



USING **GRAPH THEORY & NETWORK SCIENCE** TO EXPLORE YOUR MICROSERVICES ARCHITECTURE

Nicki Watt - CTO
@techiewatt



please

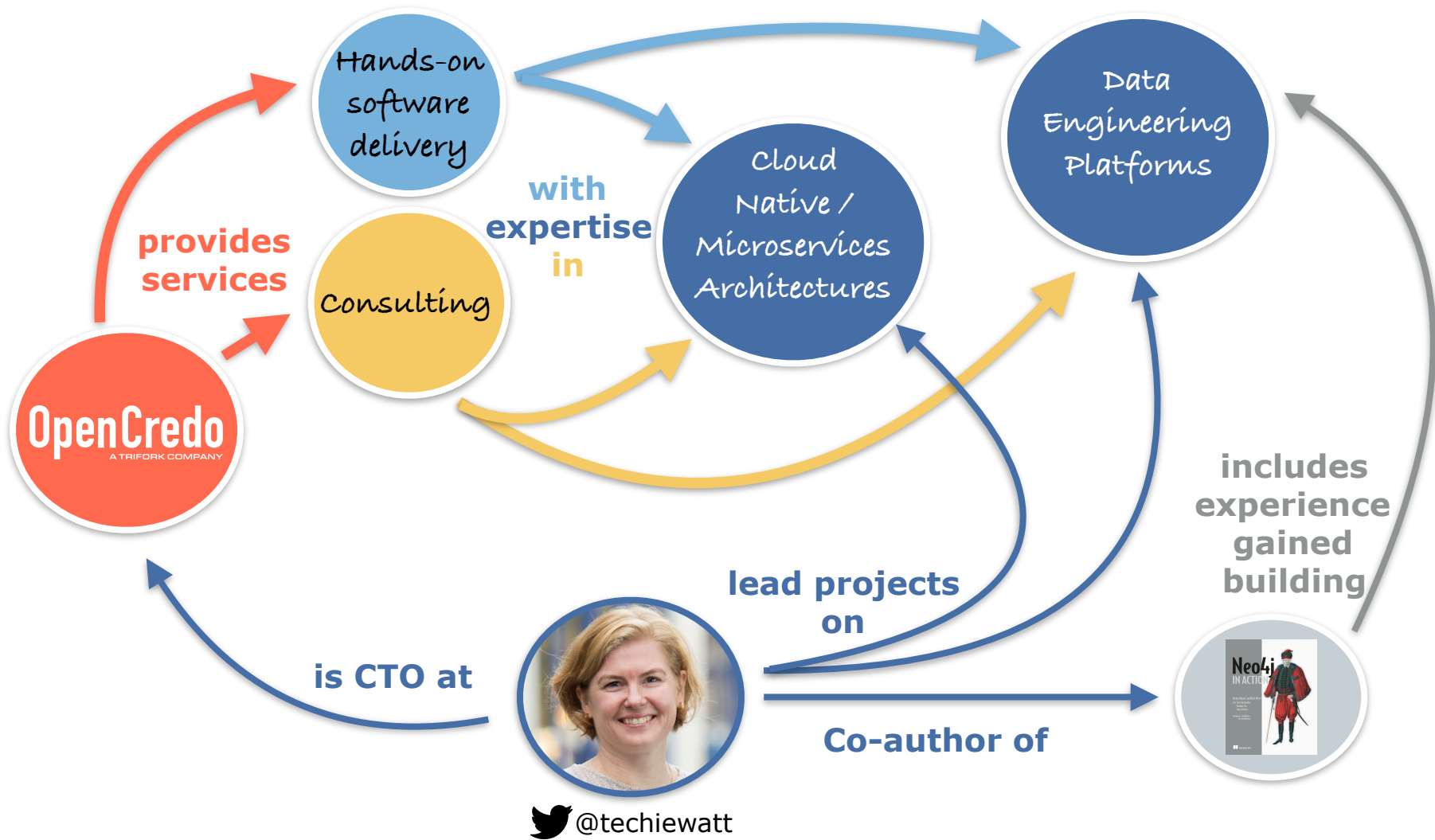
Ask questions
through the app



Rate Session

Thank you!







AGENDA OVERVIEW

- **INTRO & HYPOTHESIS**
- **A BIT OF THEORY**
- **MICROSERVICE EXPLORING**
- **SUMMARY**



INTRODUCTION



INTRODUCTION

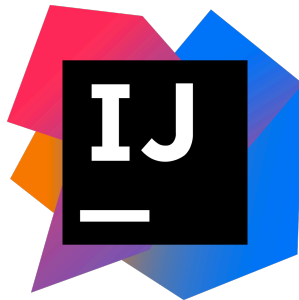


Graph theory is already being
used to drive efficiencies in, and
produce more reliable software
systems





INTRODUCTION



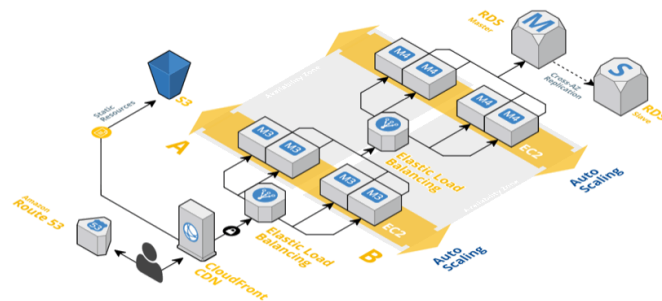
Code: Analysing software
package & license dependencies



INTRODUCTION



*Credit: Paul Hinze (HashiConf 2016)
Applying Graph Theory to Infrastructure-as-code*



HashiCorp

Infrastructure: Reliably managing
and updating cloud and other
infrastructure-as-code resources



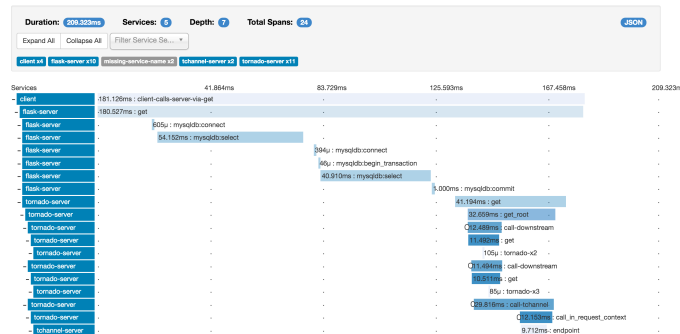
INTRODUCTION



JAEGER



OPENTRACING



Ops & Monitoring: Troubleshooting, Distributed Tracing & Latency analysis



INTRODUCTION



Can we also use it to help
inspect and drive us towards
improvements in our
microservices architecture?



