1 Executive Summary

The Reality Warp Software (RWS) team brings personal experience and lessons-learned from architecting and developing the DCGS-Army's current fusion framework, correlation engine and the Army's first cloud-enabled, mobile, biometric platform. This personal experience gained by Reality Warp Software's founder and President as the Chief Architect for these prior solutions provides a unique perspective and a promising initial direction to Reality Warp Software in providing distributed, scalable frameworks for Big Data analytics. Our architect for Entity Management encompasses the capabilities sought under this RFI. We are currently developing a similar solution for Entity Management around personal identification, biometrics, and cross-referenced data sources. The RWS team learned Entity Management through personal experience in architecting and developing solution for correlation and de-duplication engine for the DCGS-A Tactical Entity Database built on the DCGS-A fusion framework. From this experience, RWS has developed an early capability around biometric Identity to demonstrate Entity Management.

Reality Warp Software has developed several tools that provide us the capability to rapidly developing analytics. We have built the Pirkolator, which provides a distributed, scalable and extensible communication infrastructure that is the foundation for analytics. We have built in support for data ingest that supports many formats, such as JSON, compression techniques, and security protocols, such as PGP. We provide built-in support to ingest from a variety of different data source types from message queues using AMQP to cloud sources like Amazon's Web Services (AWS) to persistent stores like relational databases and NoSQL stores. The infrastructure provides a transformation library that can automatically transform ingested data into other forms. We provide an abstracted data layer component that is used to provide access, search, query, and changes to specific data source implementations. Using this infrastructure, we built the Determinator to provide a computation engine for parallel processing targeted for use with association, correlation, and aggregation analytics. We built the Trendinator to provide an engine and tools for managing analytics work with data in series, such as over time. We have built a distributed management capability to load target data into these engines to prevent duplicate computations and enforce concurrent access when analytics work with dynamic, changing data. We use these tools to explore new ways to think about data.

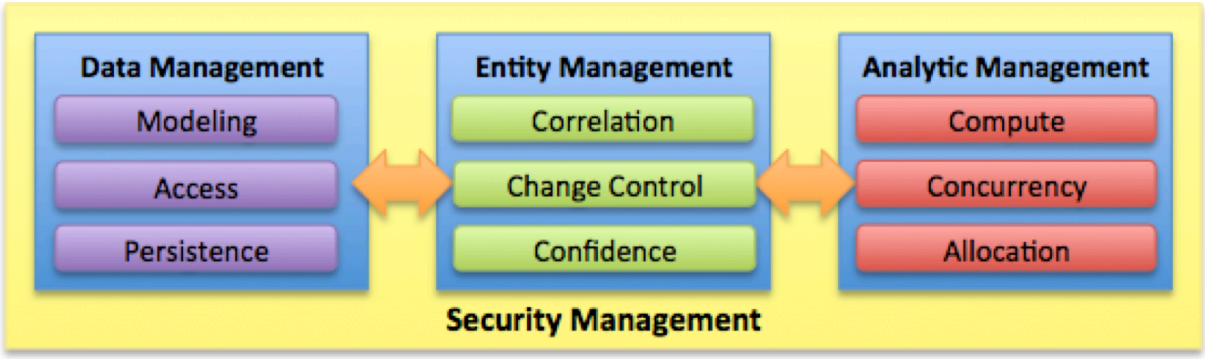

Figure 1: Management Tiers

2 Technical Information

2.1 Architecture

Our past experience building similar capabilities for entity correlation has lead us to a standard architecture that we find useful in providing solutions for Entity Management. This architecture encapsulates the capabilities sought under this RFI. Our architecture addresses the infrastructure and supporting tools required for managing large sets of Entities. Additionally, this architecture provides the analytics and computation platform needed for change management, pedigree, confidence, and correlation needed. Figure 2 illustrates our architecture, which outlines three modular, management tiers with an encompassing security management tier that supports the other three tiers.

