A Reference Architecture for Big Data Systems in the National Security Domain

John Klein
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA, USA
jklein@sei.cmu.edu

Ross Buglak, David Blockow, Troy Wuttke, Brenton Cooper
Data to Decisions Cooperative Research Centre
Kent Town, SA, Australia
{ross.buglak, david.blockow, troy.wuttke, brenton.cooper}@d2dcrc.com.au

ABSTRACT
Acquirers, system builders, and other stakeholders of big data systems need to define requirements, develop and evaluate solutions, and integrate systems together. A reference architecture enables these software engineering activities by standardizing nomenclature, defining key solution elements and their relationships, collecting relevant solution patterns, and classifying existing technologies. Within the national security domain, existing reference architectures for big data systems have not been useful because they are too general or are not vendor-neutral. We present a reference architecture for big data systems that is focused on addressing typical national defence requirements and that is vendor-neutral, and we demonstrate how to use this reference architecture to define solutions in one mission area.

CCS Concepts
• Information systems~Data analytics • Information systems~Online analytical processing • Information systems~Information retrieval • Information systems~Data management systems • Information systems~Spatial-temporal systems • Software and its engineering~Software infrastructure • Software and its engineering~Distributed systems organizing principles

Keywords
Reference architecture; big data

1. INTRODUCTION
The national security application domain includes software systems used by government organisations such as police at the local, state, and federal level; military; and intelligence. Big data systems are pervasive in this domain, with applications ranging from:

- Predictive maintenance of aircraft, ships, and vehicles, combining measured data collected on the platform with meteorological data, equipment supplier data, and other sources to optimise maintenance schedules (e.g., [1]).
- Geospatial analytics that identify movement and changes of features on the ground, to support tactical, operational and strategic intelligence analysis and planning.
- Network graph analysis to help police identify associates and organisational affiliations.

Stakeholders who specify, evaluate, and acquire these big data systems often lack software engineering technical expertise in this emerging and dynamic technology space [2]. While these stakeholders may have competence in other types of software systems, the principles and practices for big data systems are different, and general software knowledge may not be sufficient to ensure success [3].

A reference architecture (RA) serves as a mechanism to represent and transfer software engineering knowledge that bridges from the problem domain to a family of solutions. A RA defines domain concepts and relevant qualities, decomposes the solution and creates a lexicon to enable efficient communication, and provides guidance and principles for system stakeholders [4].

There are a number of published RAs for big data systems. However, these were not useful for our clients in the national security domain, because they were too general (e.g., [5], [6], or [7]) or because the solutions were specific to a particular vendor's technology (e.g., [8]). We discuss these in more detail in the Related Work section below.

The contribution of this paper is a big data RA for applications in the national security domain, which includes:

- Motivating use cases;
- Architecture decomposition based on grouping of related concerns into architectural modules;
- Mapping of current technologies onto the concerns;
- Demonstration of how to use the RA to create big data system architectures.

Each of these topics is discussed in a subsequent section of the paper.

2. RELATED WORK
RAs are a powerful software engineering knowledge transition tool, capturing both domain and solution knowledge for a portfolio of related systems [9]. In their survey of the state of the practice, Cloutier, et al. note that RAs facilitate multi-site, multi-organisation, and multi-vendor systems, which are all primary considerations in our application domain.

There are a number of existing big data RAs. The US National Institute of Standards and Technology (NIST) shepherded a community of researchers and practitioners to create a 7-volume Big Data Interoperability Framework, which includes a Reference

BIGDSE'16, May 16 2016, Austin, TX, USA
 ISBN 978-1-4503-4152-3/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2896825.2896834

Architecture volume [10]. The NIST framework reflects the contributions of more than 35 authors, and included public review and comments, producing an architecture that is broad in coverage and applicability, but uneven in depth and detail. The RA presented below bears some superficial similarities to the NIST framework, but is distinguished by several key differences:

- While both architectures are decomposed into elements with similar names, the decomposition rationale and principles are never articulated in the NIST architecture and so an architect cannot easily allocate functions and qualities to elements of that architecture. Our architecture is organised using explicit principles, discussed below, allowing architects to easily allocate new functions and qualities.
- Our RA draws a clear system boundary, with data producers and consumers outside of the scope of the system. The NIST architecture includes data producers and consumers inside the RA, which leaves the scope effectively unbounded.
- The NIST RA is domain-agnostic. As such, it does not satisfy the first key principle of that Cloutier, et al. identify for a reference, architecture, namely that of elaborating mission, vision, and strategy [9]. The NIST RA represents a "Meta RA", which must be further refined for an application domain. Our RA could be viewed as one such refinement.

