Big Data as a Service: A Neo-Metropolis Model Approach for Innovation

Hong-Mei Chen, Rick Kazman
Shidler College of Business
University Of Hawaii at Manoa
Honolulu, USA
{hmchen, kazman}@hawaii.edu

Serge Haziyev, Valentyn Kropov
SoftServe Inc.
Austin, TX, USA
{shaziyev,vkrop}@softserveinc.com

Dmitri Chtchourov
Cisco System Inc.
San Francisco, USA
dchtchou@cisco.com

Abstract

Big data as a Service (BDaaS) provides a viable alternative to circumvent many obstacles in implementing a big data strategy. Many BDaaS vendors are providing cloud platforms utilizing microservices and DevOps technologies to enable big data analytics for organizations that seek cost-effective and elastic deployments. However, existing models of BDaaS are mostly proprietary, closed-world operations and this can limit the potential for innovation. In this article, we argue for a new model called the Neo-Metropolis model—a variant of the Metropolis model—that offers an organized, coherent set of open-world innovation opportunities for vendors as well as for the platform's edge customers. We identify Neo-Metropolis model characteristics and illustrate Neo-Metropolis principles for developing BDaaS using a case study of Cisco's Intercloud Analytics platform. The implications of the Neo-Metropolis model are far beyond just BDaaS and it is foreseen to be an important model for future service platform development.

1. Introduction

Big data represents tremendous opportunities for organizations to derive new forms of analytics-driven, dynamic competitive advantage, including behavioral psychographic target marketing, real-time decision-making and customization, operational intelligence and optimization, real-time brand risk management, and game-changing innovation (crowd or design-driven). Success in big data analytics depends on having an infrastructure that orchestrates big data technology components for ingesting, processing, storing, integrating, and visualizing data from literally *everywhere*, including social media, IoT, web logs, and web-crawler information [6]. A big data system must, of course, be deployed for analytics to be performed. However, despite big data being at the peak of Gartner's 2014 hype circle of emerging technology, actual deployments are scarce as of the end of 2014 [8][9]. According to a CIO survey [13], 55% of big data projects were not completed—technical roadblocks, system complexity, and talent shortages have been cited as major reasons for these failures. The heavy up-front cost of developing big data infrastructure is also prohibitive for many companies. Big Data as a Service (BDaaS) provides a viable alternative to circumvent some of these issues.

When Google released its BDaaS—BigQuery—the company touted its "time to insight" advantage over competing, on-premise big data platforms. The company boasted that customers could upload their data and be running analytics via BigQuery in a matter of a day or two, versus weeks or more needed to build on-premise, custom Hadoop clusters. Google BDaaS thus eliminated the need to buy hardware, to hire and train dedicated staff to keep clusters up and running, and to deploy and tune Hadoop or other key software components [14].

Motivated by a US$125 billion market[19], many BDaaS vendors, following Amazon's lead, are providing cloud platforms utilizing advanced technologies such as microservices [16] and DevOps [1] [3] to enable fast delivery of big data services customized for organizations that seek cost-effective and elastic deployment. However, existing BDaaS platforms (e.g. Amazon, IBM, Google, Microsoft, SAP, HP) are mostly proprietary, closed-world operations and this can limit the potential for innovation. Given closed-world, proprietary products, an organization will have to choose and integrate different services from different vendors piecemeal to deploy a big data platform that enables the desired analysis. It is a difficult, risky, and confusing task for most organizations. For instance, once an enterprise begins loading data into proprietary system such as BigQuery and building applications on top of the platform, getting the data and analytics out again and reusing the applications with another big data platform could pose a problem [14].

Inevitably, new trends in the BDaaS market are pushing toward an open world model–what we call the *Neo-Metropolis model*—for developing BDaaS

1530-1605/16 $31.00 © 2016 IEEE
DOI 10.1109/HICSS.2016.674

platforms that integrate different open source technologies to simplify integration, ease prototyping and broaden choices in one single platform, allowing organizations to innovate while managing risk.

The Neo-Metropolis model is a variant of the Metropolis model [14]. Metropolis is the Greek word for "city." The analogy is deliberate: it describes a form of development that is more like constructing a city than a single building, a perspective called ultra-large-scale (ULS) systems [17]. The Neo-Metropolis model, metaphorically "*City in the Cloud,"* shares the basic tenets of the Metropolis model but has distinct features, offering an organized, coherent set of open-world innovation opportunities for vendors as well as for the platform's edge customers.

The implication of Neo-Metropolis BDaaS on future big data engineering in enterprises is significant. Our research question focuses on *how* Neo-Metropolis BDaaS can be developed and managed to foster innovation and productivity in a cloud platform that facilitates edge-customer innovation. The present study contributes to Information Systems (IS) and Software Engineering (SE) theories and practices by illuminating the characteristics of Neo-Metropolis model and principles for effective cloud service platform management and development. We illustrate Neo-Metropolis principles using a case study of Cisco BDaaS platform (Intercloud Analytics) development.