Data Management provides access and control to disparate data stores. This allows access to data in a single, unified manner for services using the data. The Data Management Layer provides access to existing data sources and a correlated data source. The correlated data source is needed to manage correlation between entities from different schemas. To manage different schemas we use transformation services to position the data as needed for how the data is being used. Transformation services use custom mappings to define how data is transitioned between the different schemas. Additionally, transformation services are used to position data into shareable schemas that can reside along side the existing data sources. Data change is a difficult, but important process. As data changes, you must be able to track the pedigree of changes. Not all changes are linear either, so data change often starts to resemble a tree structure with various branches and leaves defining different paths that data change takes, some ending with higher confidence and some containing more complete information sets.

Entity Management provides access and control to entities such as Identities and Events. We propose a simple Entity, Attribute, and Relationship (EAR) model to capture correlated data from cross-sources. An Entity describes a top-level record describe an Identity, Event, or other well-known construct. Attributes are used to describe the dynamic properties of each Entity.

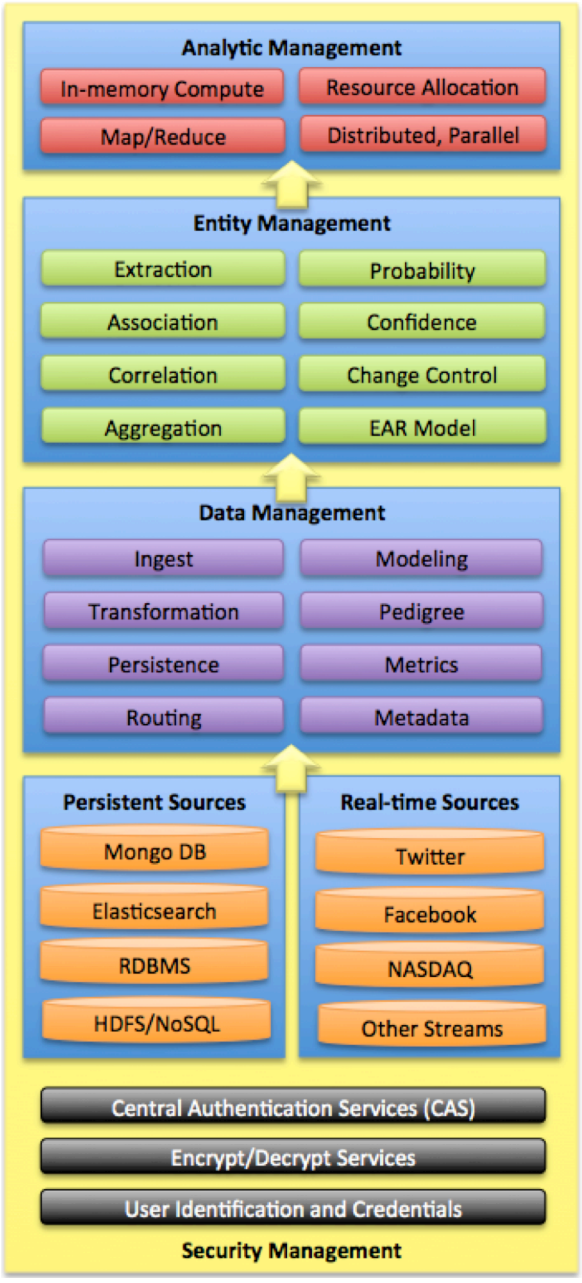

Figure 2: Suggested Architecture

Relationships are used to describe the properties and behaviors that exist between Entities. The EAR model provides us an object model that is common across analytics. This allows us to write an analytic once and have it effective against multiple data schemas. We further use transformation services to map and convert data from our EAR model into formats that are optimized for each analytic. Meta-data is an important concept in our EAR model. We use meta-data attached to Entities, Attributes, and Relationships to include data about each construct. Important information such as source of origin, last update or version, credentials, and pedigree are examples of meta-data that we attached to each object.

Analytic Management provides the access, control, and execution to analytics that contribute to Entity Management. We recommend utilizing an Analytics Management platform that leverages stand-alone, in parallel, and distributed techniques to execute analytics. Our approach to Analytic Management utilizes both file system stores, such as HDFS/Hadoop, and in-memory data grids, such as Hazelcast. Correlation, aggregation, probability, and pattern recognition are four analytic techniques that we use for discovery. Correlation is the technique of comparing two or more entities. Aggregation is the technique used to group entities. Probability is the technique to promote statistical discovery. Lastly, pattern recognition is the technique used to apply patterns across data sets for discovery. These four techniques each require different processing architectures, but the same technique can be used for different discoveries. Assessment, behavior, and similarity are three types of discovery that can be applied using the four techniques. Assessment looks to describe entities. Assessment covers generating confidence and accuracy of data. Behavior looks to provide how things interact or change in predictable ways. Similarity looks to ascribe the likeness or difference between entities. Lastly Representation is the form that discovery results take, such as an alarm or alert, a report, or visualization.