The strengths of the NIST RA include strict vendor neutrality, a stand-alone volume providing clear definitions of big data terminology, and a comprehensive inventory of use cases across many domains (although the relationship of the RA to those use cases is not part of the baseline release).

Technology vendors such as IBM [6], Oracle [7], and Microsoft [8] have produced big data RAs. Like the NIST RA, these RAs are not domain-specific, and while there are some domain-specific refinements presented, none of those refinements reflects the national security application domain.

By structuring the problem and solution domains, RAs complement other architecture knowledge sharing approaches. For example, knowledge bases can provide more detailed guidance for architects in specific areas of a RA, such as QuABaseBD that focuses on the Storage module concerns and technologies in this RA [9].

3. DOMAIN REQUIREMENTS AND USE CASES

The domain-specific requirements for this RA were discovered by analysing use cases in four mission capability areas. The mission capability areas were selected to cover a broad set of functions, deployment topologies, and data processing capabilities.

The mission segments and uses cases analysed were:

1. Strategic Geospatial Analysis and Visualisation – here we assessed map production from satellite imagery, which includes displaying the image with overlays showing known features, such as roads and buildings, and identifying and annotating new and changed features (i.e. adding metadata). This mission segment also included a use case that searched map data/metadata and rendered the results.
2. Full-motion video analysis – this capability is used in missions ranging from search-and-rescue to surveillance from fixed or mobile cameras. The use cases here were to acquire, render, and store a digital video stream, and to detect and track objects of interest.
3. Open Source Intelligence – This mission capability is used for decision support. Use cases include collecting and storing open source data, such as web sites, social media (including text, audio, and video), identifying entities (people, organisations) and relationships to populate a knowledge graph, querying the knowledge graph, and using the knowledge graph to summarise information about entities.
4. Signals Intelligence Analysis – use cases in this mission area were to capture and store electronic transmissions, and to execute analytics to match new captures to archived transmissions.

These use cases were analysed to identify requirements categories and general requirements relevant to big data, in areas such as data types (e.g., unstructured text, geospatial, audio), data transformations (e.g., clustering, correlation), queries (e.g., graph traversal, geospatial), visualisations (e.g., image and overlay, network), and deployment topologies (e.g., sensor-local processing, private cloud, and mobile clients).

4. REFERENCE ARCHITECTURE

4.1 Organisation of the Reference Architecture

The RA metamodel is shown in Figure 1. The architecture is a collection of *modules*, which decompose the solution into elements that realise functions or capabilities, and that address a cohesive set of *concerns*. Concerns are addressed by *solution patterns,* or by *strategies*, which are design approaches that are less prescriptive than solution patterns. Together, modules and concerns define a solution domain lexicon, and the discussion of each concern relates problem space terminology (origin of the concern) to the solution terminology (patterns and strategies).

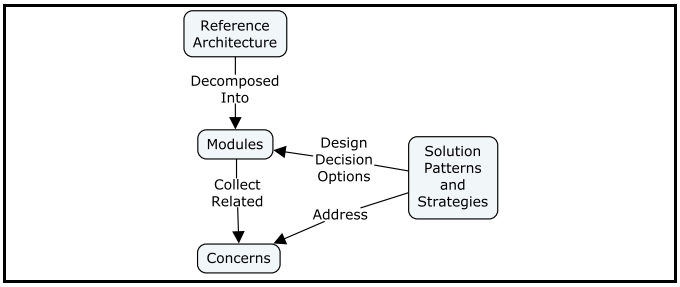

Figure 1 – Reference Architecture Concepts

The concerns are multi-faceted. Some concerns capture external constraints on the system (e.g., type of workload), design decisions (e.g., optimisations), or system quality attributes (e.g., latency and ease of programming). This type of concern has a significant impact on the design, analysis, or evaluation of a module.

A second type of concern was related to reuse or sharing of modules. These concerns included differences in execution triggers/rates (e.g., driven by input data streams, user requests, or fixed period) and whether the functions or capabilities provided by a module were typically shared within or external to the big data system.

Other concerns align with stakeholder communities of interest or stakeholders roles, such as processing algorithms or system management. This type of concern helps stakeholders orient their perspective on the RA by identifying the modules each stakeholder needs to focus on.