In what follows, we introduce the background and current market trends of BDaaS. In Section 3, we detail the characteristics, principles and innovation basis of Neo-Metropolis model, in comparison to Metropolis model. In Section 4, we present the Cisco case study. In Section 5, we discuss the implications of Neo-Metropolis BDaaS. Section 6 concludes the paper.

2. BDaaS: background and Current State

BDaaS is as an umbrella term to describe a wide variety of outsourcing of big data functions to the cloud. The BDaaS concept is not new—it is an instance of cloud solutions that have been around for the past decade. Cloud computing promises on-demand, scalable, pay-as-you-go compute and storage capacity. Compared to an in-house datacenter, the cloud eliminates large upfront IT investments and let businesses scale infrastructure, while paying only for the capacity they use.

There are three cloud service models by the NIST standards: Infrastructure-As-A-Service (IaaS), Platform-As-A-Service (PaaS), and Software-As-A-Service (SaaS) and these offer five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Current cloud deployment models include private cloud, public cloud, hybrid cloud, community cloud, multi-cloud, and inter-cloud. Providers of IaaS offer computers—physical or (more often) virtual machines—and other resources. Most existing BDaaS include offerings of basic hardware (IaaS). In the PaaS models, cloud providers deliver a computing platform, typically including virtual machines, operating systems, databases, web servers, development frameworks, transactions, programming language execution environments, applications, and services. Application developers can develop and run their software on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software. SaaS is a complete operating environment with applications, management and the user interface. Examples include DBaaS (Database as a Service), and BAaaS (Business Analytics as a Service).

There are different types of BDaaS being offered [19]. The core BDaaS implements a minimal platform, e.g., Hadoop with YARN and HDFS and a few popular services such as Hive. Amazon Web Service's (AWS) Elastic MapReduce (EMR) is currently the most prominent core BDaaS offering.

One path of vertical integration for BDaaS is downwards for PaaS to include an optimized infrastructure. This allows the elimination of some overhead of virtualization; the vendor can specifically build servers and networks that cater to big data platform performance needs. The other path of integration for BDaaS is upwards for PaaS to include features beyond the common Hadoop ecosystem offerings. Feature-driven BDaaS focuses on services and abstractions to get users started with big data quickly. Newer offerings include web and programming interfaces as well as database adapters pushing technologies like Hadoop or Spark into the background, thus reaching into the SaaS layer.

The current BDaaS market is populated with many big players and an array of new startups providing innovative solutions. Google BigQuery was launched in mid-2012, aiming for a piece of the big data cloud pie. In September 2014, Oracle introduced their Analytics Cloud: a comprehensive analytics offering in the cloud, combining business intelligence, big data analytics, and embedded SaaS analytics. In Oct., 2014. Twitter and IBM inked a deal that will see the companies jointly develop big data analytics services and applications that deliver insights using Twitter data, offered as BAaaS. In the same month, fulfilling its "cloud first, mobile first strategy," Microsoft announce its new Azure

Intelligent Systems Service, a cloud-based service to collect and manage data from the Internet of Things (IoT), and SAP embraced IBM as a "premier strategic provider" of cloud infrastructure services for its business-critical applications. The partnership between these tech titans aims to accelerate the ability to run core business functions in the cloud.

IoT back end as a service (BaaS) is also emerging, as major players—including Amazon, IBM, and Microsoft continue to stitch together a variety of PaaS offerings, including stream processing, data triggers, indexing, synchronization, and notifications, into more tightly integrated offerings directly marketed to the growing community of IoT developers.

An analysis of the survey of 100 big data startup companies [20] reveals new trends in BDaaS innovations. The trends are to provide: (1) BD analytics tools for end users; cloud-based, self-service predictive analytics platforms; (2) dashboards for visualizing data from different sources; (3) BD analytic tools for different segments: retail, human resources, IT operation, analytics for the nascent Internet-of-Things (IoT) arena; (4) cloud-based datawarehouses competing with Amazon Web Service's Redshift and Google's Big Query; (5) RDBMSs on top of Hadoop; (6) open source: Pivotal announced in February that it was open-sourcing some of its products, including the Greenplum and GemFire databases and its Pivotal HD Hadoop distribution; (7) Hadoop as a service (HaaS) running on different platforms: Amazon AWS, the Google Compute Engine and Microsoft Azure; (8) tools to simplify data preparation for analytics; (9) tools to eliminate the need for various BI products; (10) integration: The Sqrrl Enterprise database offers integrated NoSQL (column, graph and document store) capabilities; The Qubole Data Service runs on Amazon AWS, the Google Compute Engine and Microsoft Azure.

Big data system deployment is unique in that it requires orchestration of many different technologies [6]. The close-world model of leading BDaaS vendors will most likely limit innovation because they will not be able to take advantage of and keep up with integrating the afore-mentioned and other emerging new innovations into their offering (or customers will have to do the integration on their own). And switching costs, from vendor to vendor, are very high for customers.

