Agile Big Data Analytics for Web-Based Systems: An Architecture-Centric Approach

Hong-Mei Chen, Rick Kazman, *Senior Member, IEEE*, and Serge Haziyev

Abstract—This article contributes an architecture-centric methodology, called AABA (Architecture-centric Agile Big data Analytics), to address the technical, organizational, and rapid technology change challenges of both big data system development and agile delivery of big data analytics for Web-based Systems (WBS). As the first of its kind, AABA fills a methodological void by adopting an architecture-centric approach, advancing and integrating software architecture analysis and design, big data modeling and agile practices. This article describes how AABA was developed, evolved and validated *simultaneously* in 10 empirical WBS case studies through three CPR (Collaborative Practice Research) cycles. In addition, this article presents an 11th case study illustrating the processes, methods and techniques/tools in AABA for cost-effectively achieving business goals and architecture agility in a large scale WBS. All 11 case studies showed that architecture-centric design, development, and operation is key to taming technical complexity and achieving agility necessary for successful WBS big data analytics development. Our contribution is novel and important. The use of reference architectures, a design concepts catalog and architectural spikes in AABA are advancements to architecture design methods. In addition, our architecture-centric approach to DevOps was critical for achieving strategic control over continuous big data value delivery for WBS.

Index Terms—Software architecture, data systems, system analysis and design

1 Introduction

Big data is transforming business landscapes and the unprecedented excitement surrounding business analytics and big data has been generated primarily from the web and e-commerce communities [16]. Web-based systems (WBS), ranging from product recommender systems, e-commerce platforms, social networking, gambling, gaming, to CRM (Customer Relationship Management), and SCM (Supply Chain Management) applications, have traditionally relied on data analytics to operate. For WBS, data analytics is about generating predictions and actionable insights to improve real-time customer experience, increase market and customer intelligence, predict customer behaviors, optimize operational efficiency, personalize service provision, prevent security threats and frauds, minimize brand risk, and innovate processes and services.

Big data analytics is predicted to be a US $125 billion market [43]. Unlike traditional "small" data analytics on structured data collected from internally-focused transactional systems, big data analytics, utilizing data from *everywhere,* is expected to take WBS to the next level in real-time operation optimization, customer intelligence and service innovation. New applications are being rapidly developed, including web analytics, social media analytics (text mining, sentiment analysis, social network analysis, graph analysis, location-aware analytics, psychometrics analysis), real-time operations management analytics, stream analytics, visual analytics, etc.

In reaping the unprecedented opportunities offered by big data analytics, WBS face many challenges, including technical [36], organizational [29], [38] and rapid technology change [42] challenges. These challenges are not limited to WBS, but the characteristics of WBS exacerbate the problems of rapid delivery of big data analytics, including (1) High interactivity and near-real-time performance, (2) continuous operation: 24×7 availability, and (3) high scalability: must scale seamlessly, with no downtime.

First, *technical* challenges are pertinent to addressing the 5Vs of big data (Volume, Velocity, Variety, Veracity, Value) to support advanced analytics. Recently, big data analytics have been used with data sets that are so large (from Terabytes to Exabytes) and complex that they require advanced and unique data storage, management, analysis, and visualization technologies [16]. Importantly, success in big data analytics for WBS depends on building an infrastructure that effectively orchestrates technology components for ingesting, processing, storing, integrating, analyzing and visualizing large amount of data of various types (structured, semi-structured and unstructured) from both internal and external sources—literally *everywhere*, including social media, IoT, web logs, and web-crawler information [20]. The 5V challenges have not only motivated new technology development including MPP databases, NoSQL databases, New SQL databases, and in-memory databases, but also induced paradigm shifts in data management (see Table 1), requiring new architectures and new design thinking [20]. The death of relational datawarehouses has been predicted, as they are believed to be inadequate to support big data exploration for value discovery. Many new concepts such as data lake, data refinery, Lambda architecture [40] and Polyglot persistence [44] have been recently proposed and are considered to be paradigm

• *H.-M. Chen and R. Kazman are with the Information Technology Management, University of Hawaii, Honolulu, HI.*
E-mail: {hmchen, kazman}@hawaii.edu.
• *S. Haziyev is with the Architecture Group, Softserve Inc., Austin, TX.*
E-mail: shaziyev@softserveinc.com.

Manuscript received 6 July 2015; revised 10 Apr. 2016; accepted 29 Apr. 2016. Date of publication 10 May 2016; date of current version 28 Oct. 2016.
Recommended for acceptance by Y. Peng.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TBDATA.2016.2564982

TABLE 1
Small Data Analytics versus Big Data Analytics

		Small Data Analytics	Big data analytics
1	Analytics Architecture	data warehouse	data Lake
2	Data System Architecture	monolithic systems with vertical scaling	horizontally scaled
3	Agile Analytics	• Product delivery • Focus on requirement • Cycle time = 6-8 months	Value delivery • Focus on entire life cycle • Cycle time < 1 week • Need automation
4	Data Action Time Window	batch processing	real time data action
5	Data Life Cycle	data-at-rest: collection, preparation, [persistent store], analysis, action	data-in-motion (stored after action) data-in-use (stored as raw data after collection, schema on read)
6	Data Model	Relational ACID	Not Only SQL (NoSQL) BASE
7	Data Retrieval & Metadata management	Schema	Schemaless
8	Transaction & Analytics Platforms	Separate	simultaneous hypothesis validation and knowledge discovery
9	Separation of Concerns	data independence	data-program dependence
10	Implementation Effort	mostly coding	mostly orchestration (technology selection dependent)

shifts. For example, for value exploration, voluminous raw data of various types from everywhere are ingested into a "data lake" or "refinery", such as Hadoop HDFS. A process called "schema on read" is then employed when certain data are needed (data-in-use). For WBS, velocity is key to the challenges: managing voluminous "data in motion"—streaming data for (near)-real time analytics for "2 second advantages." In contrast to "small" data analytics that are often batch-processed, for data-in-motion or real time streaming data, data are analyzed on the fly, actions and first taken and *then* stored. The Lambda architecture [27] was proposed to address the real-time stream processing of raw data combined with materialized views of stored data residing in batch-processing oriented storage to offer big data real time analytics performance advantages.

Second, *organizational* challenges surrounds how business analysts, data scientists, and other stakeholders work with system design, development, and operations teams to maximize value from big data. Traditionally for "small" data systems, the agile practices for data analytics have data analysts and system engineering teams working in silos. The focus is on requirements for new features and product delivery and the cycle time is typically 6-8 months. Competitive pressures, however, require *rapid* delivery of value, e.g., useful predictions, actionable insights and real-time analytics. The cycle time may be as short as one week, preferably even shorter for WBS. This requires a tight integration of teams and automation: data scientists will have system support for value exploration, model development, testing and rapid deployment of new models while software engineers understand how decision makers, business analysts, and data scientists are going to use the data so that they can effectively design the big data systems, orchestrating different types of storage systems, tools for different types of analytics and visualizations, and so on. New methods for facilitating process integration and systematically achieving continuous value discovery and rapid deployment are needed.

Third, *rapid technological changes* pose a constant threat. The proliferation and advancements of open source and proprietary big data technologies add a new dimension of complexity to the technical and organizational challenges above. For instance, there are over 150 products in the NoSQL space alone [20]. The large space of technology choices is problematic. As shown in Table 1, big data system development is primarily a process of "orchestration" of available technologies. Hence the challenges in technology selection are non-trivial and affect the selection of architecture patterns, data models, programming languages, query languages, and access methods, and these in turn affect system performance, latency, scalability, availability, consistency, modifiability, etc. In addition, many enterprises need to refactor their systems due to rapid technology changes. Whether, when, with whom and how to adopt and integrate (often unfamiliar) technologies, potentially requiring refactoring or redesign is always a challenge but the particularly rapid pace of changes in big data technology drives the urgency.

Furthermore, agile principles are being enthusiastically pursued for managing the risks associated with these technical, organizational and technology change challenges. Although agile analytics for the traditional structured Relational datawarehouse domain is already a familiar concept [20], agile development of big data analytics is relatively new and necessitates careful adaptation as big data analytics are dramatically different from "small" data analytics as summarized in Table 1.

Addressing the aforementioned technical, organizational and rapid technology change challenges results in two pressing research questions: *RQ-1)* how should a big data system be developed to effectively support agile analytics for WBS? and *RQ-2)* how should agile practices for "small" data systems be changed for rapid, continuous big data value delivery of WBS? Existing "small" data design and development methods are inadequate for addressing these challenges and failures of big data projects abound. In fact, while big data was near the peak of Gartner's 2014 -2015 Hype Cycle of emerging technology [31], 55 percent of big data projects were not completed [35]. Our own study of 23 large enterprises confirmed

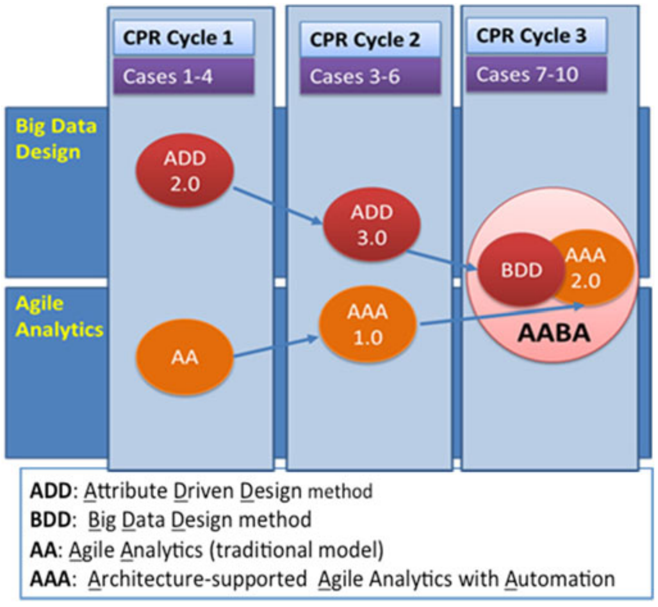
CPR Cycle 1
Cases 1-4
CPR Cycle 2
Cases 3-6
CPR Cycle 3
Cases 7-10
Big Data Design
Agile Analytics
ADD 2.0
ADD 3.0
BDD
AAA 2.0
AAA 1.0
AA
AABA
ADD: Attribute Driven Design method
BDD: Big Data Design method
AA: Agile Analytics (traditional model)
AAA: Architecture-supported Agile Analytics with Automation

Fig. 1. CPR cycles for AABA development.

