycle of advanced analytics and machine learning systems development – to minimize the time required to reach a desirable mission outcome.

II. Big Data Analytics Challenges

A. *Limitations of Prior Process Model*

Traditionally in statistical analysis, data were carefully collected to be necessary and sufficient to definitively answer a specific question, for example, in the pharmaceutical industry to pass the Food and Drug Administration regulatory requirements for releasing a drug. In the late 1990s, analytics moved beyond the realm of traditional statistics into what became known as data mining. At that time, the data mining community began to use a range of models on repurposee data—to analyze data in a new context rather than the one for which the data were collected. These new mathematical models were used to provide a probabilistically accurate answer rather than a deterministic answer. As the data mining community grew, comparing and contrasting approaches was difficult, because the activities in the end-to-end process were described in various numbers and types of steps. The solution developed by a consortium is the Cross-Industry Standard Process Model–Data Mining (CRISP-DM) [1]. CRISP-DM remains the dominant process model still in use today [2] in spite of the recognition of a number of problems [3].

Data science as a term grew out of the advances in big data engineering with the recognition that large-volume or high-velocity datasets required new parallelization techniques to be able to handle the large amounts of data efficiently [4]. In addition, often leveraging correlation was sufficient rather than having to understand causation. For example, in website A and B testing, if more customers click on a blue link than a green one, the perceptual reasons do not need to be known; the correlation itself is actionable for optimizing the design of the web page.

Big data represents data distribution and parallel processing, so the data storage, algorithms and analytics lifecycle are no longer separable from the technologies of big data. While still the de-facto standard, CRISP-DM does not address these changes, as well as automation, data science, or systems development methodologies.

B. *Outcomes-Focused Requirements*

Analytics systems development is driven by detailed requirements for the construction of each capability in the system. Most advanced analytics systems development is driven by desired outcomes rather than specific requirements. Categories of analytics can be described as a ladder of increasing complexity, as shown in Fig. 1. Reporting and business intelligence systems can be described by explicit requirements. In contrast, more sophisticated analytics require multiple computational experiments and comparative analysis through modeling, machine learning, and simulation to achieve results. Tuning and optimizing models are an iterative exploratory process of experimentation and testing and evaluation. Advanced analytics cannot be specified by a list of detailed model requirements but only by desired outcomes.

C. *Fail Fast*

One of the main tenants of data science as it emerged as a new discipline for the large-scale internet applications is the need to fail fast. The idea is that given the large amounts of data, rough order of magnitude estimations should be done first. Techniques or approaches that show promise should continue to be investigated, while others should be stopped. Conceptually, the analytics should go through a set of iterations of ever-greater granularity and fidelity, stopping when the results are sufficient to meet the outcome requirements. CRISP-DM was considered essentially a waterfall model—one step was finished before the next began. While evaluation may indicate the need to step back and modify the actions in a prior step, the results were only achieved at the end of the complete lifecycle. Data science needs to follow the lessons learned in software development to address this need for agility.

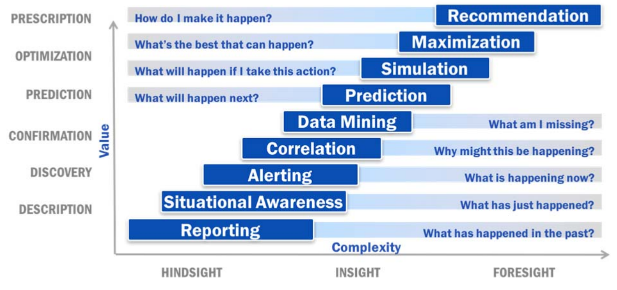


Figure 1. The Analytics Ladder

III. Agile Software Development

During the past 15 years, the use of agile for the software development lifecycle (SDLC) has gradually eclipsed the traditional waterfall model to become the dominant practice across projects and organizations of all sizes. Roughly seventy-five percent of software developers currently report using primarily agile practices, compared to twenty-five percent for waterfall. This trend is being amplified by the addition of complementary concepts from DevOps, which build on the foundation of agile practices. The popularity of the agile software development methodology (often referred to simply as *agile*) and how its principles may be applied to data science are best understood from how it evolved.