Security Management encapsulates the three management tiers described above to promote security across the entire application. This tier manages system, analytic, and data access using user credentials. PKI is used for user authentication. We utilize a data assembly process, much like the transformation services, to promote security levels to requested data down to the attribute level. User credentials are used to build the data based on security levels with a validation engine to provide assurance that the assemble data meets the appropriate credentials of the user, analytic, and/or system generating the response.

2.2 Insights of Entity Modeling

Our solution to Entity Management starts with a very simple, yet extensible, data model with three main constructs – Entities, Attributes, and Relationships. An Entity is simply a type-based element, such as a Person or Event, with attached Attributes. Attributes provide the properties that describe the characteristics of an Entity, such as a Person's name or an Event's location. A Relationship is a special type of Attribute that describes the Associations between an Entity and other Entities, such as group membership or familial relationships. Each Entity, Attribute, and Relationship construct contains a map of metadata that is used to describe structural and descriptive elements. To manage cross-data-store elements, we use a simple Reference construct that contains link information to identify the source of each data element, how to map the elements between the source schema and our EAR model. We think of a Reference very much like an Index, but with more information to help reconstruct the Entity. Additionally, the Reference also contains information about security elements and pedigree, confidence details.

We use this model combined with a Correlation Entity Manager (CEM) to provide for organizing data across data stores. An Entity has a unique identifier that is established at the first registration of a type of entity with a Correlation Entity Manager. The Entity has References that point to data sources that contribute data to that Entity. Any attached Attributes on the Entity also have References to sources that contribute data. This information is store in a central, Correlated Entity data store. We have used Mongo DB with great success, but any database can be used as the central, Correlated Entity data store. We use a Resolver to extract and translate the values from the source system. For data stores that are not under control by our infrastructure, such as read-only data store, we use meta-data with source information to promote access to the most current data. The Resolver uses this meta-data to look-up each element on-demand to ensure the most current, timely value is used. For sources where we control the infrastructure, we can promote data to the Correlated Entity data store for faster look-up.

We want to be able to consolidate as much of the important information from the source data stores in the Correlated Entity data store. The problem with this is that data changes and we have limited access to some data stores. Additionally, data formats across each source are different. To solve the data change problem we use the Reference construct to describe the basic correlation elements that are not impacted by change or access, such as the Correlated Entity Unique ID (CEID), Source Identifier, and Source Entity Identifier. The CEID is the global identifier for each unique Entity and is assigned by the CEM when the Entity first comes under control. The Source Identifier is used during assembly of data and when establishing data consolidation into the CE data store. The Source Entity Identifier tells us how to access the Entity data on the source system that contains the original data or data elements.

For disparate data formats we recommend using Transformation Services to translate data between the models. By centralizing Entity transformations, we can automatically map data between disparate schemas. Additionally, we can more easily adapt to schema changes using Entity metadata and versioning. We combine the use of custom developed transformations and open source mapping applications, such as Dozer, to provide configurable elements to transform. We feel that a centralized schema, such as the EAR model, is useful in providing Entities that are shared across platforms, but we also recognize that an end-all-be-all schema is very difficult and not necessarily the best form for all analytics. For this reason, we believe that transforming Entities into the best, optimized format for use is the optimal solution. Transformation Services provide the means necessary to support today's disparate data schemas while also providing future evolution to more established, centralized schemas and standards.

2.3 Entity Change Management

As Entities change within systems, it is important to be able to identify all the actors making changes. The first step in this process is to define the type of information to be tracked within the system. We refer to this as the Pedigree of an object or piece of data. Pedigree defines the "who, what, and whens" of Entity change. This construct also provides the means for assembling an Entity based on these elements. For instance, if you want to view an Entity as it has been modified by a specific source, this mechanism allows us to do that.