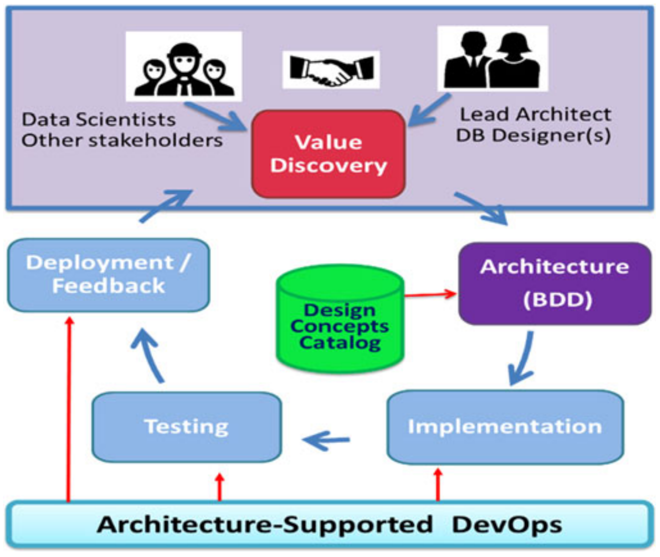

Fig. 2. AABA methodology overview.

Gartner's reports on the scarcity of big data deployment and needs for new methods [21], [22].

In filling the existing methodological void, we followed the Design Science Research (DSR) methodology [33] to identify an architecture-centric approach to develop our new method, called AABA (Architecture-centric Agile Big data Analytics) depicted in Fig. 2. Our architecture-centric approach is a novel contribution as architecture design was either trivial or non-existent in existing "small" data analytics development methodology.

We employed a Collaborative Practice Research (CPR) method [39], following nine steps of each CPR iteration in 3 cycles utilizing 10 WBS big data projects (See Table 5) in an outsourcing company, to *simultaneously* carry out DSR Activities of design, development, demonstration and evaluation of our AABA methodology (see Table 2). The CPR method balances research relevance and rigor [33], [39]. Our research falls in the quadrant of "Invention: New Solutions for New Problems" of the DSR Knowledge Contribution Framework [33].

AABA is a clear departure from the accepted ways of "small" data analytics system design thinking and existing agile practices. Our research process is a largely exploratory search over a new, complex problem space of big data. The CPR method is particularly appropriate for exploratory case studies as it allows the course of the study to be adjusted along the way to account for what is learned [18]. Our research output, AABA, contributes a new methodology that not only advances existing system engineering research but also helps practitioners be better equipped for their big data endeavors.

In what follows, we discuss our research method and provide an overview of our research process following the CPR steps. In Section 3, we show how the AABA methodology evolved, with lessons learned, and was developed and validated through 3 CPR cycles. In Section 4, we describe the AABA methodology and in Section 5 we illustrate the AABA design principles and agile process in a case study of a large online travel advising site. In Section 6, we discuss limitations. Section 7 concludes the paper.

2 Research Method

2.1 Collaborative Practice Research

The CPR method we employed is an empirical case study method to identify issues in practice, and collaboratively explore, develop and validate the results with practitioners. This method has the advantage of balancing research relevance and rigor and is suited for exploratory, "how" research questions [33], [39], [45] like ours. Multiple cases increase methodological rigor [45]).

We selected a company, SSV, in the outsourcing industry because many firms have turned to outsourcing companies for their big data solutions [37]. SSV provides global IT outsourcing solutions with over 4,000 employees and offices in the US and Europe. They have on-going big data projects that we utilized as embedded multiple case studies.

The selection of our case company, SSV, was opportunistic. SSV and the researchers came together as SSV was seeking new methods because their early big project experiences suggested existing methodology was inadequate for big data analytics development.

Table 2 shows the CPR steps and the initiation, 3 iterations, and closing of our research cycles in collaboration with SSV. An iteration in the CPR process consists of 9 steps: (1) Appreciate problem situation, (2) Study literature, (3) Develop framework, (4) Evolve Method, (5) Action, (6) Evaluate experiences, (7) Exit, (8) Assess usefulness, and (9) Elicit research results. Importantly, in each iteration, an action research in the 5th step is carried out to evaluate and validate the method(s) being evolved. We will detail results in each iteration in Section 3. In what follows, we detailed the rationale of adopting an architecture-centric approach in the Initiating phase.

2.2 An Architecture-Centric Approach

After a broad literature search in the Initiation phase, we identified an architecture-centric approach underlying a new methodology for big data analytics development.

For complex system development, architecture analysis and design is critical [6]. Architecture-centric approaches

TABLE 2
Collaborative Practice Research Process and Results Summary

CPR steps	Initiating	First iteration	Second iteration	Third iteration	Closing
1. Appreciate problem situation	*Part of ongoing research Cola bo rat ion ‘Lessons Zero-1, Zero-2	*New reference architectures; *New technologies; *New requirements for rapid vate deliveiy			
2. Study literature	Big data, "small" system development, big data analytics, agile practice, Business-IT alignment method, Architecture analysis methods, Architecture Design methods	*Architecturedesign methods: ADD 2.0, ACDM, RUP ‘Agile analytics: traditional data warehousing methodologies vs. Data lake, NoSQL polyglot persistence New big data technologies	Eco-Architecture method, NoSQL data modeling, ADD 2.5, BITAM, reference architectures, design patterns, frameworks and tactics; DevOps		
3. Develop framework	Identify new approach: Architecture-centric Approach	*Incorporate ADD to integrate EDW with NoSQL methodology ‘Formulate agile big data design process	‘Develop technology catalogue framework * Develop BDD method * Incorporate ADD 2.5 to BDD ‘Incorporate Eco-ARCH into agile value discovery process	‘Integrate BDD and AAA 2.0	
4. Evolving method		*Big data design method: ADD 2.0 *Agile Analytics (AA) method	‘Big data design: ADD 2.5, BD-DFD *Agile analytics: with architecture support (AAA 1.0)	*AABA ‘AAA2.0	
5. Action		Cases 1-4 ADD 2.0 ADD 3.0 AA ·» AAA 1.0	Cases 3-7 ADD 3.0 BDD AAA 1.0 AAA 2.0	Cases 7-11 BDD + AAA 2.0	
6. Evaluate experiences		Lessons learned : Lessons 1-5	Lessons learned: Lesson 6	Lessons learned: Lesson 7	
7. Exit		Projects 1-2 conclude;	Projects 3-6 conclude	‘Projects 7-11 conclude ‘Develop Lambda Architecture Accelerator	
8. Assess usefulness		Business goals assessments for projects; Reference architectures: Data lake& Lambda architecture; Assess ment of ADD 2.0, AAA 1.0	Assess ment of BDD, AAA2.0	Assess me nt of AAB A	Success Metrics; Client company success
9. Elicit research results		Reference architecture Polyglot model	BDD method Technology catalog method	AABA methodology Strategic prototyping	Publications; formalize AABA methodology for SSV

are not new and are well-accepted in industry. Many architecture analysis and design methods exist. In a nutshell, the software architecture of a system is “the set of structures needed to reason about the system which comprise software elements, relations among them and properties of both” [6]. Software architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by development teams. The architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be consistently achieved without a unifying architectural vision. Architecture analysis methods, such as ATAM [24], BITAM [17] and CBAM [4], analyze architectures for identifying design tradeoffs, business-IT alignment, and cost-benefit analysis. Architecture design applies decisions to satisfy a set of architectural drivers—the most important functional, quality attribute, and constraint requirements that shape a system. A number of architecture design methods have also appeared during the last decade, including ADD, BAPO, RUP, ACDM, etc. [15].

SSV has long realized the importance of architecture design for big data system and have used an older version of ADD [6], which was chosen for further development to address new agile big data analytics challenges. Their big data client companies, prior to coming to SSV, have paid heavy costs for improper architecture design and inadequate design evaluation for big data system development. We call these cases "Lesson Zero."

Lesson Zero-1: ***Improper architecture design is costly and time-consuming to rectify.*** A client created a new architecture for a system for collecting customer feedback (with a different outsourcing company). Their goal was to migrate to new technologies (as their existing MySQL-based architecture was outdated). They chose NoSQL technologies but the TCO (total cost of ownership) was too high (due to hosting costs in particular). However, they only realized this once they began migrating customers. It took almost 1 year to optimize this architecture so that they could continue to migrate the remaining 80 percent of their customers to the new platform. The problem was that they did not do a proper proof-of-concept, so they misunderstood resource usage and did not do proper capacity planning. Instead they charged ahead with implementation and created software that was too resource-hungry.