The last type of concern represents *de facto* partitioning of the commercial and open source packages and frameworks used to realise big data solutions. Reflecting this partitioning in the module decomposition simplifies the mapping between off-the-shelf technology and the RA and helps stakeholders position vendors and products within the RA.

This RA is intended to supplement other sources of general architecture knowledge: The scope of the concerns identified in the RA was limited to issues that arise from the volume, variety, and velocity of data in big data systems, and the solution patterns and strategies are focused on addressing these concerns in the big data context of high scale and heterogeneity. For example, usability is obviously a concern in any human-computer interface, and so this was not specifically identified as a concern in the RA. However, in a big data system, providing an indication of data confidence (e.g., from a statistical estimate, provenance metadata, or heuristics) in the user interface impacts usability, and this was identified as a concern for the Visualisation module. The concern-driven decomposition discussed below reflects this scoping.

4.2 Module Decomposition

Figure 2 shows the system boundary and module decomposition of the RA. The RA assumes a system of systems context [10], where Data Providers and Data Consumers are external systems that are not under the same design or operational authority as the big data system. These systems may be instances of big data systems developed using this RA (or another architecture).

The 13 modules are grouped into three categories: The Big Data Application Provider includes application-level business logic, data transformations and analysis, and functionality to be executed by the system. The Big Data Framework Provider includes the software middleware, storage, and computing platforms and networks used by the Big Data Application Provider. As shown in the figure, the system may include multiple instances of the Big Data Application Provider, all sharing the same instance of the Big Data Framework Provider.

The third module category is Cross-Cutting Modules. Each of the three Cross-Cutting modules addresses a set of concerns that impact nearly every module in the other two categories.

The following subsections discuss the modules in each of the three categories.

4.2.1 Big Data Application Provider Modules

4.2.1.1 Application Orchestration Module

Application Orchestration configures and combines other modules of the big data Application Provider, integrating activities into a cohesive application. An application is the end-to-end data processing through the system to satisfy one or more use cases.

Orchestration may be performed by humans, software, or some combination of the two, and may be fixed at system design time or configurable via a Graphical User Interface (GUI) or Domain Specific Language (DSL).

4.2.1.2 Collection Module

The Collection module is primarily concerned with the interface to external Data Providers. The Collection module is concerned with matching the characteristics and constraints of the providers and avoiding data loss.

4.2.1.3 Preparation Module

The main concern of the Preparation module is transforming data to make it useful for the other downstream modules, in particular Analytics. Preparation performs the transformation portion of the traditional Extract, Transform Load (ETL) cycle, including tasks such as:

- Data validation (e.g. checksum validation);
- Cleansing (e.g. removing or correcting bad records);
- Optimisation (e.g. de-duplication);
- Schema transformation and standardization;
- Indexing to support fast lookup.

The Preparation module may perform basic *enrichment*, which adds information from other sources to a data record. The enrichment process begins in Preparation and continues in Analytics. The enrichment preformed in Preparation is usually very simple processing, such as creating record counts for particular types or categories, or performing a lookup to add location name based on latitude and longitude values. Later, the Analytics module may perform more sophisticated enrichment, for example, using a recommendation engine to create new associations to other records.

4.2.1.4 Analytics Module

The Analytics module is concerned with efficiently extracting knowledge from the data, typically often working with multiple data sets with different data characteristics. Analytics can contribute further to the transform stage of the ETL cycle by performing more advanced transformations and enrichments to support knowledge extraction.

4.2.1.5 Visualisation Module

The Visualisation module is concerned with presenting processed data and the outputs of analytics to a human Data Consumer, in a format that communicates meaning and knowledge. It provides a "human interface" to the big data. Data Consumers are external to the big data system.

Some visualisation techniques may involve producing a static document, cached for later access (e.g. a text report or graphic), however other techniques often include on-demand generation of an interactive interface (e.g. navigating and filtering search results, or traversing a social graph). Display of data confidence and/or data provenance information is common for machine-generated data, and the interactive visualisations may include the ability to create, confirm, or correct (i.e. update) data.

4.2.1.6 Access Module

The Access module is concerned with the interactions with external actors, such as the Data Consumer, or with human users, via Visualisation. Unlike Visualisation, which addresses "human interfaces", the Access module is concerned with "machine interfaces" (e.g. APIs or web services). The Access module is the intermediary between the external world and the big data system to enforce security or provide load balancing capability.

Similar to the Collection module, the primary concern of the Access module is matching the characteristics of the external systems. The format and style of the interfaces to systems will vary, and data may be pulled or pushed by the Access module.