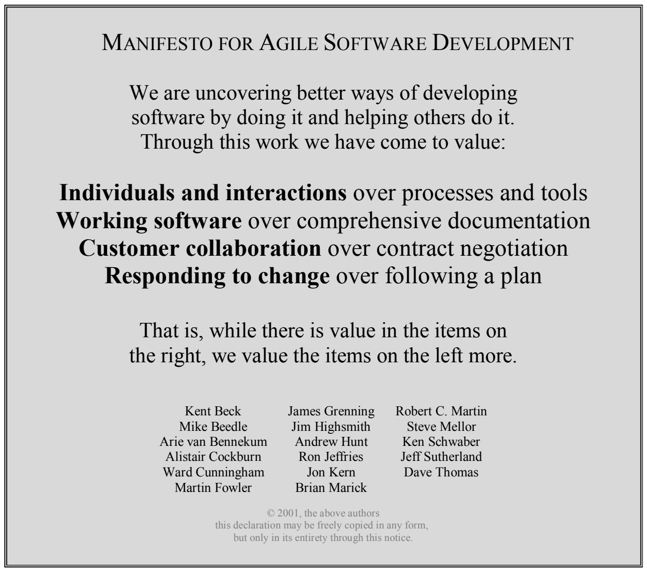
A. *The Waterfall Model*

The traditional waterfall model for software development is a linear, sequential approach—a progression through phases of requirements gathering, design, implementation, verification, and maintenance. The phases are completed in sequential order, and work begins in each phase only when the previous phase is complete. This model had its origins in the industries of manufacturing and construction and was applied to software development in the 1960s primarily because an alternative model for knowledge work did not exist.

The deficiencies of the waterfall model for development of software were recognized relatively early (the paper most frequently cited describing the model was written by Winston W. Royce in 1970, who characterized it as risky and prone to failure [5]. These issues were largely attributable to the fact that a failure in one of the later phases or a change in requirements necessitated returning to one of the early phases and restarting the process. This typically led to one of two outcomes: either the project was significantly delayed and over budget or the initial requirements were delivered but did not provide the anticipated value. Despite these issues (and the fact that alternative, iterative processes were in use throughout this period), the waterfall method persisted as the dominant methodology in software development through at least the early 2000s. Factors that contributed to its longevity include its ease of understanding, implied promise of predictability, perception of management control from stage-gate approvals and extensive documentation, and well-defined roles for various specialists.

B. *The Emergence of Agile*

Despite the widespread use of waterfall methods, many professionals working in software development from the earliest days recognized its deficiencies and proposed alternatives. Most of these alternative approaches relied on a more iterative development process, recognizing that the activities could not be completed in a strictly sequential manner. During the 1990s, several iterative approaches began to gather significant momentum, culminating in the publication of the Agile Manifesto and Principles in 2001 [6], as shown in Fig. 2. Several factors highlighted the inherent deficiencies in the waterfall method and drove the emergence of the agile methodology as a necessary evolution in software development. These can be grouped into three primary categories:


MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

Figure 2: Agile Manifesto

1. Human factors in software development.
2. Changes in cost and risk factors.
3. Understanding the nature of complex systems.

C. Drivers of Agile

There are three main drivers for the move to agile software development.

1) Human factors in software development

Agile focuses on the understanding that software development is knowledge work performed by individuals working together with other individuals in teams. While this may appear obvious, it is not something on which the waterfall method placed much importance. The waterfall process evolved from the practices of mass manufacturing physical products and valued standardization, automation, and repetition. Individuals performed limited functions, and provided they produced a result within certain tolerances, their output could be reliably consumed by the next stage of the process. The end result is a final big-bang delivery at the end, with little change in knowledge until the end, as illustrated by Cockburn [7] in Fig. 3,

Individuals could (with minimal training) be substituted at any particular stage of the overall process with no change to the final product or initial design. This model seemed to apply well enough when computing resources were scarce and software development involved producing large quantities of physical cards to input data for calculation. As computing power increased and developers gained access to their own personal computing resources, the human factors began to have significant impact.

Agile strives to address these issues relating to human factors by placing the focus on individuals and interactions over processes and tools. Small, cross-functional teams of five to nine members allow for a mix of complementary skills and efficient team communication. Teams are empowered to make decisions on the software design and to organize their own work. Product owners and stakeholders (representing the interests of the consumers of the end value) and developers collaborate frequently. Teams are kept stable so that group dynamics are optimized and performance increases over time. Teams deliver work incrementally in time-boxed iterations of 2 to 4 weeks. Each iteration contains elements of analysis, design, coding, testing, and deployment to create an increment of working software. Iterative delivery facilitates frequent feedback from users and the ability to respond to changing requirements, mitigates the risk of mistakes through limited work-in-process and continuous integration and testing.

In modern software development, the primary constraint is the rate of learning of the development team. Agile accelerates learning through its use of frequent integration, iterative releases, and user feedback. Prototyping is relatively fast and inexpensive in software, and early feedback can replace big upfront design with better end results. Sometimes multiple prototypes are employed, using A/B testing to rapidly converge on the best solution. The learning curve from Cockburn in Fig. 4 illustrates the rapid learning by the team early on in the project.