Lesson Zero-2: *The consequences of poor architecture choices are far-reaching*. A client, prior to coming to SSV, made a poor architecture decision and selected an inappropriate technology (HBase) for their new system. This system needed to monitor an IT infrastructure and it had to support both single node and cluster deployments. The problem was that HBase did not work well on single node servers: memory consumption was so high that performance was impacted, and customers complained.

These lessons propelled SSV to seek collaboration with the researchers, who collectively are creators of architecture analysis methods: ATAM, BITAM, Eco-ARCH [19] as well as the architecture design method ADD.

In what follows, we articulate that an architecture-centric approach is both appropriate and critical for two aspects of agile big data analytics: (1) big data system design and (2) agile analytics development. We then refined our research questions and show the evolution of new techniques and methods we developed based on the architecture-centric approach through three CPR cycles in each of the two aspects (see Fig. 1).

2.2.1 Architecture-Centric Big Data System Design

Architecture design has been trivial or non-existent in traditional "small" data system or datawarehouse development. This system development, dominated by the relational model, benefits from the ANSI standard 3 tier DBMS architecture which clearly defines data/program independence for data system development [22]. The data system development process is well-established, consisting of seven phases: requirement analysis, conceptual design, selection of DBMS, logical design, physical design, prototyping/testing, implementation and performance evaluation [22]. Typically, a RAD or agile lifecycle is employed for iteratively refining performance, accommodating new requirements, and managing system evolution. The architecture design for such systems has been straightforward. The key decisions—reference architecture and major patterns—in traditional DB design are already made: the N-tier client-server architecture is the norm.

But big data analytics requires capabilities beyond traditional relational datawarehouses. Traditionally, a relational datawarehouse integrates data from various operational stores, through an ETL (extract, transform, load) process which cleans and transforms data and loads it into pre-specified data cubes (based on star or snowflake schemas) for later manipulation by OLAP (Online Analytical Processing) tools, data mining, and visualization tools. The data is "at rest" and relatively small and this process is batch-oriented. The architectures of such systems are standard and seldom change.

However, the 5Vs of big data have changed these established processes, prompted paradigm shifts, undone program-data independence and thus underscored the need for architecture design. Architecture design is also critical for constantly including new source of "dirty" and "fast" data and constantly integrating the old systems with the new.

For instance, a travel review site may want to link credit card information to verify a hotel stay, to validate the legitimacy of a review. How to build the agility to accommodate new data sources and integrate and interoperate with external systems in the future highlights the criticality of architecture choices in big data system development. Hence, *RQ-1* has been refined to: How to extend existing architectural methods and to develop and integrate polyglot data modeling, architecture design, and technology orchestration techniques in a single effective and efficient big data design method to support agile analytics for WBS.

2.2.2 Architecture-Centric Agile Analytics Development

The relationship between architecture design and agile practices has been the subject of some debate over the past decade. Architecture design is "up-front" work, which appears to be contradictory to agility. Although we believe, and much research has shown, that architectural principles and agile practices are actually well aligned [6], [8], [10], this position has not always been universally accepted.

Agile practices, according to the original Agile Manifesto emphasize: "Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan" [5]. The creators of the Manifesto described 12 principles behind it. While 11 of these are compatible with architectural practices, one of them is not. This principle is: "The best architectures, requirements, and designs emerge from self-organizing teams." While this principle may have held true for small and perhaps even medium-sized projects, we are unaware of any cases where it has been successful in large projects, particularly those with complex requirements and distributed development.

Two key issues motivate our architecture-centric approach to agile development for big data analytics for WBS. First, architectural abstractions are the key enabler of agility. Architecture structure determines the key project technical abstractions (and the interfaces among them), which in turn determines technology choices, team structures, and communication paths. Conway's law [27] observed that the software interfaces structure of a system will reflect the social boundaries of the organization(s) that produced it.

Second, an architecture-centric approach facilitates future rapid iterations with less disruption, as well as lower risk and cost, by creating appropriate architecture abstractions. For example, if it was anticipated that a technology family would rapidly change (with the technologies themselves introducing new capabilities or with new entrants to the technology family) then a prudent architect would insert an abstraction layer as an intermediary between the technology family and the rest of the system, to insulate the rest of the system from such changes.

As a result, the critical research questions surrounding agile development for big data analytics are: (1) how to balance between over-architecting and doing no architecture design work at all, including finding the "sweet spot" [11]—the right amount of up-front architecture design, based on factors such as project size, requirements complexity and volatility, and degree of distribution of development; and (2) how architecture abstraction can accommodate requirements that emerge from constant value discovery in big data analytics. These questions call for architecture methods that can lay the foundation to support agility in both initial system construction and future rapid iterations in big data analytics. We thus refined our *RQ-2* to be: how should agile principles be integrated with architecture design to achieve effective agile big data analytics for WBS?

3 CPR Cycles and Results

As shown in Fig. 1, the AABA methodology grew from two aspects of big data analytics development: 1) big data design and 2) agile analytics. The version numbers of the methods are our notation to help denote the trajectory of development and differentiate their features. Cases 3 and 4 went through redesign due to technology and method changes, and thus appear in two CPR cycles. To distinguish them we term them "early stages of Cases 3-4" and "redesigned Cases 3-4". We now present our 3 CPR cycles in each of the two aspects.

3.1 For Big Data System Design

3.1.1 CPR Cycle 1: ADD 2.0

In the first iteration of the CPR (Cases 1-4, starting in late 2010), SSV employed an architecture design method called Attribute-Driven Design (ADD) that was developed by the Carnegie Mellon Software Engineering Institute (SEI). The first version of ADD (ADD 1.0) was published in January 2,000 and the second version (ADD 2.0) was published in November 2006 [6]. SSV's architects took SEI training and were well-versed in ADD 2.0.

When ADD 1.0 appeared, it was the first design method to focus specifically on quality attributes and their achievement through the selection of architectural structures and their representation through views. An important contribution of ADD 2.0 was that it included lightweight architecture analysis and documentation as integral parts of the design process. In ADD 2.0, activities included performing a more formal evaluation of the design, perhaps using an architecture analysis method such as the ATAM [6] or BITAM [17].

Based on our literature review and experiences and as shown in SSV's early projects, big data system development rests largely on the *orchestration* of a set of technologies. While ADD 2.0 was useful for linking quality attributes to design choices, there were three shortcomings that needed to be addressed for big data system development: first, ADD 2.0 guides the architect to use and combine *tactics* and *patterns* to achieve the satisfaction of quality attribute scenarios. But patterns and tactics are abstractions. The method did not explain how to map these abstractions to concrete implementation technologies. Second, ADD 2.0 was invented before agile methods became popular and thus did not offer guidance for architecture design in an agile setting. Third, ADD 2.0 was meant to be general, and hence technology agnostic. While generality is good, ADD 2.0 did not explicitly promote the (re)use of reference architectures which are an ideal starting point for big data system architects. These shortcomings were addressed in the next CPR cycle.

3.1.2 CPR Cycle 2: ADD 3.0

ADD 3.0 was developed to address the issues of ADD 2.0 through Cases 3-7. It was catalyzed by the creation of ADD 2.5. ADD 2.5, published in 2013 [14], advocated the use of software frameworks as first class design concepts. This change was to address ADD 2.0's shortcoming of being too abstract to be easily applied by practitioners, who are accustomed to thinking about system construction by tailoring pre-existing components. ADD 2.5 starts with architectural drivers and constraints, systematically links them to design decisions and then links those decisions to implementation options available via frameworks. For agile development, ADD 3.0 promotes rapid design iterations. In addition, ADD 3.0 was paired with a "Design Concepts Catalog." This catalog includes tactics, patterns, frameworks, reference architectures, and technologies. Along with each technology is a rating of its quality attributes such as security, modifiability, availability, performance, etc.

In this iteration of the CPR process, SSV used their own experiences, prototyping, and benchmarking to generate ratings for each quality attribute of each technology in the catalog. Thus SSV's design knowledge is grown, preserved, and reused for other projects.

3.1.3 CPR Cycle 3: BDD

ADD 3.0 facilitates technology selection and agile orchestration for big data system development. However, the data modeling aspect (hence, the selection of data management components) requires additional effort. As argued in Section 1 and based on experiences in SSV, NoSQL data modeling tasks are tightly coupled with architectural design. We chose a familiar conceptual data modeling method, DFD (data flow diagram), to extend architecture scenario modeling to include data modeling for polyglot persistence, which we call BD-DFD. For BD-DFD modeling, we designed a Big Data Architecture Scenario (BDAS) template which includes 14 data elements to structure NoSQL data collection for each data scenario. In this third iteration of the CPR process, ADD 3.0 was thus integrated with extended data modeling techniques to form the BDD method, as shown in Fig. 3.