3. Neo-Metropolis Model

The Metropolis Model [14] was originally presented as a new logic for software development and for the creation of crowd-sourced (open source and open content) systems, differing from prior software lifecycle models in that it was meant to account for two transformative trends that had been increasingly shaping our information environment: the rise of the socio-technical ecosystem and the inexorable business shift towards service orientation. These trends were the result of a radical rethinking of how information and software were created and evolved, reflecting the rise of commons-based peer production [1], which is now termed the "sharing economy." These trends have only intensified in the past 6 years since the original publication of the Metropolis Model and have broadened the relevance and applicability of the Metropolis Model. In particular, two tenets of the original Metropolis Model, combined with the increasing prominence of cloud computing – the third tenet, have propelled the new Neo-Metropolis Model: (1) the proliferation of *open source products* has reached sufficient critical mass of utility and popularity, and (2) *interoperability* technologies, such as microservices, have become sufficiently mature to allow low-cost, efficient, and predictable integration of open source applications.

The Neo-Metropolis model—metaphorically "City in the Cloud"—has similar structure but distinct differences from the original Metropolis model. We describe its structure and characteristics, in comparison to the Metropolis model, below.

3.1 Neo-Metropolis Structure

As shown in Figure 1, the original model divided a project into three realms: kernel, periphery, and masses. This division reflected different business concerns, stakeholder roles, and software functionality. The heart of the original Metropolis model is the "kernel"—a platform that provided the core functionality on which all else was built. Examples of well-known traditional kernels include Linux's kernel, Wikipedia's wiki, Android's application development platform, Facebook's platform and many others.

The kernel of a Neo-Metropolis platform serves the same purpose, but reflects an even larger scale: it is a platform comprised of systems, built by many other projects, residing on a cloud or intercloud infrastructure. The intent of the platform is to make it easy for many distinct projects—the periphery—to adopt, deploy, and scale systems that are built upon integrated open-source components. At the periphery, the participants are not individuals but projects.

These projects at the periphery all use the same platform, but they may be hosted on a private cloud. At the edge, they are not the masses –the crowds— but primarily companies which may have their own crowd.

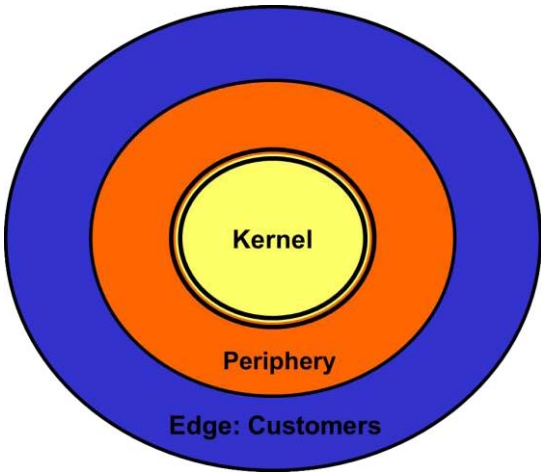

Figure 1. The Neo-Metropolis Model Structure

3.1 Neo-Metropolis Characteristics

The Metropolis model, following many of the principles of Ultra-Large Scale systems [17], enumerated eight defining characteristics. These were: Mashability; Conflicting, Unknowable Requirements; Continuous Evolution; Focus on Operations; Open Teams; Sufficient Correctness; Unstable Resources; and Emergent Behaviors [14]. We will now describe the characteristics of the Neo-Metropolis model, and explain how these differ from the original model (see Table 1).

(1) Mashability: Mashability is the ability to easily compose content and software that you may not have created is a key advantage of open source and open-content systems. Neo-Metropolis systems take mashability a step further, by providing constituent systems as services, making it relatively easy to plug them together. Indeed the trajectory of Neo-Metropolis platforms appears to be a "Lego-blocks" approach where users of the platform create systems by plugging together, configuring, and provisioning open-source components in cloud infrastructures. This service-based approach, is typically implemented as a microservices architecture.

(2) Conflicting, unknowable requirements: This does not change in a Neo-Metropolis setting. Requirements will always emerge from the periphery—in this case the various open source projects—and will always be in conflict to some extent. In the Neo-Metropolis model, the kernel may have a stronger incentive to exert influence on requirements for periphery projects.

(3) Continuous Evolution: Metropolis projects are never in a stable state -- perpetual beta. While the kernel might have traditional releases, the periphery is continually changing and thus there is no more notion of a stable state any more than there is the notion of the stable state of a city. Parts are always being constructed, modified, and torn down by the stakeholders. However, because a Neo-Metropolis model is concerned with the integration and interoperation of many open-source systems, it may impose a structure of releases on the periphery. That is, the periphery projects may evolve, each on their own timeline, but a Neo-Metropolis project will segment these into releases, where many new versions of projects are released in a synchronized fashion.

