SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Reducing Investigation Time for Researchers, And Enabling Automated Configuration Updates by Digitizing Contextual Information

A Technical Paper prepared for SCTE by

Ajay Gavagal
Data Solutions Architect
Comcast Corporation
1800 Arch Street Philadelphia PA 19102
215-286-3078
Ajay_gavagal@cable.comcast.com

Mehul Patel
Distinguished Architect
Comcast Corporation
183 Inverness Dr West, Englewood, CO
303-658-7826
mehul_patel@cable.comcast.com

Sinan Onder
Vice President
Comcast Corporation
1800 Arch Street Philadelphia PA 19102
267-260-0964
sinan_onder@comcast.com

Table of Contents

List of Figures

List of Tables

1. Introduction

Contextual Information (CI) is an asset to every enterprise for its digitization to be achieved end-to-end. The digitization of contextual information and its changes can build a sustainable growth engine for development of new products and services. It can lower cost of software changes and lead to high productivity. To explore how we can achieve digitization of contextual information, in this paper we start with a scenario, explain the personnel involved, the kind of questions that get asked by these personnel, explain the problem that is in exchange of contextual information, and finally provide a possible solution for it. In this paper we also will explore the mechanics of contextual information and how it can benefit small or large use cases for data analysis and machine learning.

We start with a simple question that requires analysis into telemetry data, investigate the journey of how that question gets answered within an enterprise. Let us assume there is a device "X." This device is part of the internet protocol (IP) network. This device also has a capability to provide consistent telemetry such as its state. State here refers to the overall condition of the device, for example, is the device online/offline, and if the device is offline then reasons for being offline such as error conditions. Given this telemetry data, business owners can ask specific questions that can help them in business impacting decisions. Such as, evaluating device models from various vendors. A sample question might be to find out if a specific device model breaks down more than others. Given the value that can be achieved from answers to such questions using telemetry data, leadership allocates time and resources to capture data from device "X," ingest its telemetry into a storage layer and then appropriately make that data available for use cases such as alerting, analysis, and machine learning.

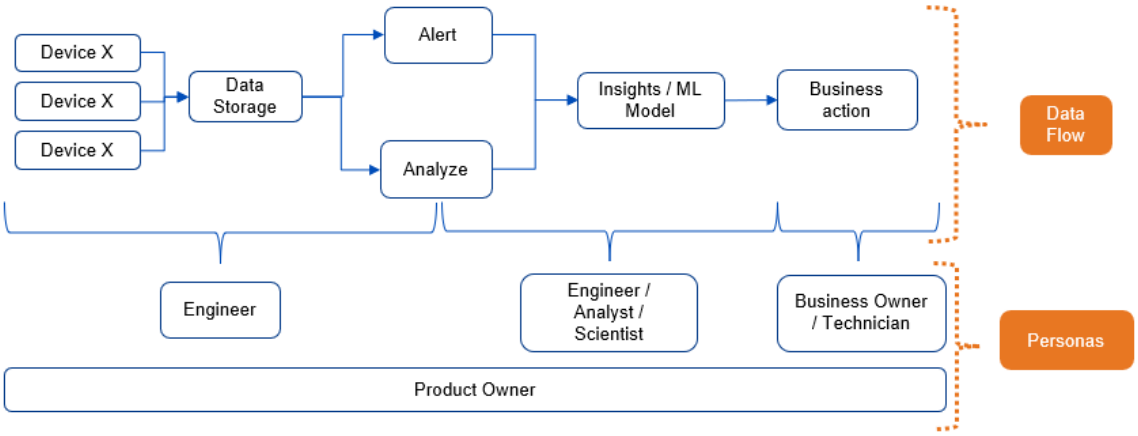

Figure 1- Sample flow of telemetry data and personnel involved

Note: Arrow direction represents data flow.

In the diagram above, which is common flow of data across enterprises, we see five different personnel getting involved in ingesting, storing, alerting, analyzing, and actioning on the insights. In the section below, each of the five personnel are analyzed. Moreover, and some of their motivations behind enabling insights from this data is highlighted.