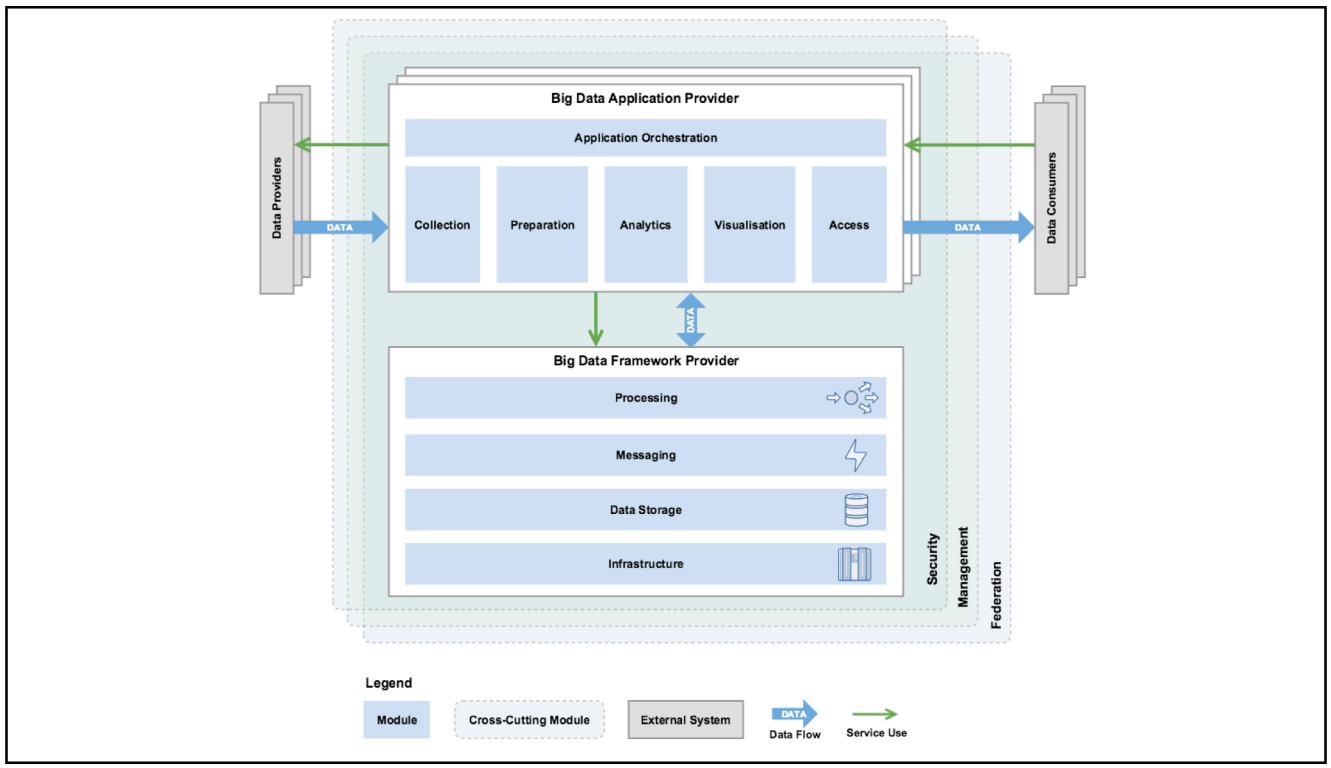

Figure 2 – Module Decomposition of the Reference Architecture

4.2.2 Big Data Framework Provider Modules

4.2.2.1 Processing Module

The Processing module is concerned with efficient, scalable, and reliable execution of analytics. It provides the necessary infrastructure to support execution distributed across 10s to 1000s of nodes, defining how the computation and processing is performed.

A common solution pattern to achieve scalability and efficiency is to distribute the processing logic and execute it locally on the same nodes where data is stored, transferring only the results of processing over the network. The large number of processing nodes and the long execution duration of some analytic processes lead to concerns about process or node failure during execution. Another critical concern of the Processing module is the ability to recover and not lose data in the event of a process or node failure within the framework.

4.2.2.2 Messaging Module

The Messaging module is concerned with reliable queuing, transmission, and delivery of data and control functions between components. While messaging is common in traditional IT systems, its use in big data systems creates additional challenges.

Big data solutions are often comprised of many different products and frameworks, making integration a primary concern. The Messaging module must support a variety of clients, programming languages, and enterprise integration patterns.