2) Changes in cost and risk factors

If computing resources are scarce and expensive, it makes sense for a development team to spend significant time and effort on extensive requirements gathering and upfront design

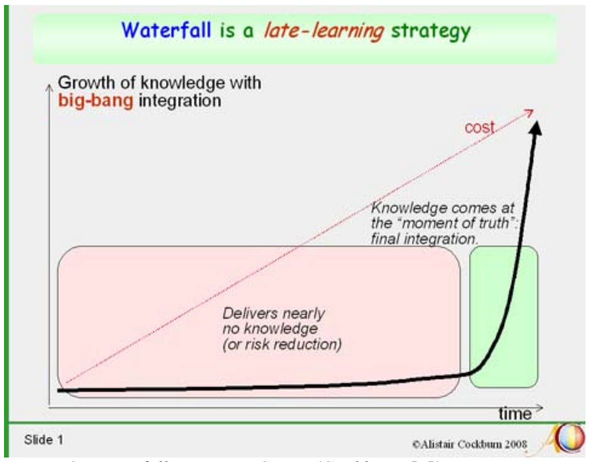


Figure 3. Waterfall Learning Curve (Cockburn [7])

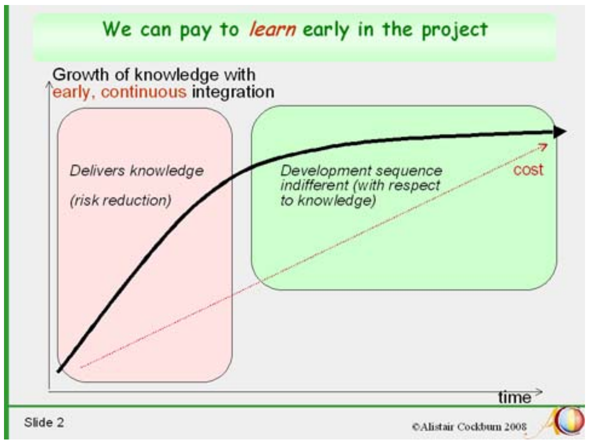


Figure 4. Agile Learning Curve (Cockburn [7])

work. Or, if the business innovation cycle is sufficiently long, perhaps the organization can tolerate the cost of delay when the development project delivers no working software for several years. In either scenario, there may not be any motivation to look for an alternative to the waterfall method. Currently, it is unlikely that either of those scenarios apply in any industry or organization.

When development projects typically spanned many years and systems were expected to operate for decades, the risk of changes to the requirements or design was very high. Currently, the speed with which technology evolves and the reduced expectation in product cycle times means that the cost of delay in releasing software is frequently the highest risk. Because of agile's imperative to deliver working software early and often, there is an early return on the investment, and early feedback allows the development to more quickly reach the level of a minimum viable product (MVP).

3) The nature of complex systems

Additional support for the applicability of agile principles to the activity of software development comes from complexity science. Software development as an activity is an example of a complex adaptive system (CAS) [8][9]. A CAS can be described as "*a dynamic network of many agents acting in parallel, constantly acting and reacting to what the other agents are doing. The control of a CAS tends to be highly dispersed and decentralized. If there is to be any coherent behavior in the system, it has to arise from competition and cooperation among the agents themselves. The overall behavior of the system is the result of a huge number of decisions made every moment by many individual agents.*" [10] Typical examples of complex adaptive systems include climate; cities; markets; ecosystems; social networks; traffic flows; human social group activities, and the brain [11].

Kurtz and Snowden provide a useful sense-making model of systems in his Cynefin framework [12]. The model categorizes systems into four domains (shown in Fig. 5): Simple, Complicated, Complex, and Chaotic. The open space in the center represents Disorder, that is, not knowing which domain you are dealing with. This framework helps us to understand the types of decisions and behaviors that are likely to be successful in each domain based on the characteristics of the system.

In the Simple domain, the relationship between cause and effect is apparent, repeatable, and predictable. In this domain, we create best practices and categorize events based on the established patterns. An example of a system in the Simple domain is a bicycle.

In the Complicated domain, causes and effects are separated by both time and space, but they do repeat and they are knowable through analysis. This domain requires expert knowledge and analysis. There is not an obvious best practice, but there may be several good practices. A Boeing 747 would be an example of a system from the Complicated domain.

In the Complex domain, causes and effects are only perceivable in retrospect. They do not repeat except by random chance. Because it is impossible to predict in advance which actions will produce desired results, the best approach is trying multiple safe-to-fail experiments, evaluating the results, and then reinforcing the experiments that work and discontinuing those that do not. Here, practices emerge. Examples of systems from the Complex domain are an ant colony and a stock market.