(4) Focus on operations: Metropolis model systems have had a strong focus on operations that systems are as reliable and accessible as a public utility. Neo-Metropolis also has a strong focus on operations. In fact, could services are called "the fifth utility." Operations with continuous evolution has meant that systems can change frequently, sometimes pushing multiple "versions"—each a small increment on the last—to market in a single day. This trend has again deepened, as evidenced by the rise of substantial research and tooling supporting continuous delivery [12] and DevOps [1]. DevOps, has required changes in the mindset, the architecture, and the associated tooling to support automation of configuration, integration, testing, delivery, and run-time monitoring [3].

(5) Sufficient Correctness. Perpetual beta of the periphery is the norm, but the kernel must be relatively stable and backwards compatible. And, *successful* Metropolis projects can attract a large community of both contributors and end users, placing "many eyes" on the project, which enhances correctness. This characteristic is also found in the Neo-Metropolis model, but with a slight enhancement: the kernel serves as a platform for rapid prototyping and simplified integration of individual services. Furthermore, as mentioned above, a Neo-Metropolis project observes a more traditional release process. To support this the project must adopt DevOps practices of running automated quality assurance and continuous integration test to ensure that there are no regression issues in the latest release.

(6) Scalable Resources. In a Metropolis project resources (effort and, in some cases, computational resources) are contributed by the periphery, and

hence are not guaranteed. But large numbers of contributors at the periphery ameliorate the risk of unstable resources. In a Neo-Metropolis model this characteristic is managed. The platform, hosted on a cloud or intercloud, provides scalable resources, where these resources are managed by the kernel.

(7) Gated Behaviors: The final characteristic of a Metropolis system is "emergent behaviors." Such behaviors are encouraged in Metropolis systems, as they are a source of innovation. However, they can also result in unintended and unwanted usage scenarios (e.g. large-scale coordinated behaviors of users may resemble denial of service attacks). Gated behaviors are thus the norm in Neo-Metropolis systems. Each customer has their own virtual cloud-based environment, gated off from every other customer. In each customer's community, they may encourage emergent behaviors, but the effects of such behaviors typically should not "spill over" to an adjacent customer's community.

Table 1: Metropolis vs. Neo-Metropolis Characteristics

Metropolis	Characteristics	Neo-Metropolis
Mashability	Open source Open content	Same; microservices simplifies mashability
Conflicting, Unknowable Requirements	Requirements emerge from the periphery	Same; the kernel may exert influence on the periphery
Continuous Evolution	Perpetual beta	Same: the kernel may co-ordinate releases; DevOps
Focus on Operations	as reliable and accessible as a public utility	Same: "the fifth utility"; DevOps
Sufficient Correctness	Perpetual Beta, "many eyes"	Same; DevOps
Unstable Resources	Peer-produced applications	**Scalable Resources**, managed by the kernel
Emergent Behaviors	Encouraged	**Gated Behaviors**

3.3 Neo-Metropolis Principles

As a testimony to the broad applicability of the Metropolis model, the Neo-Metropolis model inherits all original seven principles, albeit with one variation.

(1) Community Engagement and Negotiation. This is the one principle that differs. The first Metropolis model principle is "*crowd engagement and egalitarian management of open teams.*" Metropolis projects are created by peer production, where contributors are often unpaid volunteers. In the Neo-Metropolis model, the "open team" characteristic is also seen, and amplified. The platform is technology agnostic, taking advantage of contributions from many projects (packaged as microservices) which may themselves be Metropolis projects. Furthermore, the technology-agnostic nature of a Neo-Metropolis project means that the contributions of ever-larger communities of developers can be marshaled. The success of the platform hence hinges on the ability to make integration of the microservices simple, as mentioned above.

In addition, in the Neo-Metropolis model, the focus is on building an open city with many "gated communities" (businesses): each edge customer is a gated community. The periphery is an open community that includes many projects that may have their own crowds (open source or open content contributors). The kernel needs to engage the periphery community to create valuable services that in turn can engage their edge customers. In a cyclic way, the edge customers can influence the kernel, by pushing their platform requirements inward. There is a constant exploration to find "win-win" conditions—a process for value co-creation. Like building a city, the infrastructure and "zoning rules" must be in place to create the social and technical mechanisms needed to incentivize long-term participation, encourage community custodianship, reward merits of individuals or projects, and protect gated communities by enforcing security and privacy.

(2) Bifurcated requirements. Platform requirements are bifurcated into: kernel services that deliver little or no end-user value (e.g. the Facebook application platform), and periphery requirements contributed by each peer project that delivers the vast majority of end-user (edge customer) value. However, one slight difference from the Metropolis model is that interoperability requirements may be imposed by the kernel on the periphery.

(3) Bifurcated architecture. The architecture is divided into a kernel infrastructure and set of services created by (open source) peripheral projects. The kernel provides the means for achieving and monitoring key quality attributes (e.g. performance, security, and availability). The architecture of periphery components is enabled and constrained by the kernel through its primitives and compliance with its protocols; the periphery is otherwise unspecified. Each part of the periphery may have its own unique architecture. This lack of specification permits unbridled growth, innovation, and parallel creation at the periphery. Note also that the kernel does not have to be created through a Metropolis life cycle. Kernels

may be created through more conventional means, for example following an evolutionary model.