The volume and throughput of messages in big data solutions is a particular concern, and can necessitate distributed messaging frameworks. Volume also leads to concerns if durability (i.e. permanently storing all transferred messages) is needed.

4.2.2.3 Data Storage Module

The primary concerns of the Data Storage module are providing reliable and efficient access to the persistent data. This includes the logical data organisation, data distribution and access methods, and data discovery (using e.g. metadata services, registries and indexes).

The data organisation and access methods are concerned with the data storage format (e.g. flat files, relational data, structured/unstructured data) and the type of access required by the big data Application Provider (e.g. file-type API, SQL, graph query). It is common for the Data Storage module to provide more than one representation of a single data record (a type of de-normalisation) to support efficient analytic execution for different use cases.

When data is distributed across a cluster, the Data Storage module will be concerned with the availability and consistency of the data, and the tolerance of partitions (network or node faults) within the cluster.

4.2.2.4 Infrastructure Module

The Infrastructure module provides the infrastructure resources necessary to host and execute the activities of the other BDRA modules. This includes:

- Networking: resources that transfer data from one infrastructure framework component to another;
- Computing: physical processors and memory that execute software;
- Storage: resources which provide persistence of the data;
- Environmental: physical resources (e.g. power, cooling) that must be accounted for when establishing an instance of a big data system.

Infrastructure and data centre design are concerns when architecting a big data solution, and can be an important factor in achieving desired performance. Big data infrastructure needs to be scalable, reliable and support target workloads.

4.2.3 Cross-Cutting Modules

4.2.3.1 Security Module

Security concerns affect all modules of the RA. The Security module is concerned with controlling access to data and applications, including enforcement of access rules and restricting access based on classification or need-to-know.

Security is also concerned with intrusion detection and prevention. Big data systems can include introspective analytics that look at internal data and access patterns to perform intrusion detection.

Big data can present an attractive target to attackers, and in general, security and privacy have not been primary concerns in the development of many big data technologies (e.g., see the security survey in http://www.quabase.sei.cmu.edu). Consistent application of controls requires a holistic approach to security and privacy as data traverses multiple components of the architecture.

4.2.3.2 Management Module

Concerns for the cross-cutting Management module are grouped into two broad categories:

- System Management, including activities such as monitoring, configuration, provisioning and control of infrastructure and applications;
- Data Management, involving activities surrounding the data lifecycle of collection, preparation, analytics, visualisation and access.

4.2.3.3 Federation Module

The Federation module is concerned with interoperation between federated instances of the RA. These concerns are similar to typical system of systems (SoS) federation concerns [10], however existing SoS federation strategies may not support the scale of big data systems.

4.2.3.4 Common Concerns

This "module" collects a set of concerns that did not map cleanly into any of the other modules, and which are not related to each other in any meaningful way, but should be considered in the architecture of a big data system. These other concerns are:

- Scalability - the ability to increase or decrease the processing and storage provided, in response to changes in demand. In a perfectly scalable system, the cost of the provided resources is linearly related to the demand (usually up to some resource limit). In systems that are less scalable, the cost of the provided resources increases faster than the demand, or the resource limit may be unacceptably low. In big data systems, scalability is often dynamic or "elastic", and the architecture enables the system to adjust at runtime to changes in workload. Scalability needs to be considered at all layers of a big data architecture, from the data centre infrastructure through to the application layer.
- Availability - the ability for a system to remain operational during fault conditions such as network outages or hardware failures. Similar to scalability, a holistic approach needs to be take to designing for availability, as a single component can prevent a system from providing the required level of availability.
- Data organisation is a common concern, particularly for high data volume use cases, as the way that data is stored can significantly impact performance downstream in the processing pipeline. Data organisation design decisions can't be deferred, but must be made early, so that each stage in the processing pipeline stores data so the next stage can access it efficiently.
- Technology stack decisions, both hardware and software, are driven by several interwoven considerations. In addition to features, concerns include standardization within the system, maturity, ease of operation, vendor support, cost, and staff skills.
- Accreditation, which is a domain-specific concern and involves assessing the cybersecurity qualities of the system. Software accreditation can be challenging for big data solutions due to the prevalence of open source products in solution architectures.

5. MAPPING CURRENT TECHNOLOGY

Big data system architectures and implementations rely on composition of existing open source and commercial software technologies [3]. In the national security application domain, in particular, acquirers evaluating and analysing proposed solutions need an understanding of which off-the-shelf technologies are appropriate (or not appropriate) to satisfy a function or quality within the RA. Furthermore, most users of this RA already have big data systems within their enterprise, and a technology mapping provides an easy first step to view those systems through the RA lens.