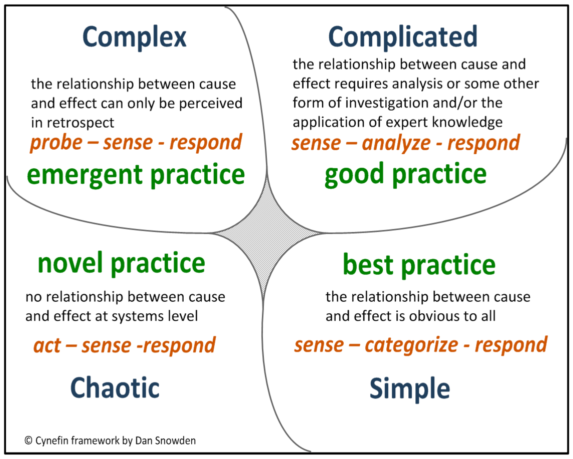

Complex
the relationship between cause and effect can only be perceived in retrospect
probe – sense - respond
emergent practice
Complicated
the relationship between cause and effect requires analysis or some other form of investigation and/or the application of expert knowledge
sense – analyze - respond
good practice
novel practice
no relationship between cause and effect at systems level
act – sense -respond
Chaotic
best practice
the relationship between cause and effect is obvious to all
sense – categorize - respond
Simple
© Cynefin framework by Dan Snowden

Figure 5. Cynefin framework

The Chaotic domain has no perceivable connection between cause and effect. The approach in this domain is to take immediate action in an attempt to establish order and move the system into some other domain. This is the domain of novel practice and can sometimes result in innovation and the breaking of old patterns. An example from the Chaotic domain is a house fire.

By examining the features of a software development project, we can see that many of the activities fall into the Complex and Complicated domains. The overall project with its many interactions between human agents and computer systems certainly are in the Complex domain. Given this knowledge, it is apparent that practices of agile are ideally suited to this system. The iterative delivery of software with regular feedback matches the domain-appropriate probe-sense-respond practice. The emergent design practiced by agile teams matches the emergent practice of multiple safe-to-fail experiments. Big-design upfront is of no value if cause and effect are only perceivable in retrospect and do not repeat. The waterfall model presumes that software development is a system in the Simple or Complicated domain and, for that reason, applies the incorrect practices.

These three drivers for agile have resulted in significant improvements in software and systems development, improvements that are equally relevant for advanced analytics.

IV. Agile Analytics Process Model

SAIC has developed a BDA process model extending the earlier CRISP-DM data mining model to incorporate the new technologies of big data and cloud. This BDA process model is called Data Science Edge^{TM} (DSE), shown in Fig. 6, which serves as our process model for Knowledge Discovery in Data Science (KDDS) [13]. It is formed around the National Institute of Standards and Technology (NIST) big data

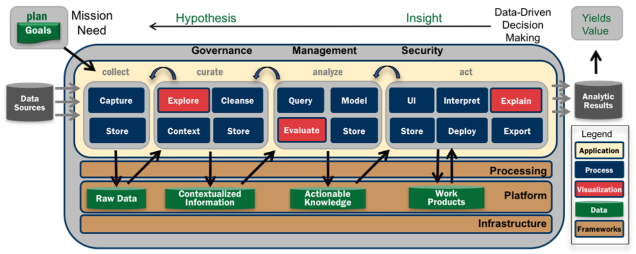


Figure 6. Data Science Edge^{TM}—a big data analytics process model

reference architecture (RA) [14], explicitly showing the frameworks that the application draws from. DSE provides a number of enhancements over CRISP-DM.

A. *The Complete Analytics Lifecycle*

DSE guides the data processing and analytics experiments that are beyond a software development lifecycle for BDA systems. DSE provides the step-by-step guidance for the additional needs of computational experiments to develop of machine learning analytics.

DSE is a five-step model—plan, collect, curate, analyze, act. It is organized around the maturity of the data to address the mission needs – going from the original raw data to contextualized information, to synthesized knowledge. The steps roughly align with the major steps from CRISP-DM but explicitly consider the collection from external sources and the data storage. CRISP-DM subtasks are re-grouped in different ways to accommodate big data techniques which change the ways data is managed. The sixth step in CRISP-DM is evaluation, which essentially was applied only to the accuracy of the modeling step. Evaluation has been pulled explicitly into each step rather that solely referring to the accuracy of models. This process model contains all the lower level activities of CRISP-DM—albeit with a different grouping—while adding in BDA, automation, and security and privacy activities.

B. *The Data Storage*

Organizing around data maturity facilitates the accommodation of the characteristics of the big data. For a large volume application, the original data are typically stored only in the form in which it is received, so the remaining curate, analyze, and act activities are performed dynamically. For high-velocity or data streaming applications, the entire process is done dynamically (typically in memory), and the only data storage is after the processing potentially to store aggregate or summary statistics. Since the analytics application is no longer separable from the storage technique, DSE includes advice on big data issues, such as choosing the most appropriate NoSQL technologies and methods for distributing data across a cluster of nodes. The determination of the types of storage at each step are stated explicitly in the process.

C. *Beyond Software Development*

In addition to the organization around big data maturity, the second differentiation of DSE from CRISP-DM or other analytics process models is the adherence to agile methodologies. As a systems integrator, SAIC's work typically involves the development of a BDA system, including software development for either integrating existing tools or developing new components and capabilities. Some analytic systems containing statistical analysis, and reporting or business intelligence actions reside in the Cynefin Simple or Complicated realms. DSE certainly covers this type of analytics lifecycle, but since the system can be described with detailed requirements, an SDLC is also sufficient. If, however, the analytics consists of any form of data mining, machine learning, or simulation - the realm is the Cynefin Complex where only outcomes can be specified. The more advanced types of analytics shown in Fig 1 involve computational experimentation and are thus not covered in an SDLC. It remains important that the additional experimental steps for analytics (including the use of multiple, ever more sophisticated, models) align with agile processes that are used for the software development portions. New sprints are not just to refine the cleansing or the models – they likely will contain new algorithms to achieve the greater refinement.