(4) Fragmented implementation. The bifurcation of the kernel and periphery has important consequences for implementation. The periphery, consisting of open communities, develops at its own pace, to its own standards, using its own tools, releasing code as it pleases. A distinct group implements the kernel, not a crowd from the open community but rather a close-knit, highly motivated, coordinated team. There is no overarching plan and little coordination of activities at the periphery.

(5) Distributed testing. Verification of the kernel differs from verification of the periphery. Though the kernel must be highly reliable, this requirement is tractable because the kernels of successful Metropolis projects are small— orders of magnitude smaller than the periphery—highly controlled, and slow to change. In the Metropolis model, the reliability of the periphery is indeterminate but the Neo-Metropolis model the kernel exercises more control over the reliability of the periphery.

(6) Distributed delivery/maintenance. Delivery and maintenance of the kernel differs dramatically from delivery and maintenance of the periphery. The kernel must be stable and when it does change must be backward compatible. At the periphery, in the Metropolis there was no notion of a stable system state and perpetual beta was the norm. In the Neo-Metropolis model, the kernel controls system state via a release process facilitated by a DevOps mindset and tooling.

(7) Ubiquitous operations. Metropolis systems are "always on," even when they are being upgraded. Complicating this mandate is the fact that upgrades are not ubiquitous; the parts of the system are on their own release cycles and operate (and interoperate) simultaneously. For Metropolis systems, operations must be a focal activity and, in particular, geared toward high availability. Also, upgrades must be backward compatible, retaining access to at least kernel functionality, since there is no assumption that all parts of the system will be upgraded at any given point in time. There is more support for ubiquitous operations in a Neo-Metropolis model due to its embrace of DevOps principles and tools.

3.4 Neo-Metropolis Innovations

Innovation is supported by the characteristics and principles of the Neo-Metropolis model. In particular, mashability, bifurcated requirements, architecture and implementation, as well as continuous operations that give the peripheral freedom to invent and combine various microservices for new applications quickly. The genesis of Neo-Metropolis innovation is straightforward:

1. **Open innovation**: Various participants from the periphery (both open source and proprietary) and the edge can interact dynamically with the kernel to generate the "collective intelligence" that potentially has more power than what can be achieved in a closed, proprietary setting.
2. **The numbers game in "Lego" innovation**: We liken Neo-Metropolis innovation as "Lego" innovation due the interoperability allows rapid mashups of services. The open world model of BDaaS offers more innovative combination opportunities than the closed-world alternative. The possible combination of a set of N elements is N!. If N is bigger, N! is factorially bigger. More Lego blocks, more possible combinations: as simple as that.

4. Cisco BDaaS on Intercloud Analytics Platform

Cisco BDaaS isthe first and currently the only commercial Neo-Metropolis development. This "unique" [22] case study focuses on the critical success factors in implementing a Neo-Metropolis BDaaS platform. The quotes below are from the two project architects.

4.1 Neo-Metropolis Vision and Competitive Advantages

Cisco offers BDaaS as a part of the Cisco's overall cloud strategy, which is to build the platform for the *Internet of Everything*. Their mission is to connect clouds (and applications) seamlessly so as to increase their customer base. "Our challenges and opportunities are: scalability and elasticity, simplicity for access, make things easy for (Windows, Apple, etc.) developers." "We don't compete directly with Amazon; our strategy is to develop technology for microservices (higher up the stack) so that it can be deployed anywhere." "Public product cloud offering is not our core business; we want to invest in the internet in general, providing the capabilities for B2B interactions, e.g., Cisco's Intercloud network."

Their Neo-Metropolis vision and also their competitive advantage lies in their "**platform, vendor agnostic" approach**: Cisco chooses, integrates, and provisions the kernel—a platform upon which customers at the periphery build their big data applications. "We recommend technologies and solutions that suit our customers best. And to provide such a customer-facing approach, we made

partnerships with 3 major Hadoop vendors (MapR, Hortonworks and Cloudera) and our customers can get one which better suits them, e.g., MapR provides a best-in-class file-system with unique replication functionality for Hadoop clusters across data centers, while Hortonworks can provide full Hadoop cluster automation after their recent acquisition of Sequence IQ and their Cloud Break product." "And even more, we're working closely with many 3rd party software companies, startups and open-source development groups to add as many useful services as we can and to make the platform as open for technologies and products as possible. For example, we work with Mesosphere to get a best-in-class clustering solution, with Platfora and Zoomdata to get modern, customer friendly and ad-hoc ready BI tools, Basho to get one of the best NoSQL databases, etc."

Figure 2. Cisco's Neo-Metropolis Vision [7]

The second competitive advantage realized from the Neo-Metropolis vision is that Cisco offers **open source solutions to build microservices automatically** across many different cloud platforms. They have developed a set of comprehensive automation scripts using technologies like Terraform to provision infrastructure (virtual machines, network, storage) and Ansible to bootstrap provisioned VM's with necessary software to run microservices cluster (based on Mesos, Marathon and Consul) and run services on top of it (such as Hadoop HDFS, Apache Spark, Kafka, Cassandra, etc.). "Core functionality of our product is open sourced from the very beginning and most of the 3rd party software we use too. This means customers can contribute into it, use it for free in most cases on premise, and so on. It also helps to keep software transparency to ensure there are no hidden issues or bugs which can lead to awkward results in the future." "Other clouds are not open enough."