3.2 For Agile Analytics

3.2.1 CPR Cycle 1: AA

In the first CPR cycle, SSV was involved as a subcontractor in a number of projects that employed the traditional Agile

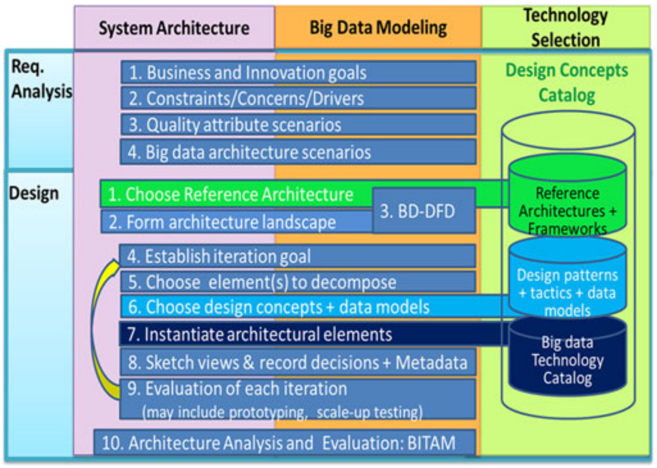
System Architecture
Big Data Modeling
Technology Selection
Req. Analysis
1. Business and Innovation goals
2. Constraints/Concerns/Drivers
3. Quality attribute scenarios
4. Big data architecture scenarios
Design Concepts Catalog
Design
1. Choose Reference Architecture
2. Form architecture landscape
3. BD-DFD
Reference Architectures + Frameworks
4. Establish iteration goal
5. Choose element(s) to decompose
6. Choose design concepts + data models
Design patterns + tactics + data models
7. Instantiate architectural elements
8. Sketch views & record decisions + Metadata
9. Evaluation of each iteration (may include prototyping, scale-up testing)
Big data Technology Catalog
10. Architecture Analysis and Evaluation: BITAM

Fig. 3. Big data design (BDD) method.

Analytics method without adequate architecture support, which resulted in issues that led to a number of lessons learned in Cases 1-4.

Lesson 1. *Need to include Data Analysts/Scientists early:* In Case 1 (Network Security, Intrusion Prevention Systems), SSV initially applied a traditional datawarehouse development methodology. The challenge was to meet performance demands of complex IP/geography analysis and queries over billions of rows to improve intrusion detection techniques. Leveraging technologies to improve system performance became a driving goal. Scrum was practiced for coping with the velocity of technology change. Following the traditional agile approach, feedback from the client included requirements changes such as more data, scalable solutions, better performance, etc. Users wanted to get rid of some static reports and create new ones. After the 1st version, the client also decided to change the deployment model, from centralized to distributed. SSV created a throw-away prototype to explore these new requirements, and finally selected the HP Vertica platform. However, the project struggled to meet its business goals. In hindsight, the issues in the early stage were caused by inefficient feedback mechanisms: if domain-savvy data analysts were involved earlier, they would have provided feedback and catalyzed changes in the system architecture.

Lesson 2. *Architecture support is required for big data analytics*. The early struggle of Cases 1-4 revealed the importance of architecture support for agile development. For example, Case 3 is one of the world's largest digital coupon web-sites, serving more than 20 million unique users monthly. In collaboration with the data science team, SSV initially implemented a traditional data warehouse. The datawarehouse architecture did not easily accommodate changing requirements. The data analytics team wanted more data, more data sources, semi-structured data, and more flexibility. The development team then created a data lake architecture using Hadoop and Hive to augment the existing datawarehouse. However, as data grew and more questions needed to be addressed, the performance of Hadoop and Hive became inadequate. This led to the next lesson learned.

Lesson 3. *Architecture-supported agile "spikes" are necessary to address rapid technology changes and emerging requirements.* SSV needed to find alternatives to Hadoop and Hive to speed up query performance. By that time, new technologies had appeared, such as Drill, Impala and Spark. SSV created a separate spike (a throw-away prototype) and selected Spark. This was a success: the data science team could then do their queries much faster.

Lesson 4. *The use of reference architectures increases architecture agility*. In a reference architecture, every "block" represents a technology family. The use of a data lake reference architecture in Case 3 supported easy swapping of different technologies within a family, e.g., changing from Hive to Spark SQL. A key part of architecture agility is the ability to change technologies within a block.

Lesson 5. *Feedback loops need to be open*. The feedback includes: (1) technical feedback about quality attribute requirements such as performance, availability, and security; and (2) business feedback about emerging requirements e.g., the business model might be changed, or new user-facing features might be needed.

3.2.2 CPR Cycle 2: AAA 1.0

Given the lessons learned from CPR Cycle 1, in Cycle 2, AAA 1.0 (Architecture-supported Agile analytics with Automation) was developed. AAA 1.0 connected to ADD 3.0, which employs reference architectures as the first step of architecture design. The creation and utilization of the *Design Concepts Catalog* reduced the cycle time for both spikes and main development. Cases 3-4 were redeveloped applying the new method, where automated testing and deployment was found to be essential to support rapid cycle time.

It was impossible to use traditional operations (that previously worked on small data) to view and interpret big data, e.g., creating 20K graphs (20 metrics x 1,000 servers) in 15 minutes for monitoring anomalies in system releases. For the WBS in our study, there were several releases a day. In addition, to scale the testing and reduce time to market, "hacking" is not feasible. The lesson learned from this CPR cycle was:

Lesson 6. *Automation must be supported by the architecture to be efficient and effective*. This motivated our next CPR cycle.

3.2.3 CPR Cycle 3: AAA 2.0

In Cycle 3, AAA 1.0 evolved into AAA 2.0 (Architecture-supported Agile analytics with Architecture-supported DevOps) and integrated with BDD to form the AABA methodology, which will be described in Section 4.

To support high availability and continuous operations, WBS must be self-monitoring, requiring architectural capabilities such as self-test, ping/echo, heartbeat, monitor, hot spares, etc. AAA 2.0 improved on AAA 1.0 by focusing on architecture support for continuous delivery [34] and, more specifically, DevOps [7]. DevOps is a set of practices intended to reduce the time between making a change to a system and the change being placed into normal production, while ensuring high quality [7], blurring the distinction between "Development" and "Operations". While DevOps is not inherently tied to architectural practices, if architects do not consider DevOps as they design, build and evolve the system, then critical activities such as continuous build integration, automated test execution, and operational support will be more challenging, more error-prone, and less efficient. For example, a tightly coupled architecture can become a barrier to continuous integration because small changes require a rebuild of the entire system, which limits the number of possible builds in a day. To fully automate testing the system needs to provide architectural (system-wide) test capabilities such as interfaces to record, playback, and control system state. Cases 7-10, and many others [9], validated that (*Lesson 7) an architectural approach to DevOps was critical to achieving strategic control over continuous delivery goals*.

4 The AABA Methodology

Our resulting AABA methodology (as depicted in Fig. 2) distinguishes itself from traditional agile analytics through the central role of software architecture as a key *enabler* of agility. AABA offers the BDD method for building the foundation of big data analytics systems and the AAA method for agile development of WBS. AABA provides a basis for reasoning about design tradeoffs, for value discovery, for planning and estimating cost and schedule, for supporting experimentation, and for supporting DevOps and continuous, rapid delivery of value.

AABA differs from other system development methodologies in that it begins with a value discovery process employing tight collaboration between data scientists (as well as business analysts, operation managers, product managers, customers, suppliers, etc.) and a software architect (as well as database designers, application developers, QA, operation engineers, etc.). These groups come together to determine the value proposition (including what analytics and metrics need to be used) for the system being built, based on their business and innovation goals.

Value discovery encompasses innovation processes, use case development, and strategic deployment planning, including cost-benefit analysis, sourcing decisions and talent management. In this Value Discovery phase the key functional and quality attribute requirements are elicited through architecture scenarios. These, along with system constraints, are the inputs that drive architecture design using the BDD method. BDD includes futuring scenario generation techniques of the Eco-Arch method [19] for new design thinking for discovering value—opportunities for innovation.

The BDD method, as shown in Fig. 3, is an iterative architecture design method for big data systems, based on reuse of well-known, assessed design primitives (patterns, tactics, reference architectures, and technologies) from a Design Concepts Catalog. As shown in Fig. 2, three components of the catalog are used in different BDD design steps: (1) reference architecture and frameworks; (2) design patterns, tactics and data models; and (3) a big data technology catalog. The use of a technology catalog makes architecture design simpler and more repeatable. The output of BDD is a set of architectural decisions, documented in sketches—perhaps just enough to create a skeletal architecture and begin fleshing out key use cases.

This design, or design fragment, is then implemented, tested, and deployed, with the help of DevOps technologies. By embracing DevOps, small iterations are supported and encouraged, creating an environment where agile spikes are easy to create, deploy, and test providing crucial feedback to the architect and data scientists.

Furthermore, in BDD, data-program *dependence* is prominent, reflecting the paradigm shifts and addressing 5V requirements. Architecture design, big data modeling and technology selection are integrated and performed in parallel in the Requirement Analysis (RA) and Design phases. RA formalizes inputs from the Value Discovery phase. In RA Steps 1-2, business and innovation goals as well as design constraints, concerns, and drivers are modeled. In RA Steps 3-4, big data scenarios focusing on "futuring" are generated along with quality attribute scenarios. BDD uses a scenario-based approach to motivate the design, and to elaborate ideas for innovation. The big data scenarios focus on architecturally significant requirements such as performance, availability, etc. We then record big data modeling inputs using the BDAS template for each scenario. This template captures 14 data architecture elements, including data source quality, data variety, data volume, velocity, read/write frequency, time to live, queries, OLTP or OLAP, etc. Each input has a direct implication on subsequent architecture choices, data model selection, technology selection and data access patterns.

