# Service Application Knowledge Graph and Dependency System

Hanzhang Wang, Chirag Shah, Praseeda Sathaye, Amit Nahata, and Sanjeev Katariya
Applied Research and Marketplace Architecture
eBay, USA
hanzwang@ebay.com, chiragshah@ebay.com, psathaye@ebay.com, anahata@ebay.com, and skatariya@ebay.com

***Abstract*— Service architecture adoption is widespread and brings many benefits, such as agile development and immutable infrastructure. However, it's hard to govern and understand the vast service ecosystem as each application evolved independently and differently (e.g., features and development methods) from each team. In this paper, we present an approach to model and process application ecosystem as a knowledge graph. The application knowledge graph can help with architectural visibility, operational efficiency, and developer productivity.**

***Keywords—Service Ecosysystem, application knowledge graph, dependency graph.***

## I. Introduction

Today, eBay's production system has over 6,000 application services. It's challenging to achieve efficient development, due to various reasons like complex systems, not having proper knowledge about internal consumers and lack of documentation. In addition to this, platform upgrades, data-center exit, compliance and governance, microservices architecture adoption consumes huge efforts. Hence, gaining visibility inside service ecosystem is critical. In this work, we propose a novel approach to provide better visibility, which can help us improve operation and engineering efficiency (e.g., speedy information retrieval); support domain-driven design [1]; and recommend improvements to the above mention challenges.

Hence, we conduct multiple research pieces and build a system to solve the following high-level problems:

*Visibility*: Limited visibility inside the eBay ecosystem which impacts diagnostic-ability. For example, unable to detect the architectural issue such as inappropriate dependencies for software and/or hardware; or unable to envision the eBay infrastructure and ecosystem with customized search.

(Operational) *Efficiency*: No service or architectural metrics to systematically and algorithmically provide suggestions to improve operational efficiency.

*"If you can't measure it, you can't improve it."*

*-Peter Drucker*

(Developer) *Productivity*: There is no centralized system for efficient information retrieval. In general, operators or developers have to go through many different tools to retrieve information.

## II. Approach

We propose a vision to connect data sources and break the boundaries between domains; Figure 1 explains the high-level concept and possibilities.

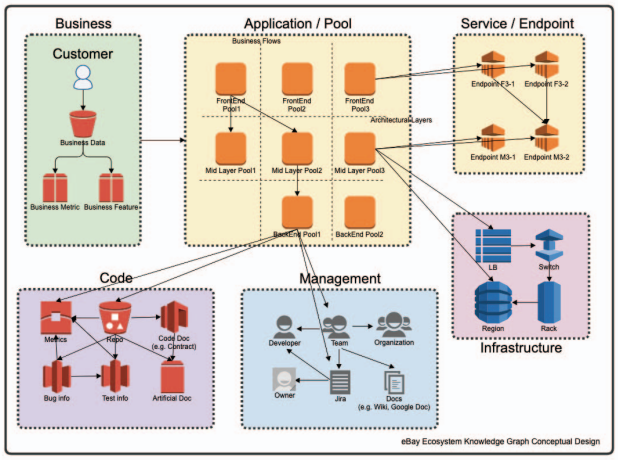


Fig. 1. Knowledge Graph Concept Across eBay Domains

### A. Knowledge Graph Construction

To address the problems, Figure 2 explains high-level steps to construct, enhance and use the knowledge graph. We build the graph with real-time metrics, business features, and operational metadata.