Cisco is cognizant of their competitors: Oracle, IBM, SAP, Amazon, Microsoft, etc. So their open source strategy is directly aimed at this risk. It also brings good will as Cisco is seen as helping smaller companies. In what follows, we describes how Cisco employs Neo-Metropolis principles.

4.2 Realizing the Neo-Metropolis Principles

(1) Neo-Metropolis Community Engagement and Negotiation:

Cisco employs different community engagement strategies for the edge and the periphery. First, *for the edge*, BDaaS customers are initially drawn from their existing customer base. Cisco wants to help move their enterprise clients into private (managed) clouds as many customers will not accept public clouds. This is an easy migration for these customers since they already trust Cisco. However, there is still a need to prove to many clients that they will enjoy long-term benefits in terms of scalability. As a result, Cisco provides the cost/benefit analysis for these enterprise clients. It often shows that in the short run, there is a loss by giving up their existing investment but the clients always need more, and hence benefit, in the long run. Scalability is the key success factor.

Second, *for the periphery*, they draw participation from vendors of open-source products by: (1) promoting the marketing benefits that will improve vendor's footprint; (2) counting on a crowd/network effect: "Usually if Hortonworks joins, Cloudera would like to join; MapR would join too and they can provide custom support together. If 95% of vendors are there, the rest would want to join too." "It's about "being 'in' with the in crowd, following the trend"; (3) contracting with vendors to create desired features. Being adopted by Cisco is a great marketing tool for vendors; (4) being open to any BD products and cloud solutions; and (5) partnering with smaller vendors (e.g., Zoomdata) to provide more features. This is different from IBM or Oracle, which often just acquire competitors or smaller technology companies.

Moreover, if there is any vendor who does not want to cooperate due to competition or any other reasons, Cisco would leave them out of the community. Cisco wants the best technical solutions to their products but is strict about control and policy for cooperative community participation.

The key success factor for sustaining the periphery community includes technology and product roadmaps for strategic partnerships that can evolve dynamically. According to a roadmap,

potential partners are invited, evaluated, and their products tested. Some strategic partnerships are paid, others are just agreements for mutual benefit**,**

A critical implication of the Neo-Metropolis principle of community engagement is a focus on the custodianship to safe-keep the gated communities. Cisco has a vision of building dedicated clusters for big data products. For microservices, they will give endpoints to shared infrastructure (shared tenancy). But for the edge customers, services such as Hadoop as a service (HaaS), are in dedicated clusters (gated communities), with full control for the client and no shared tenancy, for security and privacy reasons. Cisco's security team are constantly scanning their cloud infrastructure and taking proactive steps to keep it secure. Being a successful network company, their strength is in network infrastructure and security. As soon as they find a virus they unplug the offending client from the infrastructure.

(2) Bifurcated Requirements:

In traditional projects, requirements are generated by user requests. In the Metropolis model, requirements emerge from the periphery. The Neo-Metropolis model merges features of the Metropolis model and traditional projects. Although the constituent projects are primarily open source and hence have their own trajectories, Cisco attempts to "exert strong influence" and persuading these open source communities. Some requirements come directly from their big customers (e.g., billion dollar contract) for B2B advanced services. Some are from their own R&D such as Cisco Lab.

(3) Bifurcated Architecture:

Cisco is using a traditional top-down, plan-driven process to create the kernel of its platform. Although they are considering open-sourcing this code-base, it is, at the moment, proprietary. As stated in section 3, this architecture not only supports and enables the periphery, but it influences the periphery by incentivizing individual projects to comply with the architecture.

(4) Fragmented Implementation:

Just as the requirements processes and architectures of the constituent projects are minimally influenced by Cisco's kernel and business processes, so too is the implementation. Each project develops using its own methods and tools, project planning, and release cycle.

(5) Distributed Testing:

Cisco not only manages the testing of its kernel, but also exerts oversight on the quality of constituent projects via automated acceptance testing. This testing is for more than just functional requirements. It includes performance and security testing (as mentioned above in safe-keeping gated community) which are also automated. In this way regression tests are automatic and require relatively little human input or oversight.

(6) Distributed Delivery/Maintenance:

At Cisco, a focus on DevOps techniques helps them to deploy microservices quickly. DevOps means automating repetitive and error-prone tasks as much as possible (e.g., build, testing, and deployment maintain consistent environments). And it means employing automated testing analysis tool as well as performance dashboards. Tools such as Apache Mesos are being routinely used to better manage and deploy resources, and to scale better.

(7) Ubiquitous Operations:

Similarly, Cisco has hired DevOps engineers instead of developers. DevOps people are writing scripts and this is the primary form of development requiring expertise in Python, Linux administration, and Docker. The goal is to automate as much of the operations as possible.