The Design phase (Steps 1-3) starts with choosing a reference architecture, mapping scenarios to the reference architecture, optionally forming an architecture landscape [19], and sketching data flows in the landscape to form a BD-DFD context diagram. Each scenario exercises some architectural alternatives. By considering the full set of scenarios, an architectural landscape can be drawn, circumscribing the range of alternatives for system realization. Each alternative is a significant architectural decision that must be made. The alternatives may be based upon logical options (e.g., acknowledgement of messages or not), commercially available components (e.g., types of load balancers or file systems), or decisions within an architectural element (e.g., frequency of messaging).

In Design steps 4-9, a system and the BD-DFD context diagram are decomposed and modeled in detail in a number of iterations. Each design iteration will establish iteration goals (Step 4) and then select a driver or a system component to focus on (Step 5). The selection of design concepts (patterns, tactics, components, and frameworks from the Catalog) and selection of data models (based on inputs captured in the BDAS template) are performed simultaneously in Step 6. In Step 7, the instantiation of architectural elements includes selection of databases, analytics platforms and other system components. The technologies chosen to realize conceptual components are critical architectural decisions. This step is assisted by the big data technology catalog that links quality attributes to design choices. In some cases, Steps 6 and 7 can only be achieved via experimentation, e.g., a spike.

A spike, in agile methodologies is a miniature project, typically driven by a user story, aimed at answering a specific question. An architecture spike is similar, but is driven by a quality attribute question, typically regarding performance but it could, in theory, address any system quality [6]. In an architecture spike, a prototype (throwaway or not) is commonly created to address the quality attribute question [23]. It may be developed on the main branch, but

TABLE 3
Operational Excellence Qualities by Use Case

ID	Quality Attribute	Big Data Scenario
QA−1	Performance UC-1, 2, 5	The system shall collect up to 15000 events/sec from ~300 web servers
QA−2	Performance UC-1	The system shall automatically refresh the real-time monitoring dashboard for on-duty Operations staff with < 1 min latency
QA−3	Performance UC-2	The system shall provide real-time search queries for emergency troubleshooting with < 10 seconds query execution time, for the last 2 weeks of data
QA−4	Performance UC-3, 6	The system shall provide near-real time static reports with per minute aggregation for business users with < 15 min latency, < 5 seconds report load
QA−5	Performance UC-4	The system shall provide ad-hoc SQL-like human-time queries for raw and aggregated historical data, with < 2 min query execution time, < 1 hour latency
QA−6	Scalability UC-2	The system shall store raw data for the last 2 weeks available for emergency troubleshooting (via full-text search through logs)
QA−7	Scalability UC-4	The system shall store raw data for the last 60 days (~1 TB of raw data per day, ~60 TB in total)
QA−8	Scalability UC-3, 4, 6	The system shall store per minute aggregated data for 1 year (~40 TB) and per hour aggregated data for 10 years (~50 TB)
QA−9	Extensibility UC-1, 2, 5	The system shall support adding new data sources by just updating a configuration, with no interruption of ongoing data collection
QA−10	Availability UC-1, 2, 3, 4, 5, 6	The system shall continue operating with no downtime if any single node or component fails
QA−11	Deployability UC-1, 2, 3, 4, 5, 6	The system deployment procedure shall be fully automated and support a number of environments: development, test and production

successful architecture spikes are typically developed on a separate branch and (if successful) merged with the main branch.

BDD follows ADD's emphasis on lightweight documentation of design decisions as sketches. This is crucial for consistency and communication in distributed development. For example, in Step 8, metadata is determined as part of the design documentation. In each iteration the design

TABLE 4
Operational Excellence Design Constraints

ID	Constraint
CON-1	The system shall be composed primarily of open source technologies (for cost reasons). For those components where value/cost of using proprietary technology is much higher then proprietary technology should be used.
CON-2	The system shall use the corporate BI tool with a SQL interface for static reports (e.g., MicroStrategy, QlikView, Tableau)
CON-3	The system shall support two deployment environments: Private Cloud (with VMware vSphere Hypervisor) and Public Cloud (Amazon Web Services). Technology decisions should be made to keep deployment as vendor-agnostic as possible.

decisions made will be quickly evaluated based on the iteration goals in Step 9 and measurements taken from any deployed prototype. Once the design iterations are complete, a more complete analysis may be performed on the architecture to discover risks and identify design tradeoffs before full implementation. In BDD, we employ the BITAM method [17]—a light-weight architecture analysis method based on the ATAM [6]—to perform business and IT alignment assessment.

5 11TH CASE STUDY: ILLUSTRATING AABA

The 10 cases utilized in the CPR research are outlined in Table 5. To give a further validation and a concrete understanding of AABA for WBS, we now present a simplified version of Case 11, to illustrate how AABA can be applied to a new WBS big data analytics design to effectively achieve business goals and architectural agility. Some facts were altered to protect the identity of the client while retaining the spirit of the case. Note that we only chose one business goal, operational excellence, for this illustration of a design iteration. In reality many design iterations were performed in Case 11.

The company is a large travel advising site, hereinafter called LTA. Users visit LTA.com to research vacation options, book hotels, flights, vacation rentals, restaurants, etc., and they post reviews and ratings. LTA retains user generated content of over 150 million reviews and user feedback from over 30 countries. Their site involves more than 500 K hotels and 200 K attractions. It has more than 200 million unique visitors per month and the big data system has to handle about 1 million hits per minute.

As the data grew, they realized that existing in-house systems could no longer process the required log data volume and velocity. Moreover, new requests were coming from stakeholders such as managers and data scientists, who wanted to leverage data collected from multiple data sources, not just logs, to develop new analytics and discover new opportunities.

Following AABA, LTA started with a value discovery process. More than 30 stakeholders were identified, including B2B and B2C customers, reviewers, business analysts, data scientists, operation managers, LOB managers, collaborators, engineering teams, etc. Business innovation goals

TABLE 5
10 Big Data Analytics Web-Based Systems at SSV

Case #	Business goals	Start	Big data	Technologies	Analytics
1 **Network Security, Intrusion Prevention** MNC IT corp. Revenue > US∃ 100 billion	• Provide ability for security analysts to improve intrusion detection techniques; • Observe traffic behavior and make infrastructure adjustments: • Adjust company security policies	Late 2010-2011	• Machine generated data - 7.5BLN event records per day collected from IPS devices • Near real-time reporting touch Billion rows < 1 min.	• ETL - Talend • Storage/DW – InfoBright EE, HP Vertica • OLAP – Pentaho Mondrian • BI – JasperServer Pro	Descriptive Analytics: • Interactive Reports • Self Service Ad-Hoc Analysis
2 **Anti-Spam Network Security System** MNC IT corp. Revenue > US∃ 50 billion	• Validation of the new developed set of anti-spam rules against the large training set of known emails • Detection of the best anti-spam rules in terms of performance and efficacy	2012-2013	• 20K Anti-spam rules • 5M email training set • 100+ Nodes in Hadoop Clusters	• Vanilla Apache Hadoop (HDFS,MapReduce,Oozie, Zookeeper) • Perl/Python • SpamAssassin • Perceptron	Predictive Analytics: • Machine Learning
3 **Online Coupon Web Analytics Platform** MNC Revenue > US∃200M	• In-house Web Analytics Platform for Conversion Funnel Analysis, marketing campaign optimization, user behavior analytics • Clickstream analytics, platform feature usage analysis	2012, Ongoing	• 500 million visits a year • 25TB+ HP Vertica Data Warehouse • 50TB+ Hadoop Cluster • Near-Real time analytics (15 minutes is supported for clickstream data)	• Data Lake - (Amazon EMR) /Hive/Hue/ MapReduce/Flume/ Spark • DW: HP Vertica, MySQL • ETL/Data Integration – custom using python • BI: R, Mahout, Tableau	Descriptive and What-if Analytics: • Exploratory Data Analysis • Interactive Reports • Self Service Ad-Hoc Analysis
4 **Social Marketing Analytical Platform** MNC Internet marketing Revenue > US∃ 100 M	• Build in-house Analytics Platform for ROI measurement and performance analysis of every product and feature by the e-commerce platform; • Provide analysis on end-users interaction with service content, products, & features	2012, ongoing	• Volume - 45 TB • Sources - JSON • Throughput - > 20K/sec • Latency (1 hour – for static/pre-defined reports /real-time for streaming data)	• Lambda architecture • Amazon AWS, S3 • Apache Kafka, Storm • Hadoop - CDH 5, HDFS (raw data), MapReduce), Cloudera Manager, Oozie, Zookeper • HBase (2 clusters: batch views, streaming data)	Descriptive and Predictive Analytics: • Customer retention prediction • Exploratory Data Analysis • Interactive Reports
5 **Cloud-based Mobile App Development Platform** private Internet Co. Funding > US∃100M	• Provide visual environment for building custom mobile applications • Charge customers by usage • Analysis of platform feature usage by end-users and platform optimization	2013-2014	• Data Volume > 10 TB • Sources: JSON • Data Throughput > 10K/sec • Analytics - self-service, pre-defined reports, ad-hoc • Data Latency – 2 min	• Middleware: RabbitMQ, Amazon SQS, Celery • DB: Amazon Redshift, RDS, S3 • Jaspersoft • Elastic Beanstalk • Integration: Python	Descriptive Analytics: • Interactive Reports • Self Service Ad-Hoc Analysis
6 **Telecom E-tailing platform** mobile phone retailer Revenue > US∃ 1B	• Build an OMNI-Channel platform to improve sales and operations • Analyze all enterprise data from multiple sources for real-time recommendation and cross/up sales	End of 2013, (only discovery)	• Analytics on 90+ TB (30+ TB structured, 60+ TB unstructured and semi-structured data) • Elasticity: through SDE principles	• Hadoop (HDFS, Hive, HBase) • Cassandra • HP Vertica/Teradata • Microstrategy/Tableau	Descriptive and Predictive Analytics: • Recommender System • Exploratory Data Analysis • Interactive Reports
7 **Social Relationship Marketing Platform** private Internet Co. Funding > US∃100M	• Build social relationship platform that allows enterprise brands and organizations to manage, monitor, and measure their social media programs • Build an Analytics module to analyze and measure results.	2013 ongoing (redesign 2009 system)	• > one billion social connections across 84 countries • 650 million pieces of social content per day • MySQL (∼ 11 Tb) Cassandra (∼ 6Tb), ETL (> 8Tb per day)	• Cassandra • MySQL 5.6/5.1 • Elasticsearch • SaaS BI Platform - GoodData • Clover ETL, custom in Java, • PHP, Amazon S3, Amazon SQS • RabbitMQ	Descriptive and Predictive Analytics: • Sentiment analysis • Interactive Dashboard and Reports
8 **Web Analytics & Marketing Optimization** MNC IT co. Revenue > US∃ 50B	• Optimization of all web, mobile, and social channels • Optimization of recommendations for each visitor • High return on online marketing investments	2014, Ongoing (Redesign 2006-2010 system)	• Data Volume > 1 PB • 5-10 GB per customer/ day • Data sources – click-stream data, webserver logs	• Vanilla Apache Hadoop (HDFS, MapReduce, Oozie, Zookeeper) • Hadoop/HBase • Aster Data • Oracle • Java/Flex/JavaScript	Descriptive Analytics: • Exploratory Data Analysis • Interactive Reports • Self Service Ad-Hoc Analysis
9 **Network Monitoring & Management Platform** OSS vendor Revenue > US∃ 20M	• Build tool to monitor network availability, performance, events and configuration. • Integrate data storage and collection processes with one web-based user interface. • IT as a service	2014, Ongoing (Redesign 2006 system)	• Collect data in large datacenters (each: gigabytes to terabytes) • Real-time data analysis and monitoring (< 1 minute) • Types of devices: hundreds	• MySQL • RRDtool • HBase • Elasticsearch	Descriptive Analytics: • Interactive Dashboard and Reports • Self Service Ad-Hoc Analysis
10 **Healthcare Insurance Operation Intelligence** health plan provider Revenue > US∃10B	• Operation cost optimization for 3.4 million members • Track anomaly cases (e.g., control schedule 1 and 2 drugs, refill status control) • Collaboration tool between 65,000 providers (delegation, messaging, reassignment)	2014, (Phase 1: 8 month,) ongoing	• Velocity: 10K+ events per second • Complex Event Processing - pattern detection, enrichment, projection, aggregation, join • High scalability, High-availability , fault-tolerance	• AWS VPC • Apache Mesos, Apache Marathon, Chronus • Cassandra • Apache Storm • ELK (Elasticsearch, Logstash, Kibana) • Netflix Exhibitor • Chef	Predictive Analytics: • Complex Event Processing