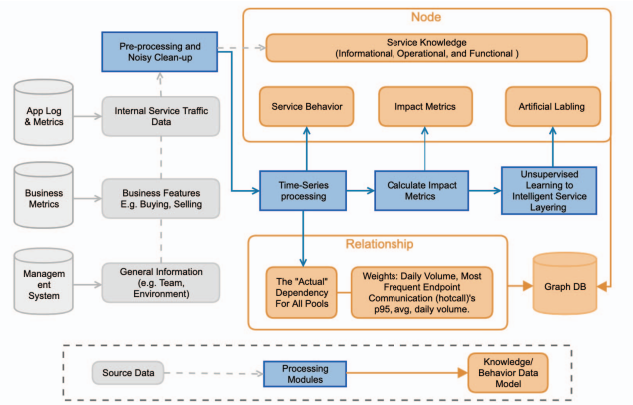


Fig. 2. High-level Overview to Build Knowledge Graph

978-1-7281-4136-7/19/$31.00 ©2019 IEEE
DOI 10.1109/ASEW.2019.00043

### B. Behavior Metrics and Intelligent Layering

In this step, we calculate the proposed application metrics and apply machine learning algorithms to automatically cluster the applications. In the literature, related works of application metrics are limited to QoS [3] or code quality [4,5].

We propose new metrics which measure the application popularities, based on run-time dependencies and real-time traffic volume. We conduct the experiment for all eBay applications.

For any given application (cluster), $p_x$ may has the dependencies:

$$(P_{x_i})\text{- }[D_{x_i}] \text{ -> } (p_x) \text{ - } [D_{x_o}] \text{ -> } (P_{x_o})$$

The inbound dependencies: $D_{x_i} = \{d_{x_{i1}}, d_{x_{i2}}..d_{x_{in}}\}$ from the application clusters:$P_{x_i} = \{p_{x_{i1}}, p_{x_{i2}}..p_{x_{in}}\}$; and outbound dependencies: $D_{x_o} = \{d_{x_{o1}}, d_{x_{o2}}..d_{x_{om}}\}$ to $P_{x_o} = \{p_{x_{o1}}, p_{x_{o2}}..p_{x_{om}}\}$.

For every dependency $d_x$ is associated with attributes: $d_x^t$: The count of target endpoints of $p_b$ are being called from $p_a$; $d_x^s$: The count of source endpoints of $p_a$ are calling to $p_b$; $d_x^v$: The daily average volume between all end points; $d_x^l$: The 95-percentile latency for most frequent call between all end points.

We define the following metrics to measure the connectives between eBay application clusters:

$$Struc_{Essential(p_x)} = \frac{Log\left(\sum_{d_x \in D_{x_i}} d_x^s\right)}{Log(MAX_{d \in D}(d^s))} + \frac{\dim(D_{x_i})}{MAX_{D_i \in D}(\dim(D_i))}}{2}$$

$$Struc_{Consum}(p_x) = \frac{\frac{Log\left(\sum_{d_x \in D_{x_o}} d_x^t\right)}{Log(MAX_{d \in D}(d^t))} + \frac{\dim(D_{x_o})}{MAX_{D_o \in D}(\dim(D_o))}}{2}$$

The $Struc_{Essential}$ and $Struc_{Consum}$ are trying to measure the structural inbound and outbound impact of a clusters. To extend and consider traffic volume (the service behaviors):

$$Essential(p_x) = \frac{Struc_{Essential(p_x)} + \frac{Log(\sum_{d_x \in D_{x_i}} d_x^v)}{Log(MAX_{d \in D}(d^v))}}{2}$$

$$Consumption(p_x) = \frac{Struc_{Consum}(p_x) + \frac{Log\left(\sum_{d_x \in D_{x_o}} d_x^v\right)}{Log(MAX_{d \in D}(d^v))}}{2}$$

$$Interoperability(p_x) = \frac{Essential(p_x) + Consumption(p_x)}{2}$$

The$Essential$,$Consumption$, and $Interoperability$ metrics are designed to define the popularity of a given clusters. All metrics are nominalized after getting calculated.

We calculated above metrics for all eBay clusters and used K-means[6] and canopy clustering[7] to cluster all services and based on their popularity scores and two sets of clusters are: "customer-facing oriented", "domain services", and "backend"; "low-activity", "high-activity", "mid-activity (frontend)" and "Mid-Activity (backend)". Interesting results are also discovered such as around 80% of the clusters are labeled as low-activity.