4.3 Neo-Metropolis Innovation

Following the above-mentioned Neo-Metropolis vision and principles, Cisco already has fruitful co-development with the periphery, their strategic partners. Neo-Metropolis innovation is evidenced in the new array of BDaaS products that are being developed by including everyone (i.e., by being vendor-agnostic). Components for big data applications (microservices) developed so far include Data Storage as a service (e.g., HDFS), Data Processing as a Service (e.g., MR, Spark), Data Insights as a Service (pre-processed data as Data Marts and Data Insights ready for consumption; this is still under development), and Data Visualization as a service (e.g., Zoomdata). They believe *everything* can be a service: Schema as a service, Data as a service, Analytics as a service, etc. They are creating a collection of data marts as services and making it easy for others to create new ones, moving towards the vision of a "data mall" (e.g., IoT with a collection of data marts). However, they are most interested in creating the framework for sharing datasets while less interested in creating the datasets themselves. This provides open innovation opportunities for co-creating with edge-customers. Cisco achieves innovation for BDaaS by making all the microservices readily available, easy to scale and consume for the customer, through APIs. This makes

it easy for everyone to create cloud analytics and publish to the world. DevOps helps them to deploy microservices quickly and reliably [3]. The "global identity" allows an app to know about other apps. Technologies such as Mesos are utilized to better manage resources sharing, deploy more easily, and scale better.

Cisco attributes their innovation to three factors in their development process:

First, adopting a generalist approach for executing end-to-end tasks: "since it is an infrastructure project, all of our developers are DevOps engineers and every single member in a team can do DevOps work including QA, PM and development. This helps the team to cover each other and ensure they can drive tasks to logical end without sending tasks from one engineer to other."

Second, open sourcing "passionate" developers and motivating them by meeting their needs: "We develop open-source projects so our guys are actually being paid for something they love to do for free, this makes them much more happy compared to people who write proprietary code: imagine that you can show the results of your work to anyone on GitHub (colleagues, friends, next employee). Guys realize that they contribute to something that can and will be seen and used by many people; see how many stars, forks and contribution commits we have on GitHub," referring to the website [10].

Third, close working relationships with strategic partners: "I have Hortonworks, etc. on speed dial." And Cisco provides workshops for developers. "It's like a big data family."

5. Discussion

The Cisco BDaaS case study serves to illustrate the features and principles of the Neo-Metropolis model. It has empirically validated how the Neo-Metropolis model fosters innovation within the platform which enables customers to cost-effectively innovate with big data. It also sheds light on the critical success factors in employing Neo-Metropolis principles and achieving innovation. The contributions of the present study has limitations due to the inherent nature of the case study methodology [22]. In addition, our case study focuses on the development of kernel and peripheral components of the Neo-Metropolis model and only reports on indirect evidence from edge customers. We will directly study the factors that affects how edge customers choose a BDaaS solution and how they innovate with BDaaS in our planned future research.

We are not seeking the generalizability of our model, but to describe a model that *has emerged in practice*, as a variant of the Metropolis model, due to unstoppable market forces. In doing so, we are capturing a new logic for developing future service platforms.

The fact that the Metropolis and Neo-Metropolis models embody very similar principles is a testimony to the power and generality of the original Metropolis model. However, subtle but important differences in the two models imply different consequences for design and implementation.

We foresee Neo-Metropolis BDaaS development will prevail in the future. There are several practical implications of this trend:

First, big data will not just not just for the "rich." Smaller companies that lack IT or financial resources can benefit from the scalability and low starting cost for experimenting and innovating with big data. An early analysis of four SMEs [15] found that cloud services can offer both economic and business operational value previously denied them.

Second, Neo-Metropolis BDaaS could fundamentally change big data engineering practice in enterprises: (1) Data could be rented and bought, (2) Prototyping will be economical utilizing cloud elasticity and an array of BDaaS, (3) BDaaS will alter the infrastructure and application landscape, (3) BDaaS could reduce the need for in-house expertise, or ameliorate lack of big data talent in the market, (4) Application design will shift from coding to primarily a mashing-up of BDaaS services, (5) Innovation can be achieved from both the insights from the big data and from the unique combination of BDaaS applications to create new products or new processes for achieving business goals, (6) A Neo-Metropolis platform could be used to connect with supply chain partners' services, and thus could generate service innovation in unexpected way, and (7) BDaaS offers shields from consequences of rapidly changing technology, reducing the risk and complexity of system upgrade and maintenance.

Third, Neo-Metropolis BDaaS offers flexibility and a range of choices. However, this introduces new challenges such as who, what and when to choose and how to optimize all the choices available. This calls for future research on creativity [5] and decision support for technology orchestration and service design utilizing BDaaS to align with business and innovation goals: a new kind of Business-IT alignment [3] in the Neo-Metropolis world.