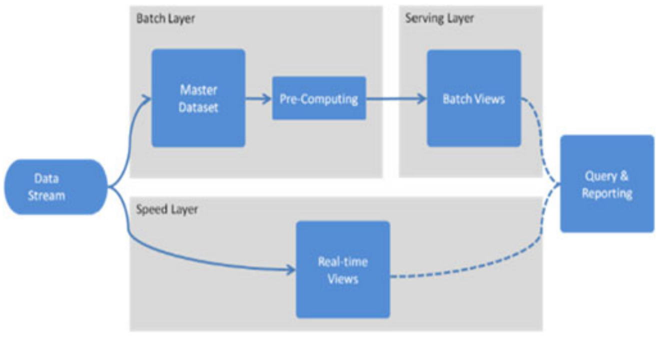

Fig. 4. Lambda architecture.

were brainstormed and prioritized among the stakeholders. The top five innovation goals for the new systems were: (1) operational excellence, (2) revenue growth, (3) reduce brand risk (improve reputation), (4) service innovation, and (5) time to market. Use cases to support each goal were generated and brainstormed, focusing on big data and analytics scenarios.

For example, for revenue growth, a use case (G2-UC1) for Click Through (CT) rate analytics was generated. Part of LTA's revenue is through a subscription service in which B2B clients, such as hotels, pay for the right to display links to their own websites and other data within their profile pages on LTA.com. Whether a hotel renews its subscription is largely determined by the number of CTs and bookings it gets. Big data analytics can help increase revenue by (1) predicting how many more CTs are required for a given client to renew; (2) calculating how much marketing will be required to reach the number of CTs for that client to renew; and (3) Determining if it makes economic sense to do so. For some clients, LTA found that even a significant increase in CT rates only marginally increased its likelihood to renew. For others, even a modest increase in CTs dramatically impacted the odds of renewing. Knowing this, LTA could focus its marketing budget on the latter and not the former.

Another example: To reduce brand risk, a use case (G3-UC1) of verifying reviews through cross-checking was generated. This included matching payment of hotel stays with reviews. Reviewers were linked with previous reviews at the LTA site and their reviews in other sites. Abnormal behaviors, such as reviews of five hotels in a short time period in different countries, could be *discovered* by big data analytics to identify likely fraudulent reviews.

Many use cases were generated for each goal, discussed, refined, and prioritized. Each use case is modeled as a quality attribute scenario which includes 6 elements: source, stimulus, environment, artifacts, response, and response measure. To illustrate the architecture-centric, quality attribute-driven design, we focus on Goal-1, operational excellence for WBS. This goal is illustrated with the following six use cases:

- UC-1: Monitor online services,
- UC-2: Troubleshoot online service issues,
- UC-3: Provide management reports,
- UC-4: Support data analytics,
- UC-5: Anomaly detection,
- UC-6: Provide security reports.

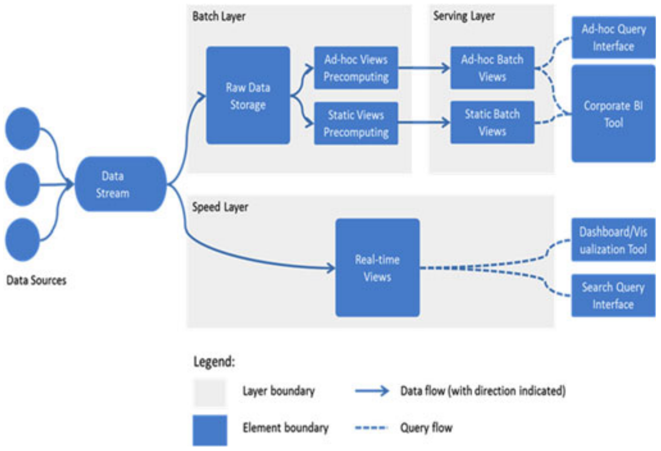

Fig. 5. Lambda architecture instantiated: 1st iteration.

There were 11 quality attribute requirements developed with these 6 use cases (see Table 3) and three design constraints (see Table 4).

Given these inputs, architecture design begins by choosing a reference architecture from the Design Concepts Catalog and mapping scenarios to the chosen architecture. Many alternatives were considered and the Lambda Architecture was chosen (see Fig. 4). It supports access to both real-time data (UC-1, UC-2, UC-5) and historical data (UC-3, UC-4, UC 6). The batch layer is based on non-relational techniques, and the speed layer is based on streaming techniques to support real-time processing [40]. These layers provide "complexity isolation", meaning that design decisions and development of each can be done independently, which increases fault-tolerance and scalability [6].

This reference architecture is instantiated through several design iterations. In the first iteration, there are three major design decisions, as shown in Fig. 5. The design splits the Query & Reporting element into sub-elements associated with the primary use cases. The drivers considered for this decomposition are:

- Ad-hoc Query Interface (UC-4, QA-5).
- Corporate BI Tool (UC-3, QA-4, CON-2).
- Dashboard/Visualization Tool (UC-1, QA-2).
- Search Query Interface (UC-2, QA-3).

The reason behind this decision is to have flexibility in selecting appropriate technologies; there could not be one "universal" tool to satisfy all of these use cases, constraints, and quality attributes. Thus, we choose to separate concerns which provides more design options. Another difference from the "classic" Lambda Architecture is that we may not need to merge together the results of queries: according to our use cases they can be executed independently for batch and real-time views.