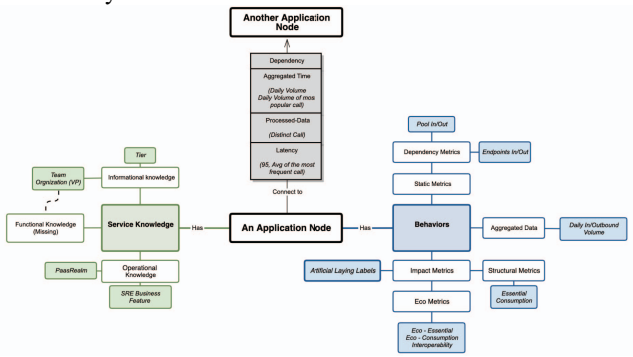


Fig. 3. Knowledge Graph Design

### C. Graph Search and Result Visualization

To tackle the *Productivity* issue, we build a complete batching system which fetches data from different sources and build the knowledge graph automatically. From a research perspective, we build an intelligent graph search that dynamically generates a query to explore the knowledge graph (which includes service metrics, and intelligent layering).

Figure 3 catpures our POC graph design and Figure 4 (application names are grammaticalized) shows a motivating example of graph search: high criticality clusters from a team (blue) are calling low criticality tier application clusters.

The visualization focuses on delivering readable and rich information in a directed graph format: the edge thickness represents edge property (e.g. volume); Node size represents the behavior metrics; and team or organization can differentiate by different colors (e.g., yellow is one domain team)

Fig. 4. Knowledge Graph Search Engine

## III. RESULTS

We calculated metrics and service layering in section 3.2 on over 6000 eBay production applications. The initial results with popularity metrics and automated clustering are manually validated by three senior architects. We discovered that some of

applications are running under incorrect availability zone which can impact operational performance and uptime. With 65-80% accuracy (manual validation for 100 applications), our results show around 10% of the "high-activity" applications fall into above mentioned condition.

Since our approach towards a functioning system, the main outcome is a dependency system powered by a graph DB (Figure 5).

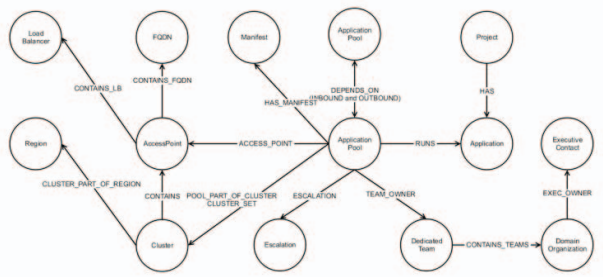


Fig. 5. Dependency System Graph Model

The functionalities and use cases enabled by our system named galaxies, include: top-down and bottom-up view of application along with the dependencies and increased accuracy; enrich data to enforce application compliance; governance with clear ownership details; and operational performance recommendations.

## IV. CONCLUSION

In this paper, we presented our approach for building eBay knowledge graph and use cases: customizable visualization, application metrics, intelligent service layering, and graph search. As the next step, we plan to enhance the graph to support site anomaly detection by presenting suspected events on the graph with full causality details of each incident. We also plan to extend this graph to include service API metadata, which will enable service layering, recommendation and clustering.

## REFERENCES


[1] Evans, E., & Szpoton, R. (2015). Domain-driven design. Helion.

[2] E Li, H., Li, S., Sun, J., Xing, Z., Peng, X., Liu, M., & Zhao, X. (2018, September). *Improving API Caveats Accessibility by Mining API Caveats Knowledge Graph*. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 183-193). IEEE.

[3] Wang, H., Kessentini, M., Hassouna, T., & Ouni, A. (2017, June). On the Value of Quality of Service Attributes for Detecting Bad Design Practices. In *Web Services (ICWS), 2017 IEEE International Conference on* (pp. 341-348). IEEE.

[4] Goyal, P. K., & Joshi, G. (2014, February). QMOOD metric sets to assess quality of Java program. In *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on* (pp. 520-533). IEEE.

[5] Boehm, B. W., Brown, J. R., & Lipow, M. (1976, October). Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering* (pp. 592-605). IEEE Computer Society Press.

[6] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100-108.

[7] McCallum, A., Nigam, K., & Ungar, L. H. (2000, August). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 169-178). ACM.