We use a GIT-like architecture for change management. This allows us to provide branching and merging of Entities around each element of Pedigree. The Correlation Entity Manager uses the Entity Change Management process to provide a Correlated Change Management System (CCMS). When an Entity comes under management, it is "checked into" or committed to the change architecture. This establishes the Entity with the change management system and allows further changes to be versioned, branched, and merged as the

Entity evolves. Changes are propagated through three-tiers. The first tier is the Working tier. When an analytic is working with an Entity, changes are only available to that analytic and are not propagated outside of that realm. When an analytic is ready to promote changes to the next tier, a "commit" of the Entity is issued to the CCMS. This commit makes the changes available to the Local tier. The Local tier provides Entity change across a system instance of the CCMS. This is generally a single Java Virtual Machine (JVM) instance running the Correlation system. Analytics within a single JVM instance can share changes through the Local tier. To propagate changes across all instances, a "push" command with the Entity is issued to the CCMS. This propagates, or pushes, the changes to all the other instances and makes the change globally available.

2.4 Entity Assembly

With an Entity under Change management, it is necessary to provide tools for building an Entity from configurable parameters. We use an Entity Assembly process to build Entities from a given configuration. The Entity Assembly process traverses the Entity Change trees with assigned Pedigree to build an Entity under various configurations. Configurations for assembly are things like by version, source, confidence, or time. This process allows an analytic develop to build an Entity from a specific version, source, or other Pedigree elements.

It is also necessary to perform validation during the assembly process to enforce access control. We use the concept of a Validator to perform thus function. During the assembly process, Validators perform the checks to ensure that the appropriate Entity elements, Attributes, and Relationships are assembly for that Entity based on user credentials. This allows us to provide Attribute leave authorization of data elements.

2.5 Correlation and Probability Analytics

We build Entity comparison analytics to run in our distributed, parallel computation platform called the Determinator. Using this engine we can apply any number of analytics to compute information about two entities. We recommend using in-memory techniques to optimize running analytics, but our infrastructure also allows use to run analytics across HDFS/Hadoop clusters if needed. For our in-memory architecture we provide an Entity Cluster Manager (ECM) to load individual, target, Entities across available resources. For instance, on our Amazon EC2 cluster, we have 3 small compute nodes dedicated to managing 10,000 Entities each. The ECM loads each Entity into memory from a persistent data store. In this case, we have a Mongo DB data store and an Elasticsearch data store that each contains different, generated Entity data. The ECM also provides concurrency for Entity change propagation. When a request is published to the system, the subject Entity is pushed to each compute node. The compute nodes then perform the compute analytics between the subject and each of the 10,000 target Entities on each node, in parallel. Each subject and target Entity result can then be aggregate to produce a single result or individual results can be handled by the system. For example, one request process may want to count all the similar Entities, in which case an aggregator is used to collect the individual results for summation. Conversely, one requesting process may want to store each, individual result, in which case an aggregator is not needed.

Our computation engine uses Calculators to perform the computations using our EAR model to define the elements to compute. We develop Calculators to evaluate individual Attributes, combinations of Attributes, or all Attributes of Entities. We define three types of Calculators – one each for Entities, Attributes, and Relationships. An Entity Calculator performs analysis between two complete Entities, such as for comparing two Person Entities to find

similarity. The Attribute Calculators are used to compute a solution of an Attribute that exists between two Entities, such as a Levenshtein name comparison. Lastly, we use the Relationship Calculator to perform analysis and traversal of Relationships that may exist between two Entities.

We have developed statistical probability inference analytics that use the results of computations to assign probabilities. Computation of probabilities, while a potentially resource intensive operation, is a simple mathematical function. The real power of probabilities is through the inference that can be established given the resulting calculation of a probability. An understanding of the mathematical function is necessary to ascribe the inference type that can be implied to a data set. There are many inference types and for this opportunity, we will focus on accuracy, similarity, and occurrence types. Statistical accuracy can be derived through probabilities computed against similar data from many data sources. For instance, given three data sources A, B, and C each with an entity X that contains several attributes, a computation is performed to match the attribute values across the three data sources. Suppose that both A and B compute that all the attributes are equivalent, but C computes that not all the attributes match. Given that each data source is equally trusted, you can infer that entity X on data source C may not be 100% accurate. Similarity probabilities produce measurements of the likeness between sets of data in a fashion similar to accuracy probabilities. Similarity computations rely on known configurations and other analytics to provide a determination of what is similar and what is not. Entity de-duplication across data sources is an example of the application of a similarity inference. Occurrence inferences are used to establish order and then predictions for the next iteration or outcome given some new data. Occurrence probabilities are applied to discover the likelihood, based on existing data that a behavior will occur. For instance, if you apply 3 states to the closing value of a company's closing market price – low, no change, and high – you can determine, based on previous performance, the likelihood that the next price will be lower, no change, or higher. This is also an example of the application of the hidden Markov model.