HYPOTHESIS



HYPOTHESIS



General Hypothesis: Data Driven Architectural Improvement

You can extract metrics and KPIs from a
microservices architecture using graph theory

AND

use these to gain insight into the structure
and characteristics of your microservices
architecture



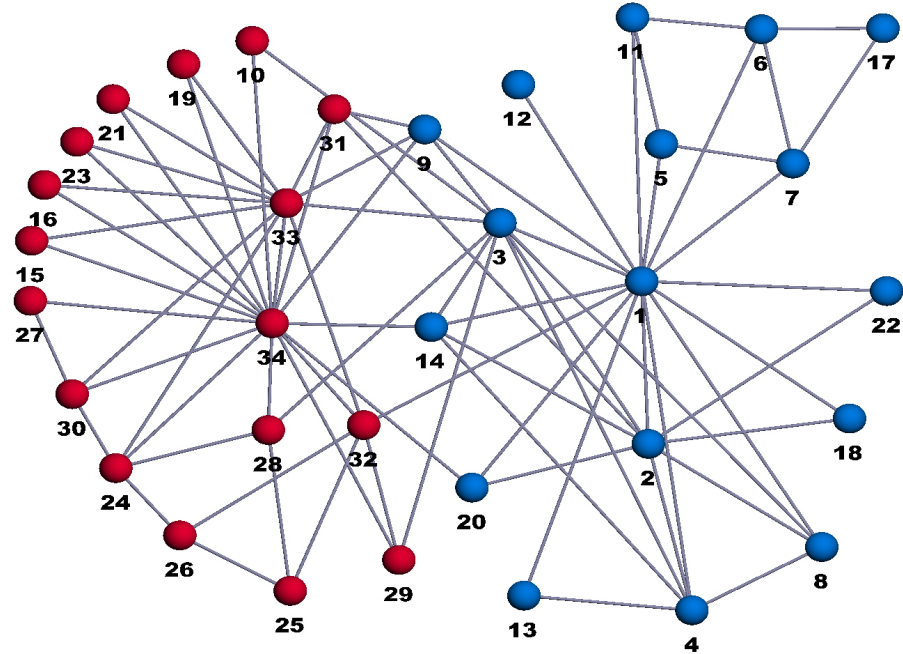
For this talk ...

Can we use these metrics to detect bad microservice architectural smells and anti-patterns like a tightly coupled architecture (distributed monolith)



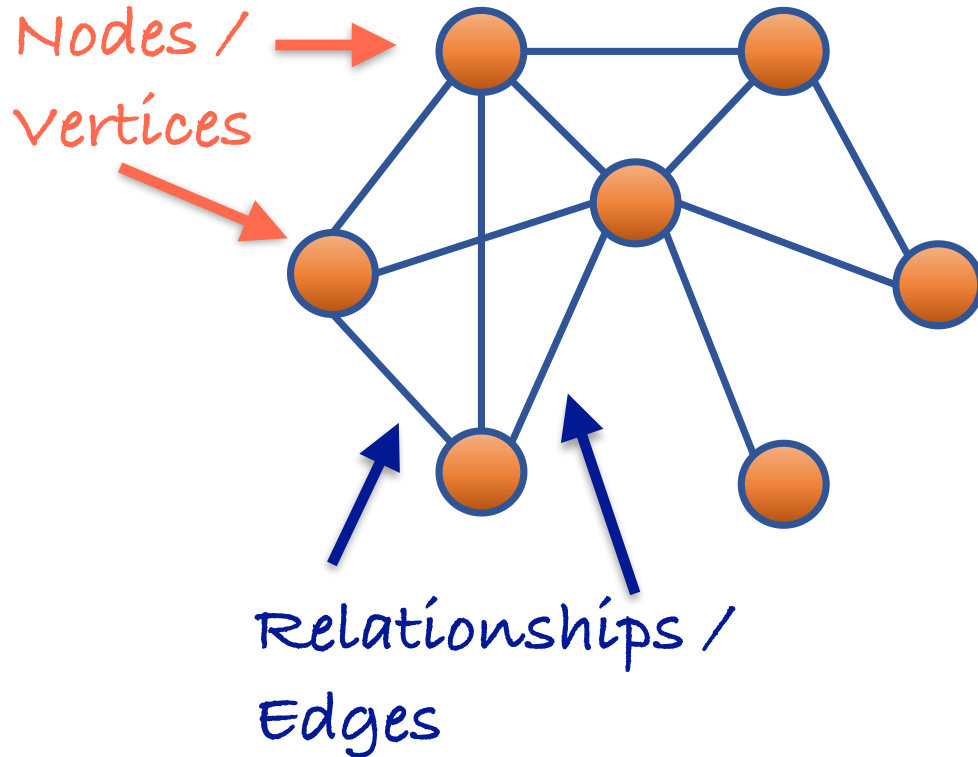


A QUICK BIT OF THEORY





GRAPH THEORY 101



A graph is a way to formally represent a network, or collection of related objects, in a mathematical way



GRAPH INSIGHTS 101



- **Graph Analytics**

- ***An Action Performed:*** The act of analysing connected data - using any appropriate graph-based approach or tools (visualisations, queries, statistics, algorithms)

- **Graph Algorithms**

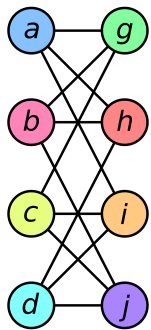
- ***Programmable process or set of rules:*** Leverages the mathematical properties of graphs to explore, classify and interpret connected data. A subset of tooling used to do graph analytics.



GRAPH INSIGHTS 101



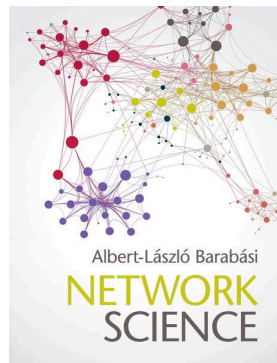
GRAPH THEORY



Roots date back to 1786, includes study of:

- Abstract theoretical graph forms (e.g. random graphs, trees, directed graphs)
- Graphs of any size

NETWORK SCIENCE



New academic field, circa 21st century includes study of:

- Real-world representations in order to understand the universal properties of networks (Biological, social, transport)
- Large, complex networks



GRAPH INSIGHTS 101



“Based on the mathematics of graph theory, graph algorithms use the relationships between nodes to infer the organization and dynamics of complex systems. Network scientists use these algorithms to uncover hidden information, test hypotheses, and make predictions about behavior”.