D. *Step Decomposition*

The major challenge to making DSE follow an agile methodology was to provide the guidance for which activities in each step would be fundamental for getting started and how to move quickly to an MVP product while considering all else as enhancements for later iterations.

The first issue was that the analytics lifecycle has always been considered a waterfall process. Each step in the process proceeded in sequence. At each step, an evaluation could determine that some changes were required upstream, such as a new dataset or a different cleansing or sub setting, essentially rewinding the process back to that step and then proceeding again through the waterfall process. Shifting to agile was a significant change in that some activities from each of the steps could be performed in a given sprint.

The second issue was the difficulty in determining how long specific steps will take. Advanced analytic systems have always operated in the Complex domain of Fig. 5. Datasets are typically not as described in the documentation; over time changes have been made and not documented. Datasets are typically not as clean as expected. Data cleansing is typically an open-ended task unless the data are well-governed and managed in the organization. This led to the introduction of an initial assessment phase, described in agile parlance as a spike. This initial effort clarified the issues in the data sources to guide the initial sprint planning.

E. *Analytics Improvement*

During systems development, the focus of each sprint is on creating new production ready capabilities. When the software development portions are "finished", there is often a need for continual refinement of analytics – as data or outcome requirements change over time. In a traditional systems development methodology, this would correspond to the continual operations and maintenance of the system. Under agile, DSE provides for an improvement phase that focuses purely on analytic refinement. The evaluation of the results of this phase could be the need to continue to enhance analytic models, to recommend the use of different analytic models, or the need to introduce new datasets into the system. Each of these results would have been out of bounds of normal operations and maintenance activities, but within an agile methodology these AgileOps changes can be re-introduced into the agile planning process for rapid accommodation.

V. Related Work

Because of the success of agile methodology in software development, analytics practitioners and researchers are seeking to determine how agile principles can be applied to the analytics lifecycle. Five representative areas are presented: (A) lifecycle models, (B) business intelligence systems, (C) the management of machine learning models, (D) architecture, and (E) analytics platforms.

A. Agile Data Science Lifecycle Models

In [15], Jurney establishes a theoretical framework to perform data science combined with agile philosophy. He outlines several areas:

- Iteration
- Intermediate output
- Prototyping
- Listening to what the data are telling us
- A data-value pyramid for structuring the process
- Pursuing the critical path to a product
- Documenting the analytics process as it unfolds

Like many data science process models, the author emphasizes the iterative nature of the task to create, test, and train learning algorithms, including any manual learning done during the stages of data preparation and exploration. Throughout these cycles of iteration, any intermediate output is committed to the source at the end of a sprint for sharing with other team members or end users if far enough along in development. The motivation behind sharing work on a continual basis, even if incomplete, is to encourage feedback to incorporate back into any future design decisions and requirements changes and to validate existing design. Prototyping ideas and experiments is important because of the uncertainty of the outcome of an experiment leading to any value insight or action. Applications are built based on what is possible with the data.

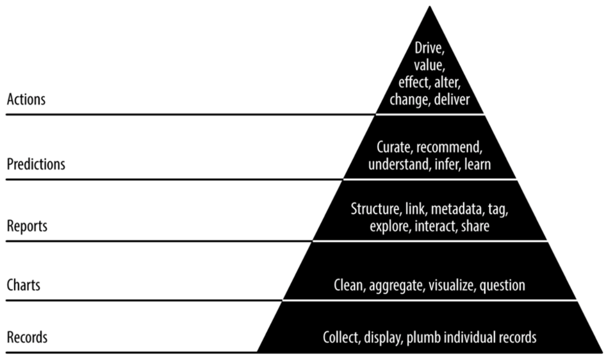

Actions
Drive, value, effect, alter, change, deliver
Predictions
Curate, recommend, understand, infer, learn
Reports
Structure, link, metadata, tag, explore, interact, share
Charts
Clean, aggregate, visualize, question
Records
Collect, display, plumb individual records

Figure 7. Agile Analytics Pyramid

The data-value pyramid, shown in Fig. 7, provides a process path from initial data collection to discovering useful actions. Value generation increases as the data scientist is able to work in the higher layers of the pyramid. The applied research process will not necessarily take a linear route up the pyramid but often will involve skipping back and forth and over layers.