To satisfy these stakeholder needs, our RA provides a mapping of commercial and open source products to modules. This is a simple tabular mapping, where rows are products and columns are modules in the RA. An "x" at an intersection indicates that the particular product is an appropriate technology to use in the module. Products were identified based on stakeholder's current big data systems, and from proposals for new systems. There were 35 products mapped to Big Data Application Provider modules, and 64 products mapped to Big Data Framework Provider modules.

This mapping is maintained in a separate volume of the RA, as it is the most dynamic content, and is the least normative and prescriptive content.

6. USING THE RA TO DEFINE SYSTEM ARCHITECTURES

Our RA concludes with a volume that contains tutorial information showing stakeholders how to use the architecture to create concrete solution designs, including examples of identifying relevant concerns, making design decisions, and selecting appropriate strategies and design patterns.

Although the RA modules are presented "input to output" and "top to bottom", as described above in Section 4, the tutorial recommends that stakeholders consider the modules in a different order at design time, which reflects a user-centric requirements perspective and also reflects the main design decision dependencies among the modules. The recommended design time order is shown in Figure 3. The recommended design process produces an initial system architecture, which would typically be refined as the system is prototyped and developed.

1. **Visualisation** - What information do the users need?
2. **Collection** – What are the data sources and how to collect the data?
3. **Analytics** - What information needs extracting from the data?
4. **Preparation** - What transformations are needed prior to the analytics?
5. **Data Storage** - How will the data be stored to support analytics, visualisations and access?
6. **Processing** - How will analytics execute?
7. **Application Orchestration** - Does the processing pipeline need orchestration?
8. **Access** - What API access is required? How will the data be retrieved?
9. **Messaging** - Is supporting messaging infrastructure required?
10. **Management** - How will the application and infrastructure be managed?
11. **Security** - What security controls are required?
12. **Federation** - Does the solution need to operate within a federation?
13. **Infrastructure** - What infrastructure is needed?

Figure 3 – Design time ordering of RA modules

The rest of this section describes how the RA is used to design a simple open-source intelligence (OSINT) system, that takes data from Twitter feeds (Tweets) and detects events (e.g., protests, riots, etc.), and then correlates detected events with data from news media websites. This description is highly abbreviated, touching on some of the important module refinements and skipping over many of the less interesting concerns.

We begin with Visualisation concerns, and decide that the primary visualisation will display detected events on a map display, and allow filtering to a specified time range. This need for geospatial display and processing leads to an initial architecture iteration shown in Figure 4.

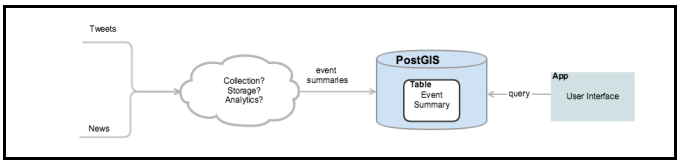

Figure 4 - OSINT Architecture – Step 1

Next, we consider the Collection concerns. The Twitter data and web page data is semi-structured, and consists of text and images. Based on the Twitter API limits, and a need to retain 3 years of live data, we decide that HBase is a good candidate for storing collected data. The result of these decisions is the second step of the architecture, shown in Figure 5.

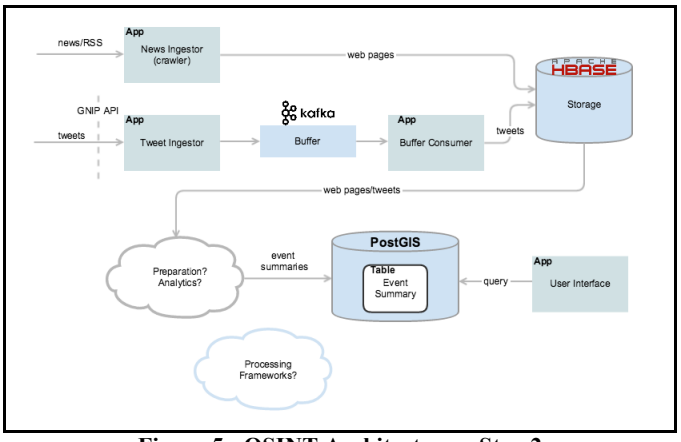

Figure 5 - OSINT Architecture – Step 2

We next consider Analytics concerns. The approach to detecting events from Twitter feeds is to detect anomalies in vocabulary (new words, phrases, tags, etc.), cluster those anomalies, and finally categorise the target events and discard non-interesting events (such as related to a major weather event or celebrity appearance). This machine learning-based pipeline is instantiated as Step 3 of the architecture, shown in Figure 6.