The selected reference architecture offers an initial decomposition and data flow that significantly saves design time and effort. In the second iteration, further design decisions will be made to select candidate technologies and analyze how use cases and quality attributes will be supported. The design concepts used in this iteration are externally developed components. Initially, technology families are selected and associated with the elements to be refined. A technology family represents a group of technologies with

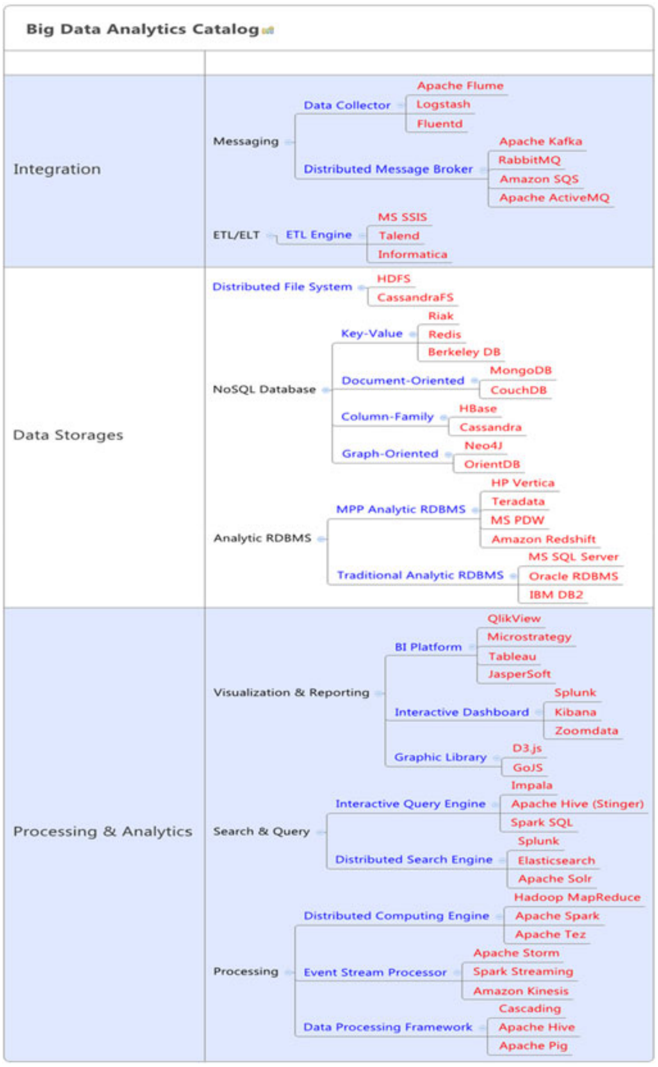

Fig. 6. Big data analytics catalog.

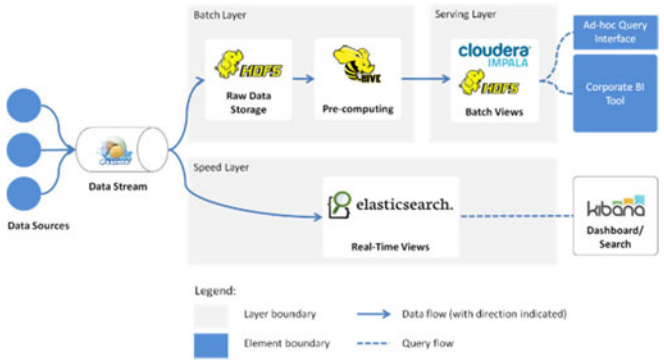

Fig. 7. Lambda architecture instantiated with technology choices: 2nd iteration.

common functional purposes. Some specific technologies may belong to several families at the same time, but having such a classification helps to make rational design decisions that result in less rework and better readiness for changes. Software components emerge, evolve, and die much faster than the patterns and principles represented by their families. The tree in Fig. 6 illustrates big data family groups, technology families and their associated specific technologies.

Six design decisions were made to select technology families: (1) Select the Data Collector family for the Data Stream element: Data Collector is a technology family (and architectural pattern) that collects, aggregates and transfers log data for later use. Usually Data Collector implementations offer out-of-the-box plug-ins for integrating with popular event sources and destinations. In our case, the system must handle Apache Web Server logs, which could be accessed either by tailing log files or by forwarding logs through IP. The destination sources are the Raw Data Storage and Real-Time Views elements.

However, real-time performance and scalability criteria $(QA-1, QA-2)$ will be difficult to meet. The distributed message broker technology family can be applied to implement the Data Stream element (and Apache Kafka is a good candidate); however, extensibility concerns related to $QA-9$ excluded this option. This shows that this step is not strictly sequentially followed by technology selection; they can be considered simultaneously.

A similar design decision process, comparing alternatives, was applied to (2) Select the Distributed File System family for the Raw Data Storage element, (3) Select Distributed File System and Interactive Query Engine families for both the Static and Ad-hoc Batch Views elements, (4) Select Distributed Search Engine for the Real-Time Views elements, (5) Use a Data Processing Framework for the Views Precomputing elements, and (6) Automate deployment of the system with Puppet scripts.

In the next step, *instantiation* is performed by choosing specific technologies from the selected families: (1) Use Apache Flume from the Data Collector family for the Data Stream element as it supports $QA-9$ (adding new data sources by just updating a configuration at run-time). We did not choose Apache Kafka because it had a significantly smaller ecosystem than Apache Flume (at the time of design, 2014), which means additional programming is needed to connect data sources and support $QA-9$.

As shown in Fig. 7, other instantiation decisions made in this iteration include: (2) Use HDFS from the Distributed File System family for the Raw Data Storage element, (3) Use HDFS from the Distributed File System family and Impala from the Interactive Query Engine family for the Static and Ad-hoc Batch Views elements, (4) Use Hive from the Data Processing Framework for the Views Precomputing elements, and (5) Use Elasticsearch from the Distributed Search Engine family for the Real-Time Views elements. As an example of our decision process, we selected Elasticsearch because it also provides a visualization tool: an interactive dashboard called Kibana. Although Kibana is a simple dashboard without role-based security, it satisfies use cases UC-1, 2 and $QA-2$ (auto refresh dashboard with < 1 minute period). Splunk also provides indexing and visualization capabilities (with more features than Elasticsearch and Kibana); however, CON-1 drives us to an open source solution.

The candidate technology choices were made based on the 11 quality attribute scenarios and 3 constraints, as shown above. And many *architecture spikes* were used to validate the performance of selected components. For instance, Hive and Impala provide SQL-like languages, thus utilizing the skills of existing datawarehouse designers, satisfying CON-2.

However, we needed to ensure that QA−4 (<15 min latency) was satisfied by creating a proof-of-concept in an architecture spike. Since the big data field is young and technologies are rapidly evolving, proofs-of-concepts are necessary to mitigate technology risks (e.g., slow performance, unsatisfactory reliability, limitations of claimed features) and to have the option to switch to an alternative early, to save overall time and budget due to avoided rework.

For instance, we performed another design iteration to address concerns that were introduced by the selection of Apache Flume in the Data Stream Element. These design iterations and architecture spikes for prototyping continued until all the quality attribute goals and design constraints were addressed. Every WBS has unique use cases, quality attribute goals, and constraints, and so the selection of technologies may be different for each. But the LTA case study illustrates how easily future technologies can be considered, assessed, and orchestrated by employing the AABA methodology.

6 Discussion and Limitations

Although we employed a relatively large number of cases in our study to increase methodological rigor, the generalizability of our results may be limited due to the inherent nature of the case study methodology. These limitations include:

1. Of the cases utilized in the CPR study, 50 percent of the companies are large enterprises (>5000 employees and annual revenue >US $1 billion). The other 50 percent are successful Internet companies with annual revenues in the range between US $20 million to US $400 million.
2. The 10 cases utilized in the CPR cycles and the LTA case are all outsourced big data projects, which are subject to contractual agreements that have different characteristics from purely internal projects, such as relatively well-defined project scope and strict(er) budget and schedule constraints.
3. The practice team we worked with are all highly trained personnel of SSV. They are also dominantly male (1 out of 30 is female) and young (average age < 30). Their evaluation of AABA and BDD might differ from novices or designers with many years of traditional data system design practice.
4. The embedded case studies are all early adopters of big data. All 10 client companies have reported increased revenue after the deployment of their big data projects. This correlation invites two possible explanations: (1) early adopters of big data technology are more risk-taking and innovative, hence will always take advantage of new technology to increase their competitive advantage; (2) successful big data deployment indeed achieves their business goals, resulting in increased revenue.
5. A design method or development methodology, no matter how thorough, can never guarantee success. The application of an architecture-centric methodology like AABA requires discipline and creativity, which may be a tall order for organizations that do not have the required discipline and innovation mindset.

7 Conclusions

Traditional structured, batch-oriented, relational "small" data systems development methods are inadequate for big data (and analytics) system development. Our AABA methodology, filling a methodological void, addresses both technical and organizational issues as well as rapid technology change challenges of agile WBS big data analytics development. It distinguishes itself from traditional datawarehouse-based agile analytics through the central role of software architecture as a key enabler of agility. AABA was developed, evolved and validated *simultaneously* in 10 WBS case studies through 3 CPR cycles.

The CPR research methodology balances relevance and rigor, and our large number of cases in particular increases methodological rigor. Currently, AABA is being employed by SSV for new big data projects in addition to the 10 WBS cases utilized in our CPR cycles. The 11th case used to illustrate the AABA methodology in this article is a new successful case. All 11 case studies showed that architecture-centric design, development, and operation is key to taming technical complexity and achieving agility necessary for successful WBS big data development.

Our contributions of the AABA methodology include (1) an architecture-centric big data design method, called BDD, for big data analytics system development and (2) an architecture-centric agile analytics method with DevOps, called AAA, for rapid big data value *delivery*.

BDD systematically integrates "futuring" techniques, quality attribute-driven architecture design and analysis, and polyglot NoSQL data modeling techniques to address the challenges from the 5Vs of big data. The explicit use of reference architectures and a Design Concepts Catalog is an advancement to architecture design methods and has proven to mitigate much of the complexity of technology selection and orchestration.

AAA is different from previous agile analytics for small data systems in that it has architecture support for agile practices. Architecture agility, through integration with BDD, has proven to be critical to the success of SSV's WBS big data analytics projects. Agile architecture practices help to tame project complexity, narrowing the cone of uncertainty [11] and hence reducing project risk. A reference architecture defines families of technology components and their relationships. This guides integration and guides where abstraction should be built into the architecture, to help reduce re-work when a new technology (from within a family) replaces an existing one. Agile spikes allow prototypes to be built quickly and to "fail fast", guiding the eventual selection of technologies to be included on the main development branch.