7. Conclusions

The Neo-Metropolis model is a new logic for system development driven by (1) the increasing prominence of *cloud* computing, (2) the proliferation of *open source* products that have reached critical masses of utility and popularity, and (3) *interoperability* technologies, such as microservices which are sufficiently mature to allow efficient and predictable integration of open source applications.

We argue that Neo-Metropolis model is most suitable for BDaaS provision as the "sharing" economics and the open, "Lego" innovation potentials are desired by and are beneficial to both the BDaaS providers and edge customers. Not only does Neo-Metropolis BDaaS provide an effective alternative to circumvent obstacles for implementing big data strategies, but it also has significant impact on current big data engineering practice. In this paper, we have identified and illustrated the features and principles of the Neo-Metropolis model using the Cisco BDaaS case. The Cisco case we contributed is unique case as they are the first and only vendor in the current market that has implemented the emerging Neo-Metropolis model. It sheds light on the critical success factors in employing Neo-Metropolis principles and achieving innovation. We discussed how it would impact on existing big data engineering practice in enterprises. We foresee the Neo-Metropolis model to be an important model for future service platform development, going beyond BDaaS provisioning.

8. References

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

[2] Benkler, Y. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* Yale University Press, New Haven, CT, 2006.

[3] Chen, H-M., Kazman R, Haziyev, S., Kropov, V. Chtchourov, D., "Architectural Support for DevOps in a Neo-Metropolis BDaaS Platform", *Second International Workshop on Dependability and Security of System Operation (DSSO 2015)*, (Montreal, Canada), Sept. 2015.

[4] Chen, H-M., Kazman R, Garg, A. "Managing Misalignments between Business and IT Architectures: A BITAM Approach," *Journal of Science of Computer Programming*, 57(1), 2005, pp. 5-26.

[5] Chen, H-M and Kazman R. "Architecting for Ultra Large Scale Green IS," Proceedings of GREENS 2012 at the 34rd ICSE, Zurich, Switzerland, June 2-9, 2012.

[6] Chen, H-M., Kazman R., Haziyev S. Hrytsay O. "Big Data System Development: An Embedded Case Study with a Global Outsourcing Firm," Proceedings of *BIGDSE'15* at ICSE 37, May, 2015.

[7] Cisco. "A Strategic Approach to Meeting the Demand for Cloud, http://www.cisco.com/c/dam/r/en/us/internet-of-everything-ioe/assets/files/Strategic-Approach-to-meet-demand-for-cloud.pdf. Accessed May 1, 2015.

[8] Gartner. "Gartner's 2014 Hype Cycle for Emerging Technologies Maps the Journey to Digital Business," http://www.gartner.com/newsroom/id/2819918. Accessed Jan. 13, 2015.

[9] Gartner. "Survey Analysis: Big Data Investment Grows but Deployments Remain Scarce in 2014." http://www.gartner.com/document/2841519. Accesed Jan. 13, 2015.

[10] Github, "Open Source: CiscoCloud Microservices infrastructure" https://github.com/CiscoCloud/microservices-infrastructure.

[11] Hoffman, D.L. and Fodor, M. "Can You Measure the ROI of Your Social Media Marketing?" *MIT Sloan Management Review (*52:1), 2010, pp. 41-49.

[12] Humble J. and Farley D., Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley 2010

[13] Infochimps. "CIOs and Big Data: What Your IT Team Wants You to Know," http://www.infochimps.com/resources/report-cios-big-data-what-your-it-team-wants-you-to-know-6. Accessed Sep. 1, 2013.

[14] Kazman, R. and Chen, H-M. "The Metropolis Model: A New Logic for the Development of Crowdsourced Systems," *Communications of the ACM*, Volume 52, Issue 7, July 2009, pp. 76-84.

[15] Kelly, J. "Google Playing the Big Data Long Game", http://wikibon.org/blog/google-playing-the-big-data-long-game/, accessed on June 28, 2012.

[16] Lacity, M. C. and Reynolds, P. "Cloud Services Practices for Small and Medium-Sized Enterprises," MIS Quarterly Executive (13:1), 2014, pp. 31-44.

[17] Newman, S. *Building Microservices,* O'Reilly, 2015

[18] Northrop, L., Feiler, P., Gabriel, R., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., and Wallnau, K. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. SEI/CMU, Pittsburgh, PA, 2006.

[19] Press, G. "6 Predictions For The $125 Billion Big Data Analytics Market in 2015" http://www.forbes.com/sites/gilpress/2014/12/11/6-predictions-for-the-125-billion-big-data-analytics-market-in-2015/

[20] Prokopp, C. "The four types of Big Data as a Service (BDaaS)" http://www.semantikoz.com/blog/big-data-as-a-service-definition-classification/, accessed June 23, 2014.

[21] Whiting, R. "2015 Big Data 100: he Emerging Big Data Vendors", http://www.crn.com/slide-shows/data-center/300076757/2015-big-data-100-the-emerging-big-data-vendors.htm, accessed on May 7, 2015.

[22] Yin, R. K. *Case study research, design and methods*, 5th ed. Newbury Park: Sage Publications, 2009.