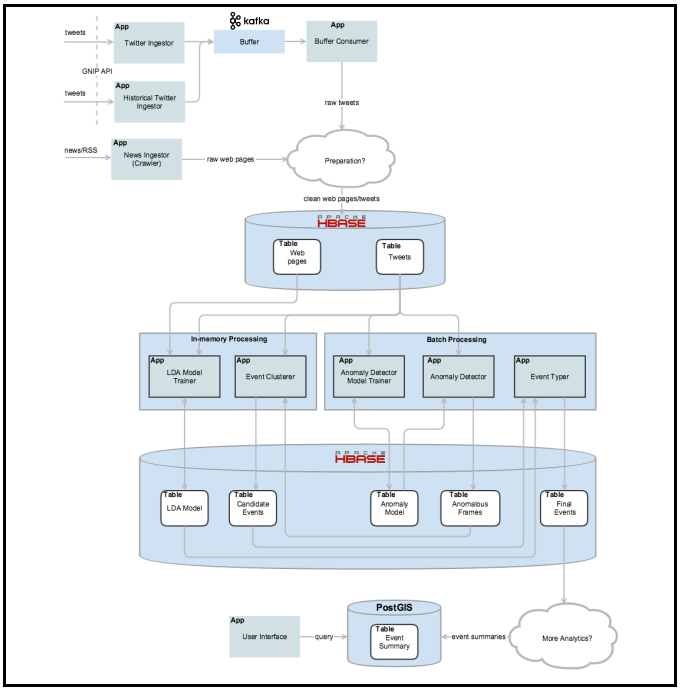

Figure 6 - OSINT Architecture – Twitter Event Detection

Concerns related to Preparation are primarily data cleansing and normalisation, e.g., Tweet geo-tags are converted to Region IDs, emoticons are converted to text, and duplicate news pages will be removed. Tweets will be processed through a stream pipeline, and new pages through a batch pipeline.

We next turn to Data Storage concerns. The primary Tweet access by analytics is by region ID and time range, so we decide to shard by region ID. We see that we can simplify some of the analytic processing and training by denormalising our data, creating an hourly summary for each region that includes counts of Tweets, unique users, new users, flags for the intervals that contain anomalies, and labelling data for training the anomaly detector. Similar reasoning is applied to define the storage structures for the news page data.

Processing concerns depend on the Analytics to be performed. In this case, we choose Hadoop for batch processing, and Spark Streaming for stream processing for both Preparation and Analytics.

Our Application Orchestration identifies continuously, hourly, daily, monthly, as shown in Figure 7. We decide to use Apache Oozie as the workflow manager, because it fits well with the processing ecosystem that we have chosen earlier.

Access concerns show that a thin browser client is appropriate to overlay events onto maps. A separate map server and event server are instantiated, with the event server hiding the complexity of the SQL queries to the PostGIS database.

Management concerns included logging, metrics collection, and health monitoring. Also, a data backup and retention strategy was chosen, and a system provisioning framework was chosen. These are shown in Figure 8.