- Graph Algorithms by M Needham, A Hodler



**Show me some of this
graph theory /
network science
stuff ...**



HIGH LEVEL ALGORITHM TYPES



**Path
Finding**

Centrality

**Community
Detection**



HIGH LEVEL ALGORITHM TYPES



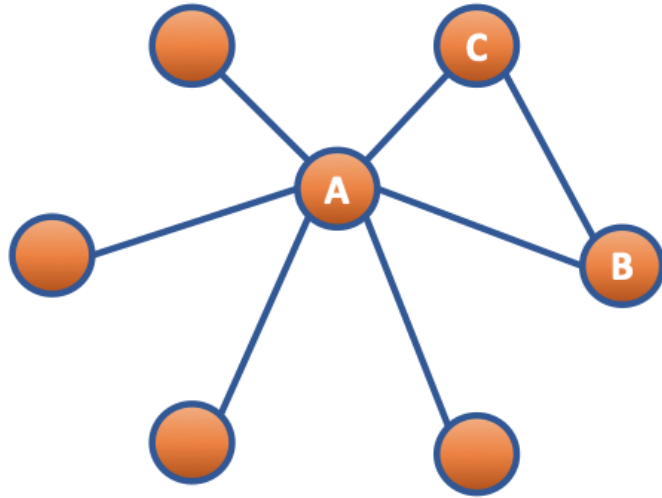
**Path
Finding**

Centrality

**Community
Detection**

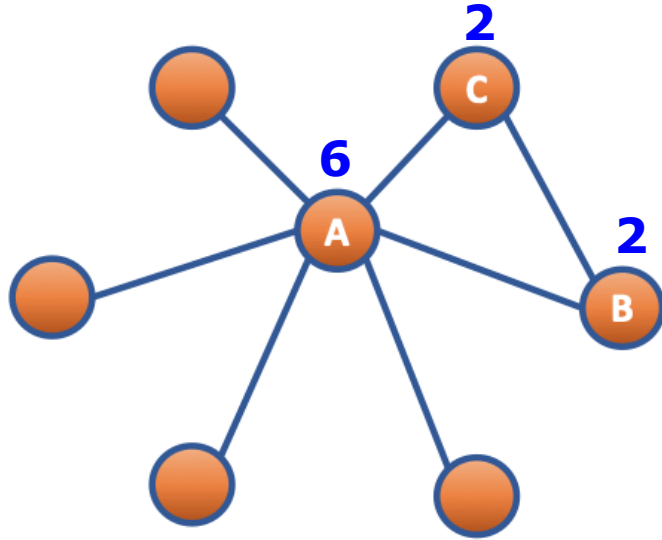


1 DEGREE



How connected is
a specific node?

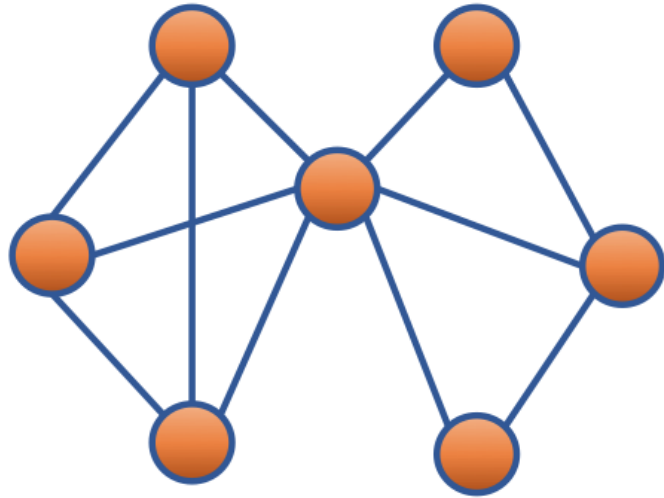
1 DEGREE



How connected is
a specific node?

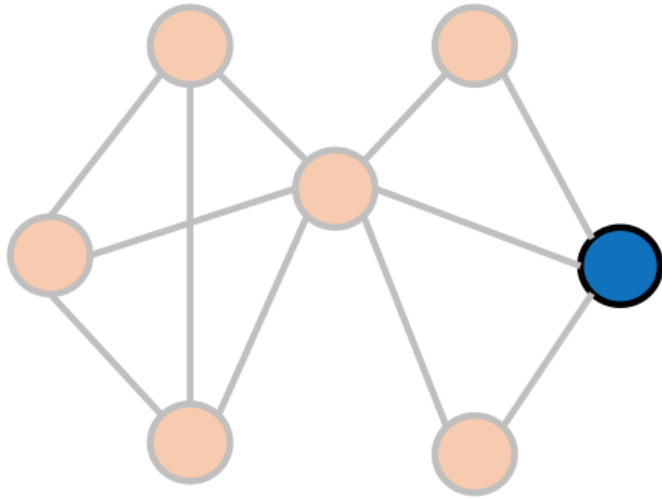
A is more highly
connected than
B and **C**

2 CLUSTER COEFFICIENT

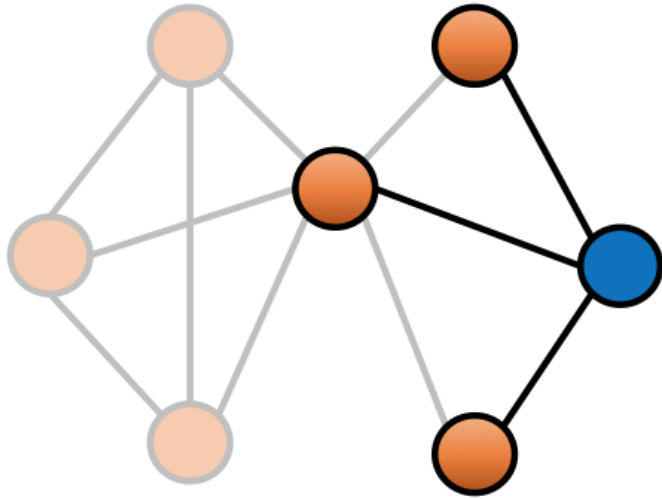


How tightly is a group clustered, compared to how tightly it could be clustered?