2. Personnel & Motivation

Business owner(s) are motivated to provide high reliability in the overall service offered to customers. Here device "X" is a critical part of the IP network and should be made reliable and stable. Any disruptions in service due to device outages can negatively impact the customer experience. By choosing the right device models with the highest reliability they look to reduce disruptions. Their eventual goal is to have high customer satisfaction by providing reliable service. Business owners also often have limited budget to achieve the previously mentioned goals. So, they look to strike a fine balance between the highest reliability possible given their allocated budgets.

Product owners are looking to gain value through telemetry data and look for opportunities where the products or features they own, can save cost, or improve productivity. Product owners typically get measured on customer satisfaction, and hence want to see their customers succeed. To ensure customer satisfaction, the speed of deploying features is one of the many criteria's they focus on, in designing and engineering features. In large enterprises, due to pace of evolution, in many instances this criterion can become the highest priority, while other criteria can take lower priority.

Engineers are specifically interested in maintaining a reliable stream of data to consumption platforms with low latency. Any disruptions in data pipelines are a disruption to data flow where engineers are called upon to fix the issue. Engineers also must optimize data pipelines for low cost of computation while providing maximum data resolution. Given these opposing set of goals, namely cost vs. data resolution, they must make decisions on the resolution of data to fit either compute or storage requirements.

Data consumers on the other hand are keen on getting data that is reliable and of high quality to help them ease their job. Data cleaning is usually a large part of their projects. Given the velocity, variety, and volume of data available within an enterprise, it is safe to assume that data cleaning is a regular routine. However, data consumers are typically/usually demotivated by data cleaning since they view this as time taken away from more exciting tasks, which is to find insights in the telemetry data that provides an accurate representation of the real world.

Finally, network technicians want to see lower repeat issues. They want to ensure lower number of disruptions in the network. They depend on meaningful insights that help reduce avoidable maintenance and repairs. They are usually demotivated with technology and process that is more of a hinderance to their work than assist them, for example: cases where device recalls are made.

Below is a representation of the personnel and exchange of contextual information mentioned above:

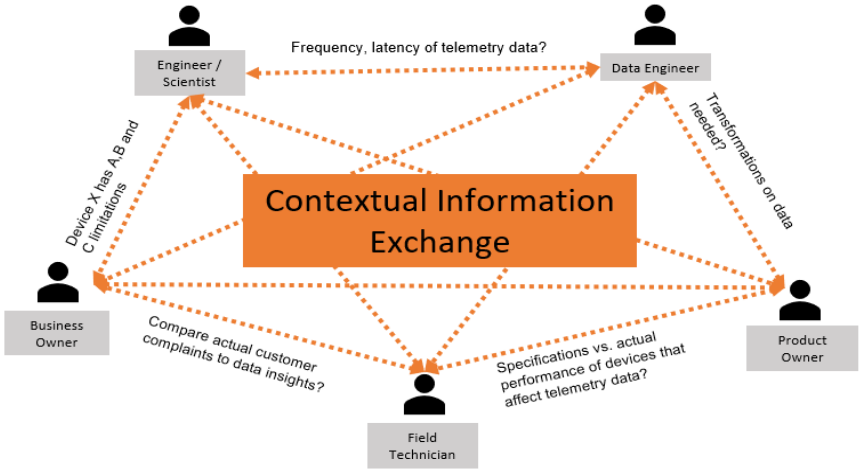

Figure 2- Exchange of contextual information across personnel

Now that we have seen the scenario and personnel involved, let us understand what we mean by contextual information and which of the above personnel might be responsible for such information and possibly where it might be made available in an enterprise.

3. Defining Contextual Information

We start by defining what we mean by contextual information. Contextual information can include but not limited to any metadata that provides context and perspective around telemetry data. For example, application configuration information used to set up the data pipeline, inherent limitations of data, data lineage information such as source and target systems for the entire data pipeline chain, data catalog such as list of data sets available, schemas and their versions, and many more. To better explain this with an analogy, consider contextual information as controls in the control pane of a manufacturing line and the telemetry data as the products being manufactured through the machinery itself.