As mentioned earlier, we are able to use this computation platform to perform a variety of different analytics. We have developed Correlation analytics using DNA, fingerprint, name, and biographic information to correlate personal Identities. We have developed Aggregation analytics to group Entities, such as radios, triggers, wires, and explosive compounds, that make-up an Improvised Explosive Device (IED). We have developed Probability analytics using the hidden Markov model to explore closing stock market prices. Lastly, we have applied the Probability analytics with Pattern analytics to explore closing price state transitions over patterns of price changes, such as transitioning from a low price to a high price. For the purposes of this RFI, we can develop analytics using these established techniques and code to provide Entity correlation, confidence building, and other required computations.

2.6 Authentication, Authorization, Security

Authentication, Authorization, and Security are important to any large system. We look at AAS as a two-tier process. The first tier starts at the system level and provides control to who can access a system and what services a user is allowed to perform on that system. The second tier provides AAS at the data layer to define what data both a system and a user are allowed to access.

At the system tier, we suggest using SSL for any network access with PKI certifications for the user. We also recommend using a Central Authentication Service, such as the open source solution provided by jasig. This provides single-sign on capability to make it easier for a user to navigate large systems. We employ a Kerberos application with x.509 certificates to help

manage user credentials. Additionally we recommend utilizing the SPNEGO mechanism on top of Kerberos to support authentication. These services are easily integrated into an application and we recommend using the open source application provided by Spring Security.

At the data tier, we suggest using Validators when accessing data to ensure secure assembly of Entities. A user's credentials are provided to the Entity Assembler when building Entities with Validators ensuring that the Entity is assembled within the correct security protocols. This includes system level controls to ensure that a lower-security system does not promote higher-credentialed data. The EAR metadata contains security information and allows for granular construction of Entities at the Attribute level.

3 Reality Warp Software Capabilities

3.1 Infrastructure

Reality Warp Software has developed an open source, distributed analytic infrastructure, illustrated in Figure 2, that provides the system platform and key tools enabling distributed data processing. This infrastructure enables analytic developers by providing access to real-time and persistent data streams, efficient use of dynamic resources, and dissemination of results to an ever changing and evolving community of consumers. The infrastructure is architected around the publication/subscription pattern to establish a communication and data distribution platform that is tailored to processing real-time data streams, persistent SQL and NoSQL data stores. We recognize the ever-changing data landscape and provide analytic writers with a stable way to access data without worrying about the changes to the underlying data sources. The infrastructure addresses this by providing an abstracted data access layer that analytic developers use to access required data. The infrastructure inherently supports several architectural patterns for analytic execution. The distributed consumer pattern is used to provide the same data to analytics executing in parallel. The competing consumer pattern is used to execute many of the same types of analytics against streams of data in parallel. The event-driven consumer pattern is used to execute analytics when events and other stimuli occur. For this opportunity, we will leverage the benefits of this existing infrastructure for easier and timelier research and analysis of the efficiency of data processing and analytic results. Additionally, by leveraging this infrastructure, RWS will be able to more rapidly develop, prototype, and test analytics.

In building this infrastructure, we leveraged the experience gained from architecting and building the fusion infrastructure currently integrated with the DCGS-Army program. This insight and experience paved the way for the creation of an infrastructure that is tailored to running analytics under a variety of conditions. By leveraging open source and open standards, we have built-in compatibility with this and other standards-based infrastructures. We built a publication and subscription infrastructure that is both scalable and distributed. We utilize an open source application, Hazelcast, to provide one mechanism for distributed pub/sub, but the infrastructure is not limited to just Hazelcast. The Pirkolator is able to incorporate additional distributed pub/sub applications, such as Redis or Terracota, to provide the concept of a "Hub" for any pub/sub need. We incorporated the open source data transfer application, Apache Camel, to provide access to the most common network protocols, data formats, and translations. The benefit of this approach is more consistent, timely access to data and greater assurance of data integrity.