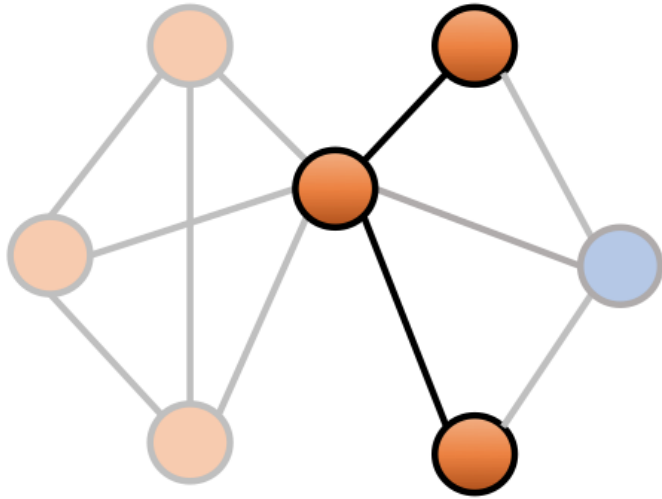
2 CLUSTER COEFFICIENT



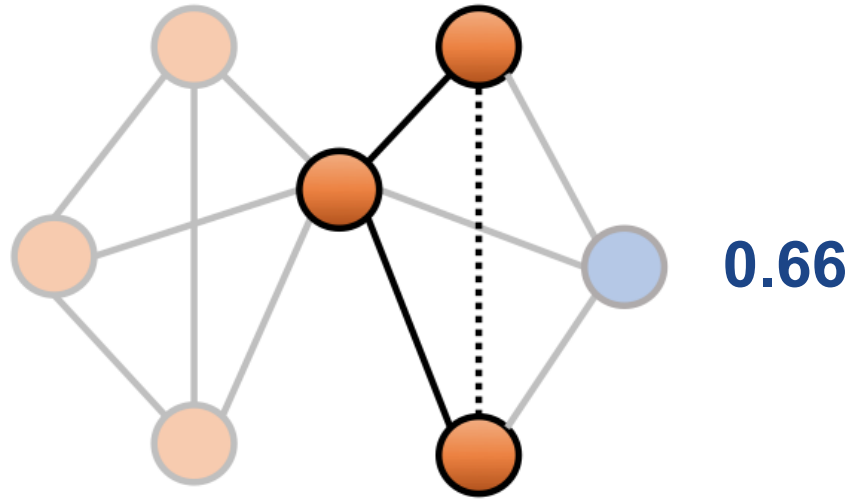
2 CLUSTER COEFFICIENT



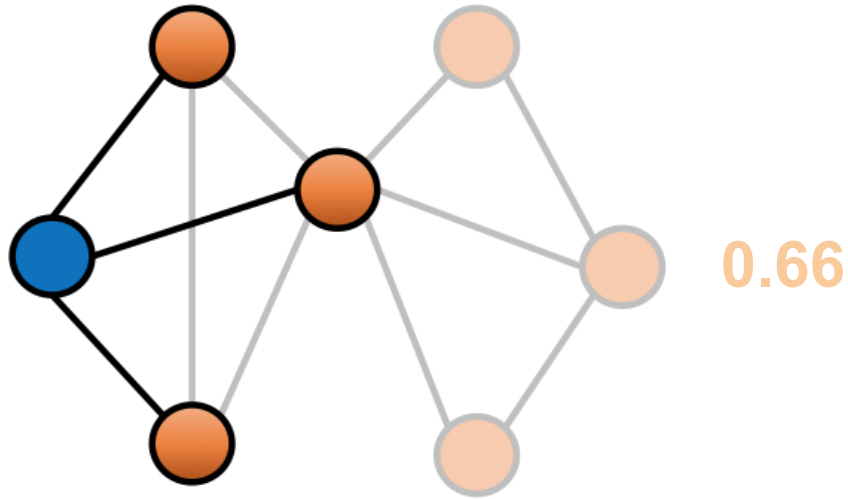
2 CLUSTER COEFFICIENT



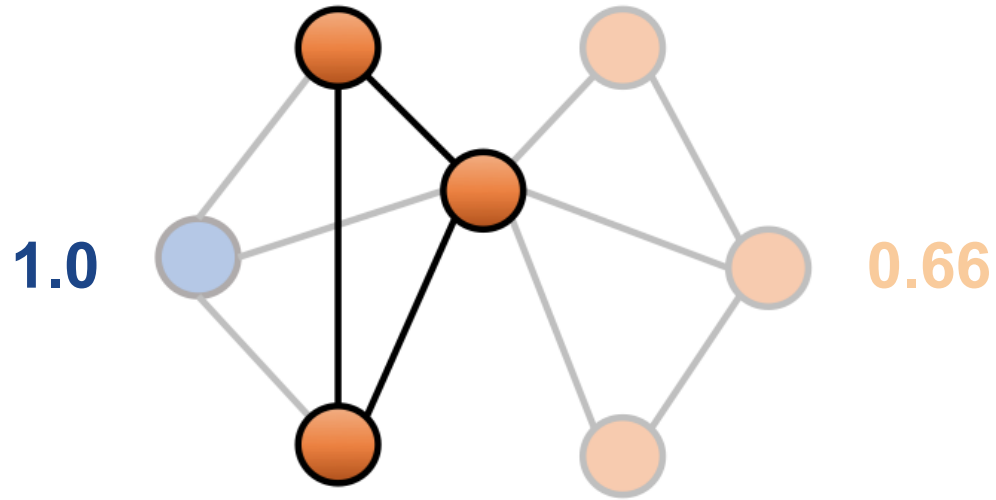
2 CLUSTER COEFFICIENT



2 CLUSTER COEFFICIENT

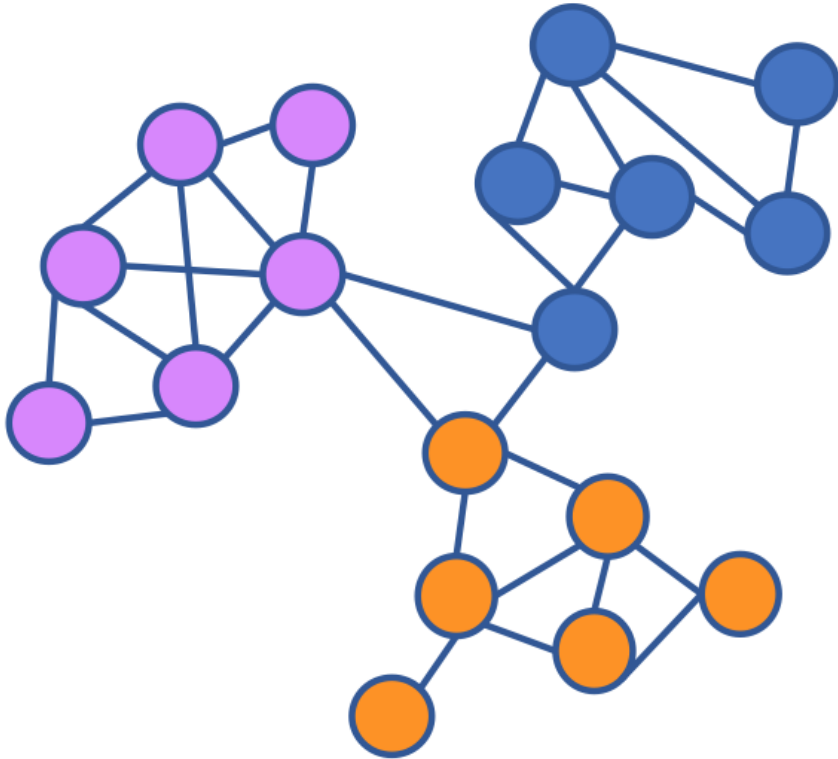