Continuing with this analogy, in a manufacturing line, the controls (switches and dials) are usually set to certain values to ensure a constant production of the product. To either increase or decrease production that meets demand the demand, these dials in the control pane need to adjust. If the controls are not digitized and automated, then a human intervention is required every time to increase or decrease production. Also, considering the changes in the control values, they are not frequent, but too rare either. If these changes in the control values are not propagated or communicated to the entire manufacturing line, as an example to the packaging department or to the inventory department, the whole manufacturing line fails.

Similarly, consider an application that polls devices in the IP network at every A mins (frequency) to receive B resolution of data. In this case the telemetry data polled by the application is the product, while the polling frequency and the resolution detail polled are the controls in the control pane. If changes to the polling frequency or the resolution of data is not communicated by the application to downstream consumers, the whole data pipeline fails, causing cascading failures.

By nature, Contextual information is usually distributed in multiple systems. These systems may or may not talk to one another. Issues that prevent systems from communicating with one and other can include but are not limited to: mix of legacy and new applications, varying hardware/software platforms, varying cloud environments and varying feature capabilities. Let us consider the first example of telemetry data

from device X discussed in this paper and see how contextual information may be distributed among many systems.

Device vendors publish object identifiers for each device property and define functions, attributes in libraries allowing customers to consume and ingest such information. When telemetry data from the device is ingested, the schema of the data is registered in a schema registry to ensure changes/versions are tracked for field names, descriptions, data type changes. Information such as source, target, transformation logic and resolution of data are stored in application configuration. If data takes few forms such as raw, transformed and aggregated with retention periods increasing with lower resolutions of data then such information might be stored in the configuration of the application as well or in a catalog. Finally, the data catalog could be from well-defined tables or a simple list of storage paths once data is processed and written to a location. The ingestion application is registered in a service catalog that is maintained across the enterprise. Such a service catalog would contain information like, application name, purpose, source, destination, developers, support personnel etc.

Table 1- Sample CI and probable storage systems

Contextual Information	Probable Storage System
Object identifiers, descriptions, limitations, values etc.,	Libraries published by device vendors
Data resolution, polling frequency etc.,	Application configuration
Quality of datasets – raw, enriched, transformed, aggregated etc.,	Wiki pages, word documentation, schemas within DB's, segregation by paths in an object store
Data catalog	Schemas within DB's, segregation by paths in an object store, wiki pages
Application information	Service catalog
Application code and versioning	Version control, code repository
Change information	Tools that support CI/CD such as Concourse

4. Day 1 vs. Day N Scenario

Now that we have outlined the scenario, personnel with their motivations, defined CI, its nature, and the systems/applications where it might be stored, let us have a look at how a data consumer looking to generate alerts on telemetry data from device X discovers CI on day 1 (i.e., when designing the application) and day 2 to day N (when the application is deployed and needs to be maintained).

Initially a "discovery" phase to scrub, understand, document, if necessary, all the above-mentioned contextual information from various systems is required. We will discuss later why it is important to reduce the time involved in this phase. Since the focus initially is on generating alerts atop of telemetry

data from device X, the engineer might take the below attributes into consideration when designing an alerting application:

- Source of device X telemetry data
- Target to provide alerts once processed
- Various conditions within the telemetry data
- Values that signify these conditions
- Any exceptions to the conditions such as occurrences of NULL values or random values
- Applications that provide enrichment information to values within telemetry data (dependency)
- Data types of values
- Frequency of source data & target
- Volume of data at source & target
- Latency of data at source & target
- Modifications/transformations done to values to make them human friendly. For example: consider state information containing '1' for online and '0' for offline or vice versa. To make this more human friendly, one might be converted to "online" and 0 might be converted to "offline" to remove ambiguity for the data consumer
- Source and destination schema of the data

In a perfect world, all the above contextual information and their source systems are digitized and work cohesively (without any linkage breaks) to make the engineers' life easy to consume such information programmatically and design the alerting application in a fully digitized manner. However, it is far from reality since a lot of this information may not be digitized for easy consumption, for example, there might not be easily accessible application programmable interfaces (APIs) that provide information on all the modifications done to data values to make them human readable or a programmable interface that provides the frequency and resolution of the polling application. CI might be available within configuration of applications (as code) or free text documentation or audio/video clips (when recorded as training videos). In such situations, scrubbing through just free form information during discovery phase and then building configuration objects that contain contextual data and using this in alerting applications might be the accepted norm, but is not ideal and does not achieve digitization. This rapid prototyping and is also known as fast go-to-market strategies to justify the acceptance of non-digitization of contextual information. As previously discussed in the section, large enterprises have deadlines, priorities to meet, especially during the development phase since it costs time and money.