3.1.1 Data Ingest and Extraction

Data streams and data ingest is provide through another open source application from Apache called Camel. This product provides us with an extensive range of data protocols and formats, as well as transformation services to position the data in an optimized way for analytics. We utilize Spring Data, an open-source platform, to provide an extensible, flexible, and abstracted data management technology. This technology allows us to access to out-of-the-box implementations for several data modeling standards, such as Relational Database Management Systems (RDBMS), Big Data, and Key-Value Stores. Custom repositories allow us to develop for any data source need. Using an abstract data access layer coupled with the open source Spring Data technology, allows analytics to focus on their job without worrying about changing data source environments. This means no code re-writes should a data source or even data format change.

3.1.2 Data Modeling

Data modeling provides for describing data in a digital form. From the personal experience of the Reality Warp Software team trying to provide a common model to describe everything is not feasible. We believe that a better, more efficient and extensible solution is to develop models that a tailored for how the data will be used. Analytics have different modeling needs than persistent storage and a signals processing model has different needs than textual processing models. A common data model is used in cases where different systems need to work together to share data in which case the model is developed around the common model attributes.

Using multiple data models, tailored to specific needs also increases the extensibility as new data types and models becomes available. This flexibility manifests through addition of new fields to an existing model and through addition of new data models working in concert. Some data models are not extensible in which case new data models must be developed to work in concert.

For RWS, data modeling's most important benefit is in positioning data for optimal utilization. By using tailored data models and architecture of extensibility, we are able to position data for processing even as the data model changes. To implement this strategy, though, data modeling is not the only function needed. The addition of transformation techniques combined with our thinking about data modeling allows us a novel way to position data that can be leveraged across systems and under a variety of analytic processing.

3.1.3 Data Transformation

Transformation services ensure that data is mapped and translated into the optimized format used by each analytic. Filtering methodologies built into the publication/subscription construct provides even further data refinement for analytics. Geospatial, temporal, and source-based filters provide analytics with data granularity, reducing the size of the data haystack. Custom filtering is provided to allow each analytic with data filtering tailored specifically to its needs. Our transformation services utilize the open source technology, Apache Camel, to provide network connectivity, data routing and common data format transformation support, such as for JSON and XML. We provide additional hooks into this data transfer framework to provide dynamic mapping and transformation of incoming and outgoing data to support any object model needed. The benefits of this approach are extensibility, reusability, and flexibility. As the data sources within the community change, extensibility within the infrastructure provides new transformation services that are automatically integrated without affecting analytics or other data-aware processes. We are able to reuse existing data mappings and transformation services

where needed and the modular nature of new transformation services allow portability to other systems outside of the systems integration effort. The flexibility of the transformation services within the infrastructure allows for analytics to get the data in the format needed without having to understand the underlying ingest or originating format of the data.

3.1.4 Data Management

The data management component provides the necessary architecture for plugging in a variety of data stores by abstracting away the details of data management. We utilize open source technologies, such as Hibernate and iBatis, to provide data modeling, persistence, and query capabilities to structured RDBMS and Object-Relational Model (ORM) schemas. Unstructured data or large amounts of data are stored in a NoSQL data stores, such as Mongo DB, Accumulo and HDFS. It is important to allow for the integration of Big Data solutions and to provide the ability to run analytics over massive amounts of data in near real time. Multiple data source solutions are optimized for storing different types of data. With the ability to integrate these disparate data sources, analytics are optimized when performing complex operations specific to a data source, such as entity correlation and complex event detection.

Our infrastructure provides integrated metrics, pedigree, and data integrity. Metrics lend support to the systems administration command to maintain a healthy platform. Pedigree is integrated with the infrastructure to allow tracking of the processes that change or affect data. This builds upon the confidence, accuracy, and other data integrity definitions that help establish a reliable data tree of origination for each piece of original data, extracted entity, or transformed data. Having this tightly coupled relationship between pedigree and analytics promotes a strong chain of authentication when data verification is needed.