The lowest layer, Records, is concerned with data collection, data flows, and the ability to simply display records. The Charts layer is where data refinement and analysis begin, including statistical summarization. The Reports layer is where much of the exploratory data analysis and identification of relationships in the data take place. The Predictions layer is about training algorithms to learn, and at the top, the Actions layer is focused on leveraging what has been discovered by the data science process to create business or mission value.

Finding the critical path is in essence eventually going through that one iteration up the pyramid through which the output leads to something actionable that creates value. By having documented the process thus far, a working set of documents have been generated to begin describing the product.

This work provides guidance on open source tools that can be brought together to facilitate the manual tasks needed in the various iterations but does not address the systems needed for automation; leveraging a large team, some potentially outside your organization; and questions of security and privacy.

B. Agile Business Intelligence

Larson and Chang [16] provide an overview of the agile process for business intelligence and examine the changes required for big data. The authors describe big data as a realm "where little time is spent defining requirements up front and the emphasis is on developing small projects quickly". They further describe big data as referring to nontraditional data sources, implying unstructured data. While there is more emphasis on unstructured data, there is no inclusion of high-volume or high-velocity datasets. Table I provides a comparison between the two Larson and Chang models and the DSE model presented in this paper.

The agile BI process does not explicitly address obtaining the data; it is assumed that the data are within the organization and that the data are already in hand. Neither model seems to address the need for data curation—the cleansing and transformation of data. In the paper discussion, there is no accommodation for the differing storage paradigms; the

Agile Business Intelligence	**Fast Analytics/ Data Science**	**DSE**
Discovery	Scope	Plan
	Data Acquisition/ Discover	Collect
		Curate
Design	Analyze/Visualize	Analyze
Development	Model/Design/ Develop	
	Validate	
Deploy	Deployment	Act
Value Delivery		

TABLE I. A comparison of the two models presented in [16] with the related SAIC DSE steps

discussion assumes the use of traditional relational databases. BDA methods are no longer separable from the choices for distributing the data across nodes and running parallel analytics. Their data science process is in good alignment with DSE except that they separate out individual steps within the analyze process.

C. *Agile Machine Learning*

In an interview this year [17], former senior data science manager and principal data scientist at Walmart Labs, Jennifer Prendki, discusses how she, as the interviewer described it, "built agile processes and teams for machine learning" and "managed and measured machine learning models in production". Prendki was a member of the search algorithms team for Walmart's online store, which is subdivided into three groups: perceive, guide, and measure. The perceive team attempts to understand a shopper's intent, and the guide team attempts to show the user products that match the intent. The measure team acts as an auditor of the other teams by creating success measurements to evaluate their models, provide feedback to ensure best practices, and make suggestions about how those teams can improve model performance. Traditionally, the data scientist who develops a learning algorithm is also the person who evaluates and selects candidate models and iterates the process until final selection. Having a separate review team supplement the evaluation process potentially reduces the risk of latent biases by the model developer during these stages and allows for a fresh perspective on how success should be defined.

Regarding model management and maintenance, Prendki introduced the notion of machine learning lifecycle management, which is essentially a checklist of what should be done before pushing models to production:

- A data scientist must be able state what the model accuracy is as well as how much CPU the model consumes.
- The agile technical debt must be minimized and extends beyond code debt to include system debt, data debt, and machine learning debt. Prendki defines data debt as issues in data quality but also situations in which the organization is not using datasets available to them when a competitor is using them. System debt is the condition of having legacy systems no longer being improved or updated or replacing them altogether. Machine learning debt is when the deployed model is not optimal, for example, incorrect retraining frequencies or not monitoring how changes in the data affect its performance
- To ensure transparency and reproducibility, everything should be documented.
- You should know your failures and weaknesses because management is a lifecycle, and there will be more opportunities to work on them.

According to Prendki, adopting this management lifecycle is about changing the culture to believe that doing things right is important. Prendki provides an excellent set of lessons learned from agile BDA development but does not describe her complete model.

D. *Agile Big Data Architecture*

Agility in architecture is an important consideration, because one may not know in advance what the requirements are for future analytics projects. The architecture must have the flexibility to adapt to change. In [18], Pääkkönen and Pakkala attempt to answer two key questions:

1. How should a big data system be designed and developed to effectively support advanced analytics?
2. How should the agile process be adapted for BDA development?

Through a series of three research cycles spanning ten case studies, the authors explored various architectural design methods in which changes were adopted after identifying the strengths and limitations of each. Some key lessons learned include the following:

- "Architecture-supported agile spikes are necessary to address rapid technology changes and emerging requirements."
- "The use of reference architectures (RA) increases agility."
- "A key part of architecture agility is the ability to change technologies within [an RA's] block."
- "An architectural approach to DevOps was critical to achieving strategic control over continuous delivery goals."
- Technical and business "feedback loops need to be open."

The lessons learned from the authors' work show that, in order to support agile advanced analytics, the architecture also needs to embrace agility. Specifically, the RA must consist of flexible technology families so new technologies can be adopted easily and accommodate changing requirements during the analytics development lifecycle. The authors provide interesting lessons learned in continuing to increase big data platform capacity to meet the ever-increasing demands of analytics.