Since some of these CI inputs arrive from data discovery for the engineer to use them for building configuration objects manually for the alerting application, we have a breakage in the chain of digitization on day 1 (day 1 here is referring to the application deployed in production) itself. We are interested in showing how using non-digitized CI from day 1 causes a cascading set of problems on day 2 to day N (refer to figure 4). So let us assume that the engineer has configured required CI and successfully deployed the application that generates alerts atop of the telemetry data.

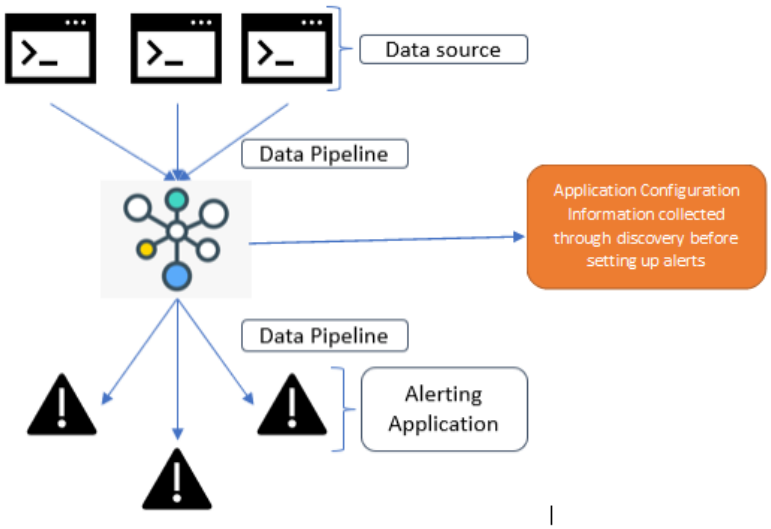

Figure 3- Alerting application deployed with contextual data within configuration

On day two, let us assume there is now a machine learning model set up to consume alerts from the alerting application. Now we have a two-layer dependance on the non-digitized contextual information.

Now let us also assume there is change in multiple attributes of the contextual information such as change in schema (data type changes), source system change, destination system change, data value changes (such as two or more values combined to one or a single value split into multiple), change in frequency of data, change in resolution of data etc. Since CI elements were not fully digitized on day 1, these can be treated as breaking changes for all downstream layers.

Below is a representation of cascading failure can occur on when CI is not fully digitized and transmitted.