We presented a simplified version of our 11th case study to demonstrate the capabilities, processes, techniques, and tools of the AABA methodology. As an example for the use of reference architectures, we detailed how the Lambda architecture was instantiated through several design iterations to achieve operational excellence quality attributes of performance, scalability, extensibility, availability and deployability. We also demonstrated tradeoff decisions made throughout the design process, considering use cases and constraints. Using this case, we also illustrated the criticality of architectural principles for guiding technology orchestration and agile spikes, addressing the uncertainty surrounding new technologies.

AABA provides a commercially validated basis for reasoning about tradeoffs, for big data value discovery, for planning and estimating cost and schedule, for supporting experimentation, and for supporting DevOps for rapid, continuous delivery of value. While further improvement of AABA is ongoing, our research has contributed a first big step toward effective agile big data analytics development for WBS.

REFERENCES

[1] M. Abbott and M. Fisher. *The Art of Scalability, Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*. Reading, MA, USA: Addison-Wesley, 2009.

[2] P. Abrahamsson, M. A. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" *IEEE Softw.*, vol. 27, no. 2, pp. 16–22, Mar./Apr. 2010.

[3] R. Agarwal and V. Dhar, "Big data, data science, and analytics: The opportunity and challenge for IS research information systems research," *Inform. Syst. Res.*, vol. 25, no. 3, pp. 443–448, 2014.

[4] J. Asundi, R. Kazman, and M. Klein, "Using economic considerations to choose amongst architecture design alternatives," Softw. Eng. Inst., Carnegie Mellon University, Tech. Rep. CMU/SEI-2001-TR-035, 2001.

[5] K. Beck, et al., "Manifesto for agile software development," Retrieved 6/21/2015 from http://agilemanifesto.org

[6] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. 3rd ed., Reading, MA, USA: Addison-Wesley, 2012.

[7] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect's Perspective*. Reading, MA, USA: Addison-Wesley, 2015.

[8] S. Bellomo, I. Gorton, and R. Kazman, "Towards agile architecture: Insights from 15 years of ATAM data," *IEEE Softw.*, vol. 32, no. 5, pp. 38–45, Sep./Oct. 2015.

[9] S. Bellomo, R. Kazman, N. Ernst, and R. Nord, "Toward design decisions to enable deployability; Empirical study of three projects reaching for the continuous-delivery holy grail," in *Proc. 44st Int. Workshop Dependability Secur. Syst. Oper.*, 2014, pp. 702–707.

[10] S. Bellomo, R. Nord, and I. Ozkaya, "A study of enabling factors for rapid fielding: Combined practices to balance speed and stability," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 982–991.

[11] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.

[12] B. Boehm and R. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Reading, MA, USA: Addison-Wesley, 2004.

[13] E. Brewer, "CAP twelve years later: How the "Rules" have changed," *IEEE Comput.*, vol. 45, no. 2, pp. 23–29, Feb. 2012.

[14] H. Cervantes, P. Velasco, and R. Kazman, "A principled way of using frameworks in architectural design," *IEEE Softw.*, vol. 3, no. 2, pp. 46–53, Mar./Apr. 2013.

[15] H. Cervantes and R. Kazman, *Designing Software Architectures: A Practical Approach*. Reading, MA, USA: Addison-Wesley, 2016.

[16] H. Chen, R. Chiang, and V. Storey, "Business intelligence and analytics: From big data to big impact," *MIS Quarterly*, vol. 36, no. 4, pp. 1165–1188, 2012.

[17] H.-M. Chen, R. Kazman, and A. Garg, "Managing misalignments between business and IT architectures: A BITAM approach," *J. Sci. Comput. Program.*, vol. 57, no. 1, pp. 5–26, 2005.

[18] H.-M. Chen, R. Kazman, and O. Perry. "From software architecture analysis to service engineering: An empirical study of methodology development for enterprise SOA implementation." *IEEE Trans. Services Comput.*, vol. 3, no. 2, pp. 145–160, Apr.-Jun. 2010.

[19] H.-M. Chen and R. Kazman, "Architecting for ultra large scale green IS," in *Proc. 1st Int. Workshop Green Sustainable Softw.*, Jun. 2012, pp. 69–75.

[20] H.-M. Chen, R. Kazman, S. Haziyev, and O. Hrytsay, "Big data system development: An embedded case study with a global outsourcing firm," in *Proc. IEEE/ACM 1st Int. Workshop Big Data Softw. Eng.*, May 2015, pp. 44–50.

[21] H.-M. Chen, R. Schütz, R. Kazman, and F. Matthes, "Amazon in the Air: Innovating with big data at Lufthansa," in *Proc. 49th Hawaii Int. Conf. Syst. Sci.*, Jan. 2016, pp. 5096–5105.

[22] H.-M. Chen, R. Kazman, and F. Matthes, "Demystifying big data adoption: Beyond IT fashion and relative advantage," in *Proc. DIGIT*, 2015, http://aisel.aisnet.org/digit2015/

[23] H.-M. Chen, R. Kazman, and S. Haziyev, "Strategic prototyping for developing big data systems," *IEEE Softw.*, vol. 33, no. 3, pp. 36–43, May/Jun. 2016.

[24] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. Reading, MA, USA: Addison-Wesley, 2001.

[25] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*. Reading, MA, USA: Addison-Wesley, 2004.

[26] K. Collier, *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing*. Reading, MA, USA: Addison-Wesley, 2011.

[27] M. Conway, "How do committees invent?" *Datamation*, vol. 14, no. 5, pp. 28–31, 1968.

[28] T. Davenport, P. Barth, and R. Bean, "How big data is different," *Harvard Bus. Rev.*, vol. 90, no. 10, pp. 78–83, 2012.

[29] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inform. Softw. Technol.*, vol. 50, no. 9–10, pp. 833–859, Aug. 2008.

[30] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 6th ed. Reading, MA, USA: Addison Wesley, 2010.

[31] Gartner. Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business (2014) [Online]. Available: http://www.gartner.com/newsroom/id/2819918

[32] T.C.N. Graham, R. Kazman, and C. Walmsley, "Agility and experimentation: Practical techniques for resolving architectural tradeoffs," in *Proc. 29th Int. Conf. Softw. Eng.*, Minneapolis, MN, May 2007.

[33] S. Gregor and A. Hevner, "Positioning and presenting design science research for maximum impact," *MIS Quarterly*, vol. 37, no. 2, pp. 337–355, 2013.

[34] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Reading, MA, USA: Addison-Wesley, 2010.

[35] Infochimps. CIOs and big data: What your IT team wants you to know (2015) [Online]. Available: http://www.infochimps.com/resources/report-cios-big-data-what-your-it-team-wants-you-to-know-6

[36] H. V. Jagadish, et al., "Big data and its technical challenges," *Commun. ACM*, vol. 57, no. 7, pp. 86–94, 2014.

[37] J. Kelly. Big data vendor revenue and market forecast 2013-2017 (2014) [Online]. Available: http://wikibon.org/wiki/v/Big_Data_Vendor_Revenue_and_Market_Forecast_2013–2017

[38] J. Livari and N. Livari, "The relationship between organizational culture and the deployment of agile methods," *Inform. Softw. Technol.*, vol. 53, no. 5, pp. 509–520, May 2011.

[39] L. Mathiasen, "Collaborative practice research," *Inform. Technol. People*, vol. 14, no. 4, pp. 321–345, 2012.

[40] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Greenwich, London. U.K.: Manning Publications, 2013.

[41] L. Mathiasen, "Collaborative practice research," *Inform. Technol. People*, vol. 14, no. 4, pp. 321–345, 2012.

[42] R. Phaal, C. J. P. Farrukh, and D. R. Probert. "Technology roadmapping—A planning framework for evolution and revolution," *Technol. Forecast. Soc. Change*, vol. 71, no. 1–2, pp. 5–26, Jan.–Feb. 2004.

[43] G. Press. 6 Predictions for the ∃125 Billion big data analytics market in 2015. (2015). [Online]. Available: http://www.forbes.com/sites/gilpress/2014/12/11/6-predictions-for-the-125-billion-big-data-analytics-market-in-2015.

[44] P. Sadalage and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. London, U. K.: Pearson, 2013.

[45] R. Yin, *Case Study Research, Design and Methods*, 5th ed. New York, NY, USA: Sage, 2009.

Hong-Mei Chen is an ITM professor at the University of Hawaii, Manoa. She has directed large-scale multi-million dollar projects in high performance telemedicine, multimedia distributed databases, Electrical Vehicle performance visualization/simulation, data mining and data warehousing. She has obtained large grants and served on various NSF review panels. She is widely published including in top-tier MIS and software engineering journals.

Rick Kazman is professor at the University of Hawaii and principal researcher at CMU's Software Engineering Institute. He has authored more than 150 peer-reviewed papers, and several books, including *Software Architecture in Practice*, *Evaluating Software Architectures: Methods and Case Studies,* and *Ultra-Large-Scale Systems: The Software Challenge of the Future*. He is a senior member of the IEEE.

Serge Haziyev is a VP of Software Architecture at SoftServe, Inc.. He has an SEI Software Architecture Professional certificate and more than 17 years of experience in enterprise-level solutions including big data, SaaS/Cloud, SOA, and carrier-grade telecommunication services. He current leads the Architecture Group of more than 60 seasoned professionals as well as designing, evaluating and modernizing large scale software solutions.