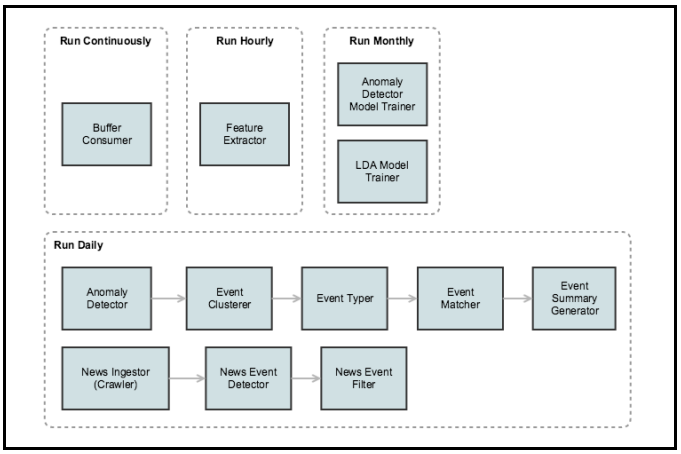

Figure 7 - OSINT Architecture - Workflows

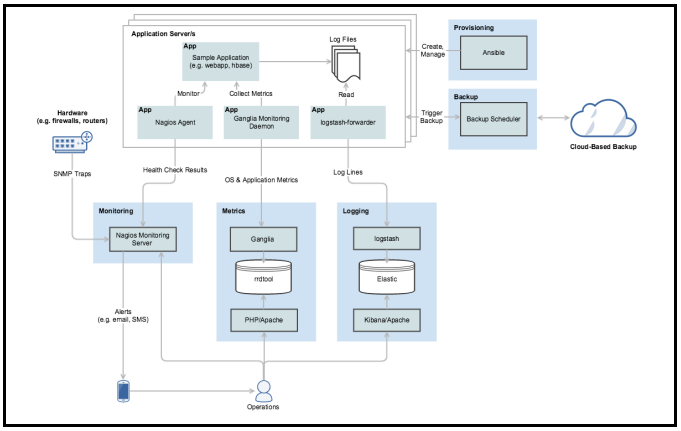

Figure 8 - OSINT Architecture - Management

Infrastructure concerns led to a combination of scale-up and scale-out approaches. A significant concern was that the volume of the data sets drives cost to the point where we must share a single deployed system between the test and production teams. This is accomplished using a multi-tenant strategy, with separate logical namespaces on the Hadoop cluster and separate tables in the PostGIS database server.

7. CONCLUSIONS AND FUTURE WORK

We have described a reference architecture for big data systems in the national security application domain, including the principles used to organise the architecture decomposition. This RA serves as a knowledge capture and transfer mechanism, containing both domain knowledge (such as use cases) and solution knowledge (such as mapping to concrete technologies). We have also shown how the RA can be used to define architectures for big data systems in our domain.

Future work includes:

- Using the module decomposition in the RA to make decisions on where to standardize interfaces and implementations within a particular enterprise;
- Creating new narrow and deep knowledge bases, similar to QuABaseBD (www.quabase.sei.cmu.edu) for other modules within the RA;
- Evaluating the utility of the RA to define software product lines for sub-domains within the scope of the RA;
- Creating instantiations of the RA for specific use cases within the intelligence domain.

8. ACKNOWLEDGMENTS

Copyright 2016 ACM. This material is based upon work funded and supported by the Data to Decisions CRC (D2D CRC), the Australian government's Cooperative Research Centres Programme, and by the United States Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute. [Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. DM-0003215.

9. REFERENCES

[1] Lockheed Martin, "Autonomic Logistics Information System (ALIS)." Lockheed Martin, Inc., Brochure, CS00086-55, 2009.

[2] Infochimps, "CIOs and big data: What your team wants you to know.", http://www.infochimps.com/resources/report-cios-big-data-what-your-it-team-wants-you-to-know-6/ (Accessed 18 Jan 2016).

[3] I. Gorton and J. Klein, "Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems," *IEEE Software,* vol. 32, no. 3, pp. 78-85, May/June 2015. doi: 10.1109/MS.2014.51.

[4] R. Cloutier, G. Muller, D. Verma, et al., "The Concept of Reference Architectures," *Systems Engineering,* vol. 13, no. 1, pp. 14-27, 2010, doi: 10.1002/sys.20129.

[5] NIST Big Data Public Working Group Reference Architecture Subgroup, "NIST Big Data Interoperability Framework: Volume 6: Reference Architecture." National Institute of Standards and Technology, Special Publication, 1500-6, 2015.

[6] D. Mysore, S. Khupat, and S. Jain, "Big data architecture and patterns." IBM, White Paper, 2013, http://www.ibm.com/developerworks/library/bd-archpatterns1/ (Accessed 1 Jan 2016).

[7] Oracle, "Information Management and Big Data." White Paper, 2014, http://www.oracle.com/technetwork/database/bigdata-appliance/overview/bigdatarefarchitecture-2297765.pdf

[8] Microsoft, "Microsoft Big Data Solution Brief.", http://download.microsoft.com/download/F/A/1/FA126D6D-841B-4565-BB26-D2ADD4A28F24/Microsoft_Big_Data_Solution_Brief.pdf

[9] J. Klein and I. Gorton, "Design Assistant for NoSQL Technology Selection," in *Proc. 1st Int. Workshop on the Future of Software Architecture Design Assistants (FoSADA'15)*, Montreal, Canada, 2015. doi: 10.1145/2751491.2751494.

[10] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering,* vol. 1, no. 4, pp. 267-284, 1998, doi: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.