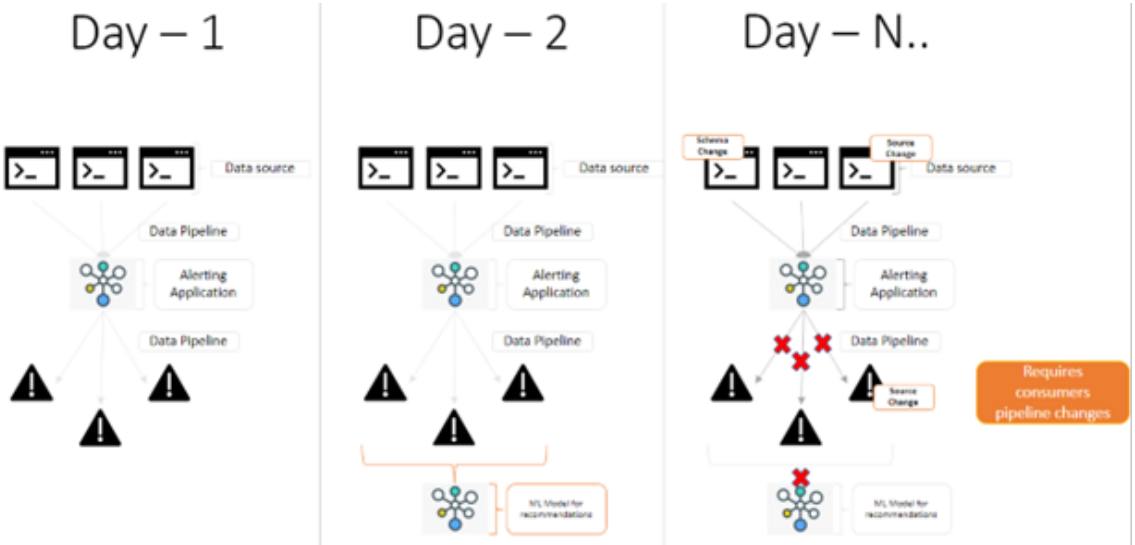

Figure 4- Cascading failure due to non-digitization of CI

If we do not address this problem of non-digitization of CI but kept adding more layers downstream that sends notifications to customers based on recommendations from the machine learning model, this cycle continues until, it collapses due to unmanageable changes in CI.

To give a scale, if there are A number of measurements available from a device going through B number of CI changes and there are C number of dependent data consumers downstream, then we can assume that there would be A*B*C number of changes to manage or intervention due to non-digitization of CI.

This problem only exacerbates if engineering groups are distributed between domains in an enterprise and reporting into various leaders. Groups having many individuals working on a single project with individuals distributed in various geographies. Then there are problems that the enterprise is not in control of such as vendors making devices smarter, software/hardware changes etc., the list goes on. And finally, the growth in volume, variety and veracity of data generation does not help either, where we are having to re-think our traditional well-defined storage formats with well-defined meta data stores to a more loosely bound structure where metadata is distributed all over the place.

If each time CI changes need to be reviewed, handled before re-deploying applications, then it is both very human centric and time consuming. This process is highly error prone and leads to lot of wastage in money and lowers productivity.

5. Proposal/Solution

One of the first things we observe as we review the list of CI elements is that it is diverse, sourced from a lot of systems/applications. Remember CI is meant to provide context and perspective to the telemetry data and its use and hence it includes everything that addresses this requirement.

With the evolution of software, we have an exceptionally good understanding of metadata needed to build applications, microservices and event-driven structures. However, we are lacking a standardized framework of required CI elements, its storage and transmission when cutting across pure software development activities and into more of the realm of data analytics/science activities that prototype and launch machine learning (ML) models that in turn assist software development. The CI elements of interest are varied between these two kinds of activities and hence an encompassing standardized set might be a good one to have to start with. We do not have a standardized list and that is for another day and another research paper. But in the absence of such a list, how should we produce one to address the problem in the short term at least?

One way is to see the kind of questions that get asked to an engineer/analyst/scientist when they present findings:

- What is the source of this data?
- Why did we not use an alternate source to answer the question on hand?
- What does a value mean? For example, in case of state information from a device X, what does online really mean? What does it signify?
- Do we know what is the source of truth between two similar datasets from two different applications?
- What assumptions have been made when generating an alert/analysis/ML model on telemetry data?
- What is the built-in latency into an alert/analysis/ML model? Can we reduce it?
- What thresholds were used to generate this alert?

- Who is the owner of this data?

The personnel asking these questions encountered doubt in the alerting mechanism, or analysis generated or the recommendations from the machine learning model. Based on this, it can be observed that this is just non-digitization of CI manifesting as the problem. The question is not asked to see if answer is known, but it is purely to get more context and perspective on the data. This data could have been transformed from pure raw telemetry to an alert, analysis, or an ML model that is helping drive the insight to help make a business decision.

With that in mind, let us start with a list that solves for these above questions and then build from there. It should be noted that this is not a comprehensive list of CI elements and should not be treated as one. We are merely trying to answer the above questions and providing a means to solve the problem. If these questions are not of high priority within your enterprise, then this list should be reformed to fit your enterprise needs.