Our strategy to data within the infrastructure provides several benefits. First, it allows for dynamic data sources. As data sources come and go online, analytics don't have to change. Transformation services ensure that data is in the format needed for an analytic. Pedigree built into the fusion infrastructure further ensures data integrity, insight, confidence, accuracy, and traceability. With cloud technology and cheap compute cycles, transformation services allow data to take many forms and support many models. If you need to map data from one data model to another, transformation services provide an easy mechanism to achieve this goal. Every analytic can have its own optimized data model while at the same time support a common data model that is used across enterprises.

3.1.5 Distributed Analytics

On top of these open source applications, RWS provides easy to use architectural patterns that allow other processes, such as trending analytics or map/reduce functions, to get the data, events, or manual stimuli needed for a solution. We employ new techniques and technologies to achieve the near real-time analytics over Big Data sources that are the first step in developing a real-time, latent relationship discovery capability. We employ an in-memory data grid that consists of Java Virtual Machines (JVMs) running on any number of servers in the data center. An in-memory data grid is a distributed memory based data store that allows the data to be distributed across many servers. This is backed by disk storage with a write behind scheme for permanent storage. The benefit of this technique is to allow fast access to large amounts of data, even under systems with load constraints, while still providing the needed streams for real-time analytics. The in-memory data grid can be implemented with off-heap storage to utilize larger amounts of Random Access Memory than is allocated for the JVM, and to avoid the penalty of the garbage collection when items are deleted. This new technique to in-memory, distributed

data affords the capability to explore computational analytics against real-time data streams for producing real-time insights.

New technologies allow us to provide parallel, concurrent processing, in-memory and distributed. This can be much faster than traditional disk-based Hadoop/HDFS infrastructures. Additionally, concurrent technologies will promote new thinking. From past experience, a new way of thinking about an analytic was identified as an economy of scale for a United States Army Communications-Electronics Research, Development, and Engineering Center (CERDEC) analytic. The Communication Effects Simulator (CES) was identified as an important analytic, but was slow due to being a Windows command-line tool that must be manually run. To achieve benefits, it was possible to run the analytic in parallel and then combine the results. This allowed for a 40% improvement in working with the analytic.

For legacy analytics that are Hadoop-based, we can leverage the results in the same way we distribute data by taking the output of a map/reduce job and sending it to where it needs to go. We use a persistent cache within our in-memory infrastructure to provide fail-safe capability in the case of system failures. Additionally, we are able to leverage large disk-based systems, just as if they were in-memory sources when the amount of data is too large to keep in memory. This type of in-memory, distributed processing is at the heart of where we think future processing platforms are heading. We are integrated with another open-source application called Spring XD. This platform provides a "unified, distributed, and extensible system for data ingestion, real time analytics, batch processing, and data export". It provides out-of-the-box access to Hadoop operations, such as Map/Reduce and HDFS. This is just another example of a tool that is easily integrated into the infrastructure that keeps us at the cutting edge of analytic development.

3.1.6 Analytic Engines

Reality Warp Software's Determinator engine is a capability we developed to allow algorithms for Entity, Attribute and Relationship (EAR) extractions to combine with Association, Correlation, and Aggregation derivations to provide more resolved entities. Our engine utilizes the distributed and scalable nature of the Pirkolator's infrastructure to provide a real-time, dynamic scoring engine. Algorithms are developed and added to the engine using the same publication and subscription filtering in the Pirkolator to provide the most granular data available. An extensible scoring construct is used to provide the results that can then be used for automatic threshold alerting or other post-calculation analytics. Additionally, results are automatically distributed to any interested analytic that is attached to the infrastructure. Lastly, results can be automatically persisted to any attached, disk-based data store to provide further analysis using functions such as Map/Reduce. The Determinator engine provides a fast, extensible, and scalable resource that gives developers a fast and easy start to providing EAR extractions, associations, correlations, and aggregations.