2 CLUSTER COEFFICIENT





3 COMMUNITY DETECTION



Used to find related communities,
uncover groupings,
and quantify the
quality of groupings



DIVE IN! **EXPLORING** **MICROSERVICES** (AS A NETWORK)

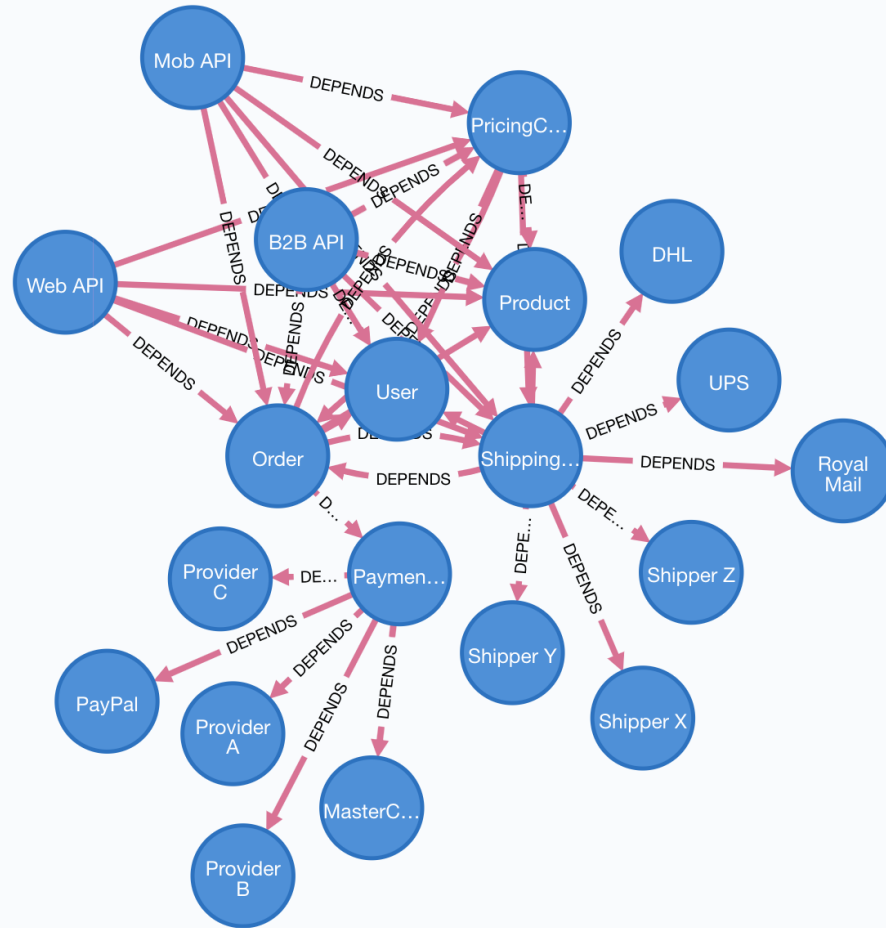


#1

A “Microservice” Architecture

(v1.0)