Table 2- Sample list of elements in CDO's

Sample CI standardization	Description
Source system	Refers to the device, application etc., providing the telemetry data
Target system	Refers to the storage layer, application etc., where data is persisted after transforming the telemetry data
Data Frequency, Data Resolution etc.,	Explaining how frequently fresh data arrives, how deep can the data go etc.,
Data lineage	The source system/application where the data originated, got transformed, stored etc.,
Data description	Describes the telemetry data and can include field descriptions, value explanations, value limitations etc.,
Data schemas	Structure of the data
Data catalog	If more than one stream in the telemetry data, then list or catalog of those datasets
Exceptions	Exceptions when data might not be transmitted, transmitted with errors etc.,
Transformations applied	Changes to the datasets applied between source to destination

Next, we discuss the approaches we can take to solve for CI list to be transmitted through the entire lifetime of the data within an organization i.e., through various domains:

1. Ensure all contextual information is documented, distributed at regular intervals
    - Time consuming, human intensive and error prone if not for impossible.

2. Ensure all applications transmitting data provides interfaces for not only data, but also contextual information about the data (including metadata)
    - Well defined micro services powering every application within the organization with great programmable interfaces that data consumers can interact with to gain contextual knowledge, we can deem this problem solved.

However, this is not always the case since there will be standalone applications that do not have interfaces but are vital within enterprises. These applications came about as prototypes or as legacy service. Information technology (IT) is no more centralized where one central IT organization sets standards. The distributed nature of modern IT ensures that applications/systems are always built in a diverse way.

Start by bundling everything we determine as necessary contextual information required for the application in question and for downstream applications and build a contextual data object (CDO) and domain contextual data objects (DCDOs). Store and transmit this CDO in a way that it can be easily queried for changes. Also ensure that DCDOs can be enhanced by multiple application owners starting from the data producer, all the way to the last data consumer. Our solution proposal resolves around this third option. This is not necessarily new, since we have various flavors of this solution in usage within enterprises today, but the semantics of how we implement this might need another look.

A typical set of personnel working on making telemetry data smarter through alerts, analysis, or ML models, are distributed in various parts of the organization from functional, hierarchical, and geographical perspectives. Given such a distributed workforce, it would be wise to ensure that we start building the CDOs in bits and pieces throughout the domain and allow each data producer and consumer to decide the kind of CI they would like to add into the CDO. And when the CDO needs to be moved between domains, ensure that domain contextual data objects are used to transmit such information.

We see a way that contextual information can be digitized into an object that gets transformed as it moves along the enterprise systems and applications. First the data producer might produce a CDO as below:

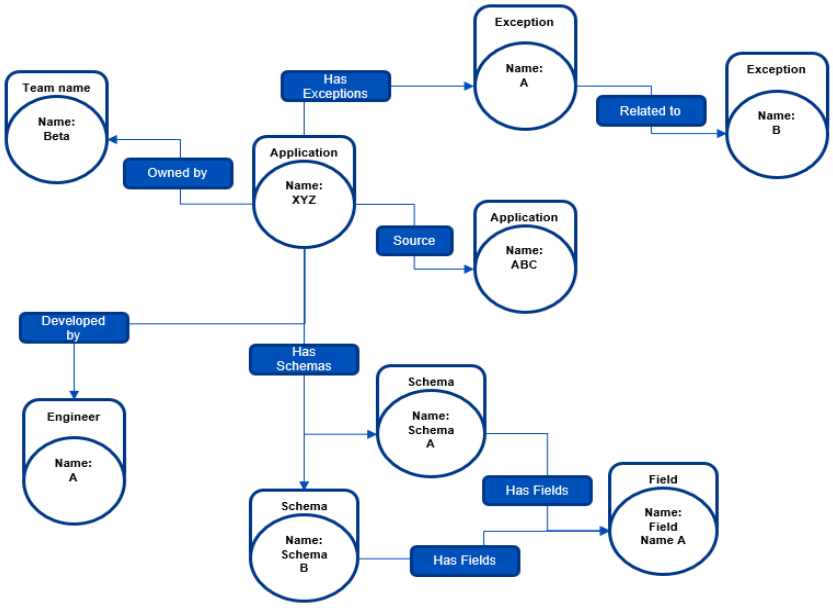

Figure 5- Contextual Data Object

Remember the above CDO is entirely up to the data producer to determine what to populate and what to leave out. In this paper we only define a common set of elements part of the CDO that is available to the producer to add into the CDO as part of the standardization. We want to ensure that this CDO is used internally by the data producer to drive changes to their application. This is an especially crucial factor, since self-use is the best motivator for the data producer to keep the CDO up to date and can help drive automation.