E. *Other Agile Small-Data Analytics Discussions*

A number of books and articles provide discussions of agile and include discussions of analytics. These discussions focus on what has come to be called *small data* - data that is manageable on individual resources - not requiring distribution and parallel processing across a cluster.

Simon [19] provides a good discussion of agile, but the analytics discussion follows a step-by-step approach more like waterfall. The use cases do provide a number of excellent examples of iterative improvements to mission questions based on analytics results.

Alt-Simmons [20] has an excellent description of analytics project development using agile. Each step is well described with the analytics itself again follows CRISP-DM, and deployment is an after-thought.

Collier [21] addresses agile in the context of business intelligence and data warehousing. He discusses the difference between software development and data warehouse development, including the challenges of refactoring. He also covers the changes in project management due to agile.

Saltz, Shamshurin, and Crowston [22] considered the effects different methodologies on master's level student analytics teams, determining that Kanban and CRISP-DM were more successful than Agile Scrum.

F. Agile Analytics Platforms

Big data engineering is a separate discipline from BDA done through data science. While data scientists need to be conceptually aware of the techniques and issues of big data platforms, they need not be the ones installing and maintaining such platforms. For agile analytics, there is a significant need for analytics platforms that insulate the data scientist from the underlying storage implementation details. Jurney [15] described an open source platform using Avro, Pig, MongoDB, Elastic Search, and Bootstrap, which could be used to address a number of data science problems. SAIC has internally developed the Machine Augmented Analytics Platform (MAAP), which is an enhancement of Zeppelin notebooks that can run across a Mesos cluster using Spark and Docker containers. The Cloudera Data Workbench (CDW) [23] enables data scientists to develop notebooks that invoke distributed compute jobs. The Griffon Data Science Virtual Environment [24] runs on Ubuntu MATE and includes a number of tools for programming languages, editors and notebooks, and machine learning.

Traditional relational databases continue to evolve, pushing analytics into the database as functions. Likewise, BDA platforms will continue to evolve, but data science needs BDA platforms that allow the rapid development of scalable agile analytics without the hindrance of continual software and platform installation.

VI. Discussion

As discussed in Section III, agile methodologies have made a significant impact on reducing development and rework time and increasing the success rate of software development projects. Following this paradigm, a number of fundamental assumptions must change for the BDA lifecycle.

A. Big Data Effects

The fundamental trade-off for agile BDA is that results must be presented before they are considered done. The data quality may be low, but the results should be presented to the customer for feedback anyway. The data science community has proposed this approach, arguing that large data volumes allows the curation assumption that most errors will cancel each other out. This shortcut to data quality improvement bypasses the usual full scan statistics to determine distributions and outliers—a task that is often impractical. However, this assumption cannot hold when there are systemic errors in the data.

A second assumption often made is that more data beats better algorithms. For large enough datasets, it is assumed that many errors will cancel. Thus, it is important to get a overview of any proposed analytic, so a quick first look at the analysis is good enough to evaluate whether to continue. It is assumed that more data will result in better outcomes than continual small refinements to analytic models.

These two assumptions are truly fundamental changes in typical systems development, especially for data warehouses or for those used to sample data for their analytics. A large part of the effort in building an enterprise data warehouse has been concerned with ensuring the absolute quality of the data. Likewise, data miners would never want to present results that have not been evaluated as accurate. While there is always the risk of inaccuracies, they should resolve in subsequent agile iterations.

B. Agile Effects

Since advanced analytics is in the Cynefin Complex realm, the objective is to meet a specific outcome rather than conform to a set of detailed requirements. This means the goal is to begin with steps toward an initial MVP. This should expose issues in the overall approach and allow the team to fail fast. The original or raw data initially should be pushed through and presented at least in a limited way in the act stage. Each subsequent iteration continues to add additional cleansing or transformation, model refinement, or results presentation to provide a continual improvement in outcomes.

Advanced analytics or machine learning are inherently experimental. As you progress up the analytics ladder, models become computationally more complex and require cycles of training and testing. This experimental process does not fit into a specific time window. This improvement process is a duration task, at best more in alignment with the Kanban agile methodology.

VII. Conclusions

Agile practices and philosophy have transformed software development. It solves some of the issues inherent in the highly linear approach of waterfall methodologies. There is similarity between waterfall development and outcomes-driven analytics development, where final results are expected to fulfill an initial set of well-defined goals. However, because of the experimental nature of analytics development, detailed requirements cannot be set with complete confidence. It is only when the results are meeting the needs of the organization that the details of the end-state analytics models become clear.

There is a critical need for BDA to follow the software development lessons learned to adopt agile methodologies. By adopting agile philosophy for analytics development, results are expected to be shared more frequently to form a feedback loop of stakeholder opinion and use those needs to validate the current state and influence its evolution to an agreeable end state.