Reality Warp Software's Trendinator engine is another capability we developed in conjunction with our infrastructure and Determinator engine to provide the grouping and threshold triggering useful during trend identification. Behaviorally, the Trendinator engine functions much the same as the Determinator engine with algorithms being added to the engine to provide scoring against any input data provided through the Pirkolator. The scoring results are run against threshold constraints and post-calculation analytics to provide automatic alerting. The Trendinator engine provides an additional function to the analytic developer through a grouping construct that can be used to build and establish related trend data. The grouping construct provides temporal and geo-spatial groups coupled with other custom, trend-specific

constraints, such as area of interest or time range limits, to automatically keep only the data elements that fit the trend grouping. The grouping constructs utilize the in-memory, distributed data grid of the Pirkolator for fast and efficient storage and retrieval of the data elements. The Trendinator engine provides developers with an immediate solution to quickly providing trend results against real-time data streams utilizing the available compute resources.

3.1.7 Analysis Visualization

Reality Warp Software leverages the Model-View-Controller architecture pattern to provide visualizations to the analyst. With our Transformation Services we are able to deliver representations of data in a format that can be easily viewed in any visualization platform, such as in browsers or standalone GUI applications, such as NASA World Wind. As data is produced within the Pirkolator, the Transformation library provides automatic conversion of the data into any view that may be in use. The Transformation library can also be used to manually transform produced data into view representations as needed. This strategy allows an extensible mechanism to provide data to multiple, different visualization platforms where required. As new ways to think about and visualize data are developed, transform modules can be added to support the new perspective. We utilize the open source JavaScript library, DOJO, to provide browser-based visualizations. DOJO provides advanced layout that can be optimized for the user experience. Additionally, DOJO provides data grids that can be hooked into a backend REST service supported by web sockets for easy generation and updates of data grids. DOJO provides advanced charting capability, such as pie charts, line charts, and bar charts, to further promote the user experience.

4 Past Performance

Our experience from acting as the Chief Architect deploying the fusion framework currently used by the Distributed Common Ground System – Army (DCGS-A) program provides us with a unique insight into the requirements of developing fusion and correlation capabilities. The DCGS-Army fusion framework was developed in 2008 to address the need of data de-duplication within the Army's centralized data model called the Tactical Entity Database (TED). From this experience was gained an understanding of the need for efficient resource management, dynamic data connectivity, and distributed information utilization. It was realized that many of the techniques and methodologies employed in the de-duplication effort could be applied across a variety of analytic processes, such as association, correlation, aggregation, and temporal and geo-spatial trending. This also lead to an understanding that providing an integrated infrastructure with core support for data management and analytic processing engines provides a robust and easily deployed enterprise application solution for addressing specific data needs. This was proven at the Army's Empire Challenge in 2011 where the Army's first, cloud-enabled, mobile biometric fusion platform was demonstrated.

Our experience on the Army's Windshear II program in developing and demonstrating the first fully functional, cloud-based, mobile biometric solution at Empire Challenge 2011 further illustrates our awareness of the challenging problems of today's networked world. Windshear II was demonstrated successfully to the Undersecretary of Defense after only 4 months of architecture and development. The platform consisted of a cloud-enabled server in a vehicle that housed a data management capability, biometric analytics for facial recognition, voice recognition, retinal patterns, and personal identification. These elements were all collected using a Motorola's Atrix Android phones. As the data elements were collected from the phones, associations against an HDFS-based graph data store providing processing to alert back to the

mobile devices when entities of interest were discovered. With a short timeline for demonstration and a relatively new application platform in Android, it was learned that open source solutions for data connectivity to mobile devices coupled with utilization of the then emerging HTML5 standard allowed a much faster path to integration of the mobile devices to the cloud-enable infrastructure. One of the stunning successes of this demonstration was the delivery of alerts not only to the mobile devices, but also through secure protocols to alert the higher echelons of the Army. This security driven effort provided the acknowledgement that an integrated, distributed, enterprise platform enables secure management of data.

These personal experiences brought to Reality Warp Software provide the key ingredients and foundation for its initial offerings and solution architectures. The Pirkolator, Determination Engine, and Trending Engine each provide a new, unique, and differentiated solution that provides the basis for RWS's approach to offering Big Data solutions. In providing these solutions as open source to the community, RWS's strategy is to provide industry specific analytic solutions built upon community driven and maintained tools. This allows for a focus on the real problems facing industry and not around the tools required just to get to the start of answering problems.