Now as this CDO makes its way through the organization to say, another engineering team that is in a different domain (in this case let us assume the elements in CDO and DCDO are same), that adds alerts atop of the "field name A." The engineering team enhances the DCDO with alert contextual information.

Observe that each of the DCDO elements connected to one another through a relationship. For example, the source data in application XYZ is FROM the application ABC. This signifies lineage information. The application XYZ was developed by team Beta and the engineer A was responsible for its development. Application XYZ has two exceptions named A and B, while B is related to A. Application XYZ also has two schemas: A and B. Both these schemas have common filed A.

Below is a representation of how a DCDO might evolve as it makes its way through the enterprise. All orange nodes and edges are added by a different domain:

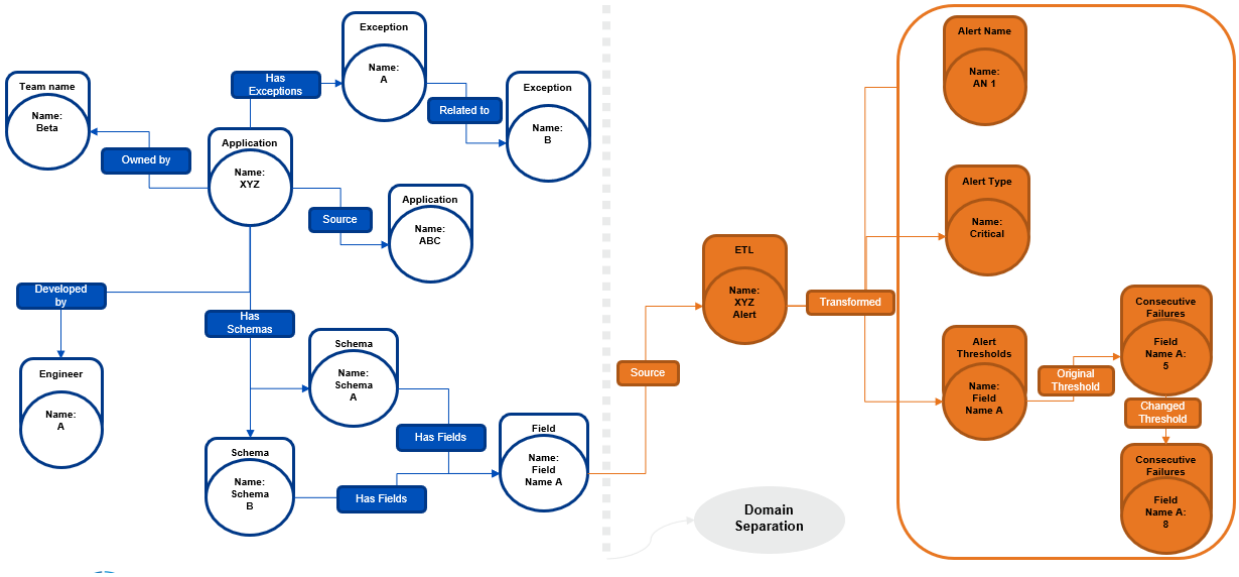

Figure 6- Buildup of domain contextual data object

Input source to an extract, transform and load (ETL) job that performs the function of generating the alert has source XYZ and transforms the data to alert name "AN 1" that has an alert type "critical" with the thresholds set as five consecutive failures originally. The team decided to change the alerting threshold to 8 at some point, which can just be another node signify the date of change.

We can also observe that application that performs the ETL is linked to the source team name and the engineer. This is immensely powerful. This is now helping connect elements that were not intended to be connected in the first place and allows for a lot of questions to be answered. This is the power of having DCDOs.

Expanding on the topic regarding the representation of DCDO and its evolution through the enterprise, an observation can be made that the nature of this data has an entity relationship during its formation. Given this nature of contextual information, two entities (nodes) being linked by a relationship (edge), using a graph storage mechanism for CDOs and DCDOs might be a good idea. As DCDOs evolve we start to build a knowledge graph across the organization giving the consumer access to powerful information that is hidden from many users.