V1.0

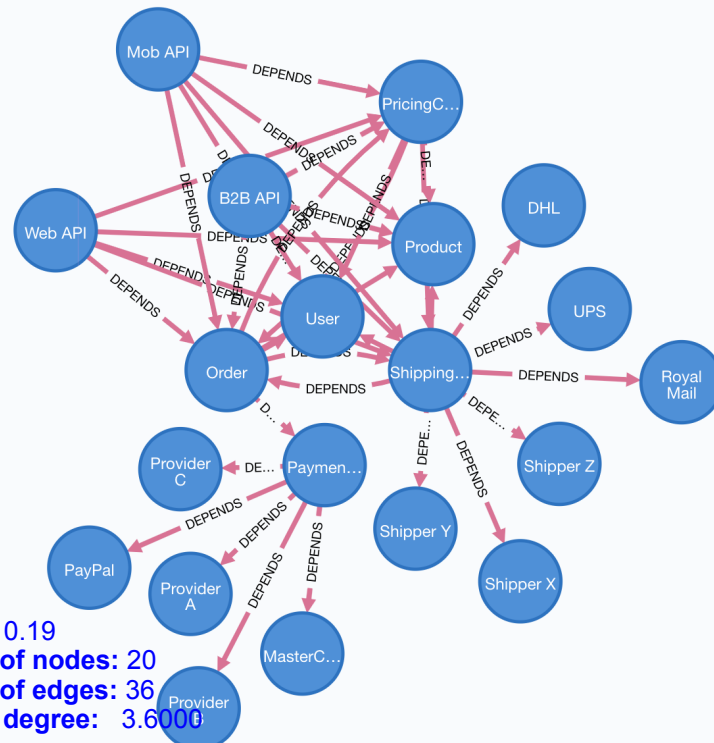




V1.0 - NETWORK STATS



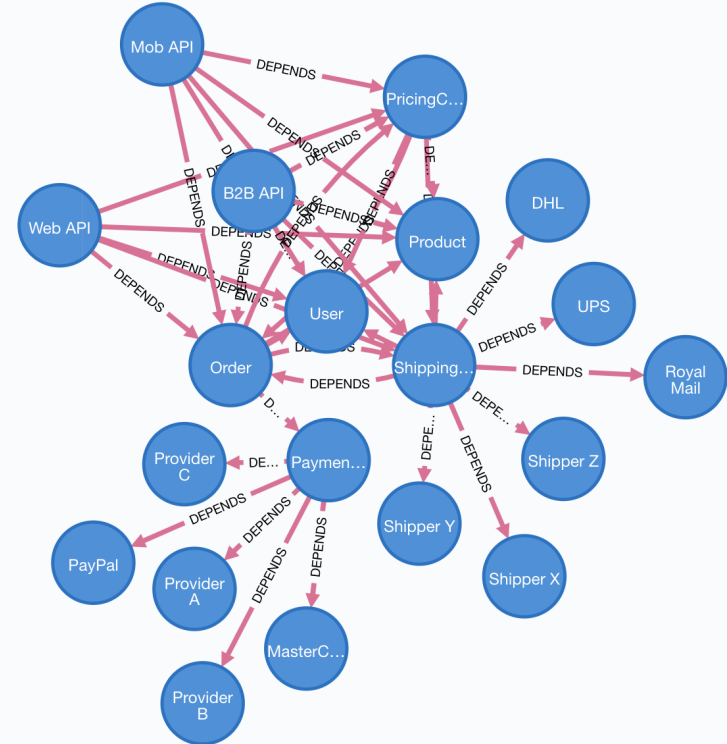
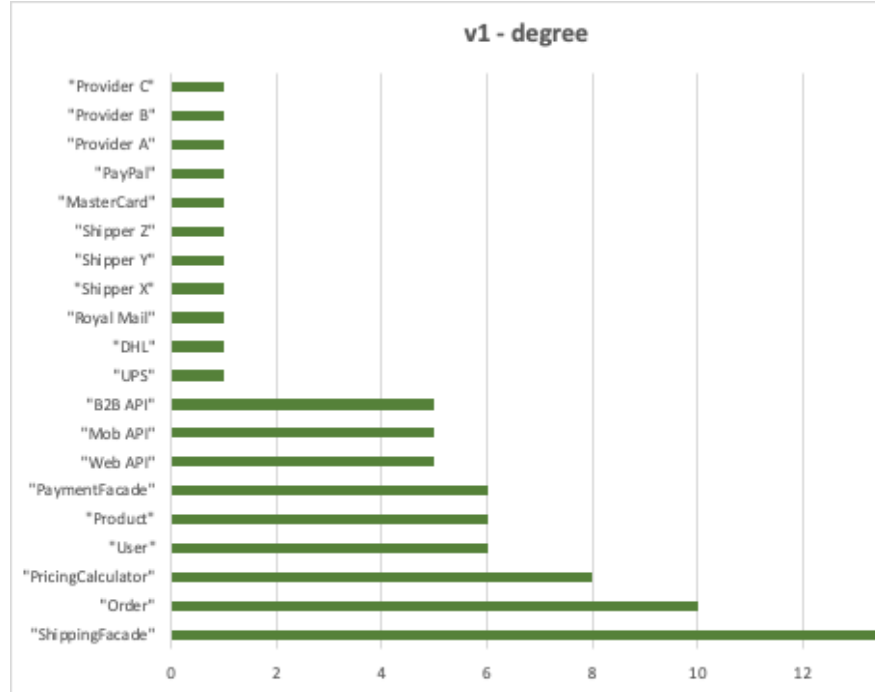
name	degree	degree_in	degree_out	triangles	cluster_coeff
ShippingFacade	14.0	5.0	9.0	17	0.21794871794871795
Order	10.0	5.0	5.0	17	0.6071428571428571
PricingCalculator	8.0	4.0	4.0	17	0.8095238095238095
User	6.0	6.0	0.0	12	0.8
Product	6.0	6.0	0.0	12	0.8
PaymentFacade	6.0	1.0	5.0	0	0.0
Web API	5.0	0.0	5.0	9	0.9
Mob API	5.0	0.0	5.0	9	0.9
B2B API	5.0	0.0	5.0	9	0.9
UPS	1.0	1.0	0.0	0	0.0
DHL	1.0	1.0	0.0	0	0.0
Royal Mail	1.0	1.0	0.0	0	0.0
Shipper X	1.0	1.0	0.0	0	0.0
Shipper Y	1.0	1.0	0.0	0	0.0
Shipper Z	1.0	1.0	0.0	0	0.0
MasterCard	1.0	1.0	0.0	0	0.0
PayPal	1.0	1.0	0.0	0	0.0
Provider A	1.0	1.0	0.0	0	0.0
Provider B	1.0	1.0	0.0	0	0.0
Provider C	1.0	1.0	0.0	0	0.0



Density: 0.19
Number of nodes: 20
Number of edges: 36
Average degree: 3.6000
Average Clustering Co-eff: 0.30