We have presented our process model for BDA and discussed how it is an improvement over the current industry standard, CRISP-DM, because of the alignment with software development, the incorporation of agile practices, and the adoption of a modern, flexible architecture based on NIST's big data RA. This new process model provides the added agility to adapt to the differences in architecture depending on the data characteristics of volume, velocity, variety or variability by explicitly considering the architecture and the choice of NoSQL methods.

For the community, there is a need to revisit the development of a new KDDS process model to accommodate the technique changes that have occurred after 2000 [13]. This paper describes the conceptual changes to a data analytics lifecycle that we have made to adapt to an agile methodology. It is hoped that the community will also assume an agile methodology in the development of any new model for KDDS.

References

[1] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. "CRISP-DM 1.0: Step-by-step data mining guide", The CRISP-DM Consortium, 2000.

[2] G. Piatetsky-Shapiro, "CRISP-DM, still the top methodology for analytics, datamining, or data science projects", https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html, 2014, accessed online October 9, 2017.

[3] J. Taylor, "Four problems in using CRISP-DM and how to fix them", http://www.kdnuggets.com/2017/01/four-problems-crisp-dm-fix.html, accessed October 9, 2017.

[4] "NIST Big Data Interoperablity Framework: Volume 1, Definitions", NIST Special Publication 1500-2, N. Grady and W. Chang, Eds. https://bigdatawg.nist.gov/V2_output_docs.php, accessed October 6, 2017.

[5] W. Royce, "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON, Aug. 26, 1970.

[6] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, D. Thomas, (2001). "Manifesto for Agile Software Development*", Retrieved October 9, 2017, from http://agilemanifesto.org.

[7] A. Cockburn (2008). "Design as Knowledge Acquisition*", Retrieved October 9, 2017, from http://a.cockburn.us/1973.

[8] J. Pelrine, (2011, May 11). On Understanding Software Agility—A Social Complexity Point Of View. *E:CO*, pp. 27-37.

[9] M. M. Waldrop, "*Complexity: The Emerging Science at the Edge of Order and Chaos", Simon & Schuster, 1992.

[10] Ivanov, D & Ivanov, M, "A Framework of Adaptive Xontrol for Complex Production and Logistics", in Dynamics and Logistics: First International Conference (LDIC 2007), Springer-Verlag, Berlin Heidelberg, pp. 151-159, doi: 10.1007/978-3-540-76862-3_14

[11] Wikipedia. "Complex adaptive system." Retrieved October 10, 2017, from https://en.wikipedia.org/wiki/Complex_adaptive_system.

[12] Kurtz, C. F., and Snowden, "The New Dynamics of Strategy: Sense-making in a Complex-Complicated World". *IBM Systems Journal*, vol. 42, no. 3, pp. 462-83 (2003).

[13] N. Grady, "Knowledge Discovery in Data Science", Proc. IEEE International Conference on Big Data, IEEE Press, Dec. 2016, pp. 1603–1608, 10.1109/BigData.2016.7840770, accessed October 9, 2017.

[14] "NIST Big Data Interoperablity Framework: Volume 6, Reference Architecture", NIST SP 1500–6, D. Boyd and W. Chang, Eds. https://bigdatawg.nist.gov/V2_output_docs.php, accessed October 6, 2017.

[15] R. Jurney, "Agile Data Science", O'Reilly Media, Inc., 2014.

[16] D. Larson and V. Chang, "A review and future direction of agile, business intelligence, analytics, and data science", International Journal of Information Management, vol. 36, pp. 700–710, 2016.

[17] S. Charrington, "Agile machine learning at Walmart with Jennifer Prendki", This Week in Machine Learning & AI, [Online] https://twimlai.com/twiml-talk-46-jennifer-prendki-agile-machine-learning-walmart/, accessed October 9, 2017.

[18] P. Pääkkönen and D. Pakkala, "Reference architecture and classification of technologies, products and services for big data systems", Big Data Research, vol. 2, pp 166–186, 2015.

[19] P. Simon, "Analytics: the Agile Way", Wiley & Sons, 2017

[20] R. Alt-Simmons, "Agile by Design: An Implementation Guide to Analytic Lifecycle Mangement", Wiley & Sons, 2015

[21] K.W. Collier, "Agile Analytics: A Value-driven Approach to Business Intelligence and Data Warehousing", Addison-Wesley, 2011

[22] J. Saltz, I. Shamshurin, and K. Croston, "Comparing Data Science Project Management Methodologies via a Controlled Experiment", in Proceedings of the 50th Hawaii International Conference on Systems Sciences (HICSS), pp. 1013-1022, 2017.

[23] Cloudera Data Science Workbench, https://www.cloudera.com/products/data-science-and-engineering/data-science-workbench.html, accessed on October 9, 2017.

[24] Griffon Data Science Virtual Environment, https://github.com/gtkcyber/griffon-vm, accessed on October 9, 2017.