The solution does not stop here. To help data producer teams and the consumer teams to create, interact with CDOs & DCDOs and build on it, we must provide them easy access ways to interact and modify them. Or else, we are just moving the breakage in digitization to a later point in time.

This is only the beginning of the solution. CDOs are built to assist the automation of functions within applications. When this automation controls are needed to be handed over to another domain, we need DCDOs or domain contextual data objects. These DCDOs can be directly used within ML models to ensure that recommendations from the ML models powers the application through automation vs. intervention.

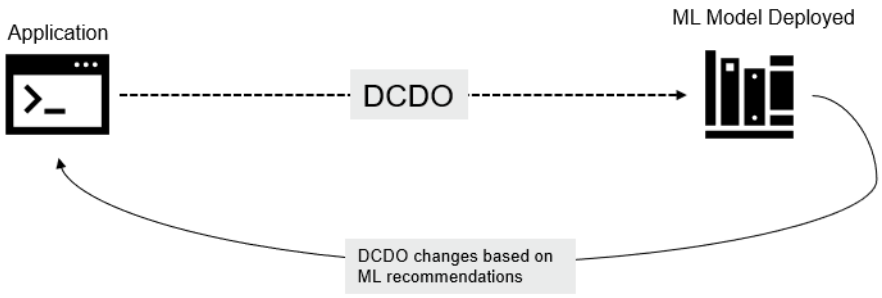

Figure 7- DCDO enabling a full cycle of automation

We should also enforce standardization of CDO/DCDO elements that the enterprise deems necessary and ensure this is adopted from the start for all new applications being developed. This is the hardest part since enforcement of a norm is hard within distributed structures unless it is used by the application.

The way to bring domains to onboard and share their CDOs as DCDOs into a common repository like an open WIKI within the enterprise is to help domains understand the time and cost savings from such an effort. The questions addressed earlier may seem trivial at first, but add in the mix of employee attrition, accidental deletion of information, modifications done for an ad-hoc request, and then having to invest time in discovering all this, the value will speak for itself.

6. Conclusion

Contextual information is vital and should be treated as an asset that needs to handle within an enterprise. Changes in CI is even more vital to enterprises and their propagation through DCDOs is essential for sustainable interaction between domains.

In this paper we first outlined a typical scenario for ingesting, persisting, and acting on telemetry data to derive alerts, insights, and ML recommendations. We also observed personnel and their motivation when interacting with telemetry data and its context. We saw how a cascading failure can occur when CI is not digitized and managed for propagation into downstream systems. Finally, we observed how digitizing CI in the form CDOs and DCDOs not only enables rapid transfer of knowledge between humans but also provides a methodology to handle updates programmatically in fully automated systems.

Digitization of CI also builds resiliency in automation against external factors not under direct control of the organization such as vendor device decommissions and recalls. This is also the case for applications where CI reside needs to evolve at a fast pace. Without effectively tracking changes to CI, models that power ML and automated systems built atop these assumptions can be ineffective. Such ineffectiveness in ML models lead to low return on investment (ROI) and sometimes even negative ROI. Assumptions underlying ML change rapidly with the evolution of network objects, applications, platforms, infrastructure, data pipelines, storage mechanisms and application configuration information. Digitizing CI and storing for easy access, ensures changes can be discovered programmatically by automated systems without intervention. Although we are not entirely solving for end-to-end digitization of all CI in an IP network, we provide means to show how it can be done.

Finally, imagine the amount of time required by engineers, analysts, scientists in discovering contextual information every time a change needs to be made. It is both wasteful and lowers productivity. Digitizing CI as mentioned in this paper by starting to create CDOs for internal use by application owners and DCDOs when transmitting to different domains saves money and time.

Abbreviations

API	application programming interface
CDO	contextual data object
CI	contextual information
CI/CD	change integration and change deployment
DCDO	domain contextual data object
DB	database
ETL	extract, transform and load
IP	internet protocol
IT	information technology
ML	machine learning
ROI	return on investment