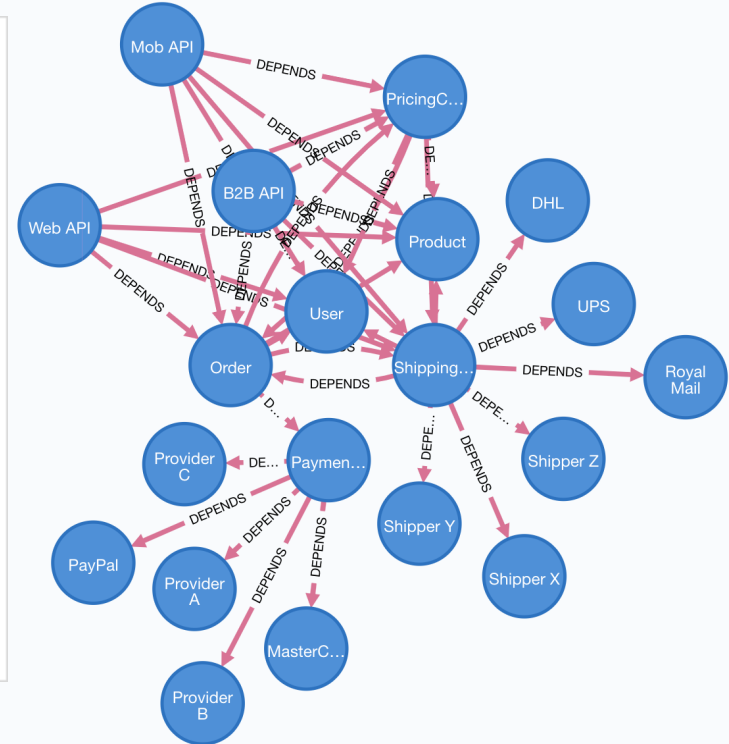


V1.0 - DEGREE





V1.0 - CLUSTER COEFFICIENT



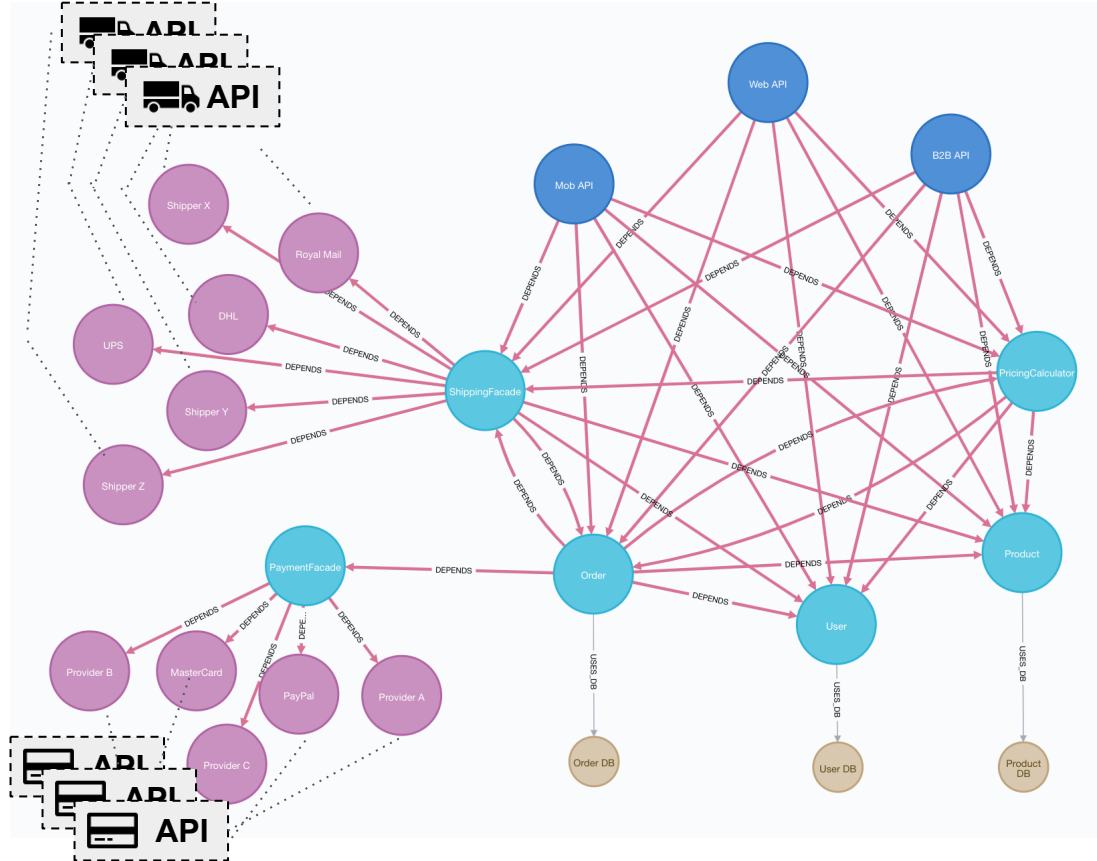


We've been directed to look a bit closer at a few potential problem services ...

Where do they fit in architecturally?



V1.0 - ARCHITECTURE DIAGRAM



Front End Service

Back End Service

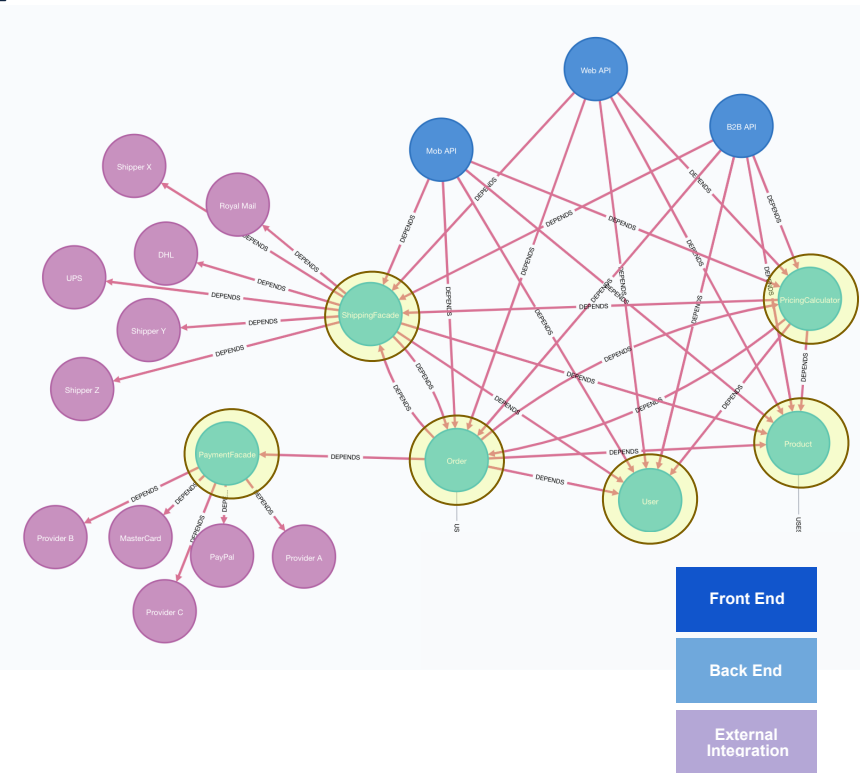
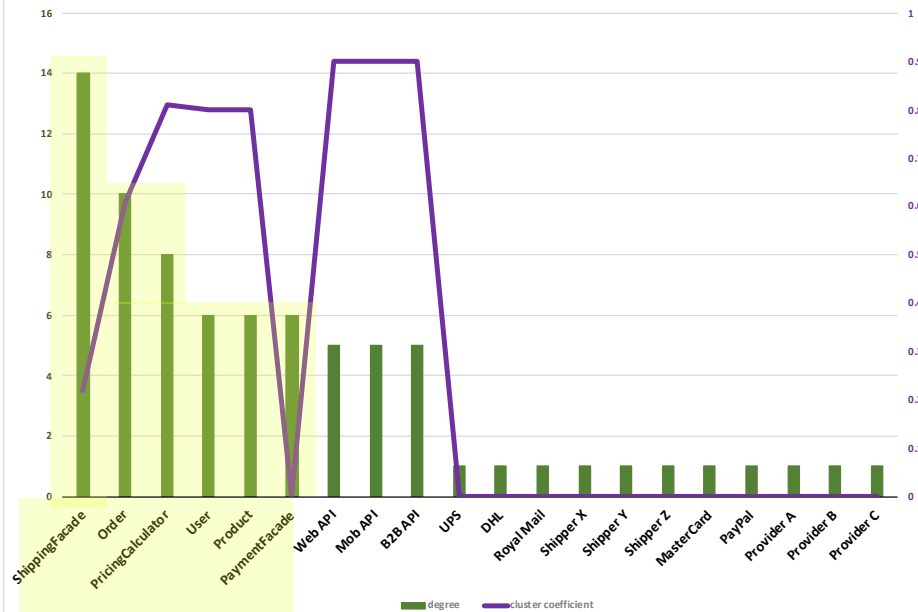
External Integration Adapter Service

Database



V1.0 ANALYSIS (DEGREE)

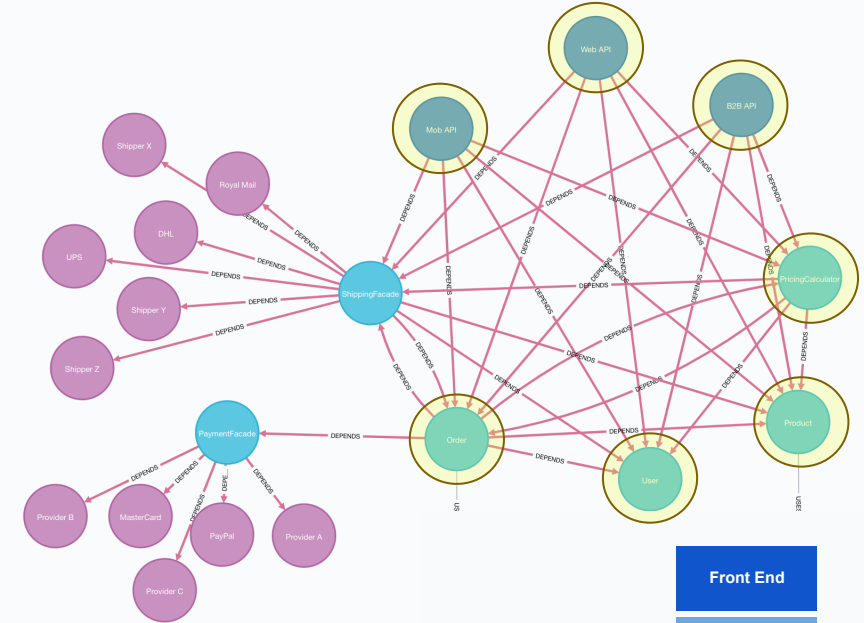
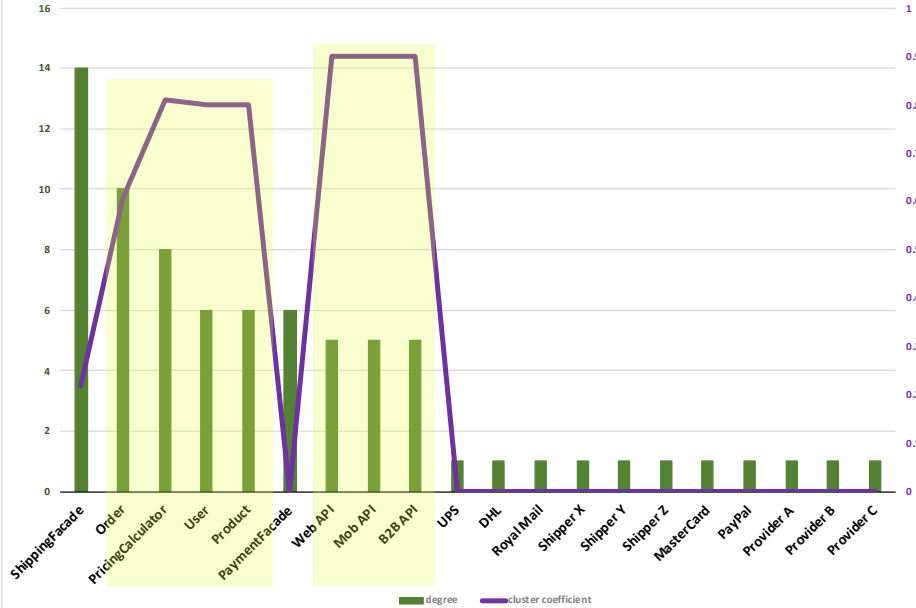
v1 - degree & cluster coefficient





V1.0 ANALYSIS (CLUSTER COEFF)

v1 - degree & cluster coefficient



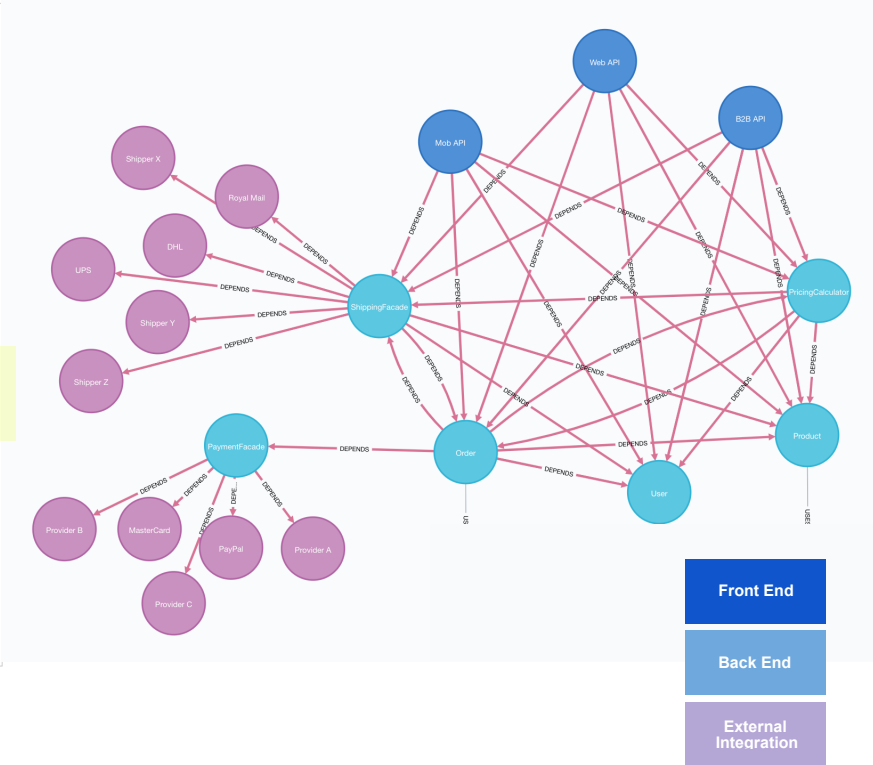
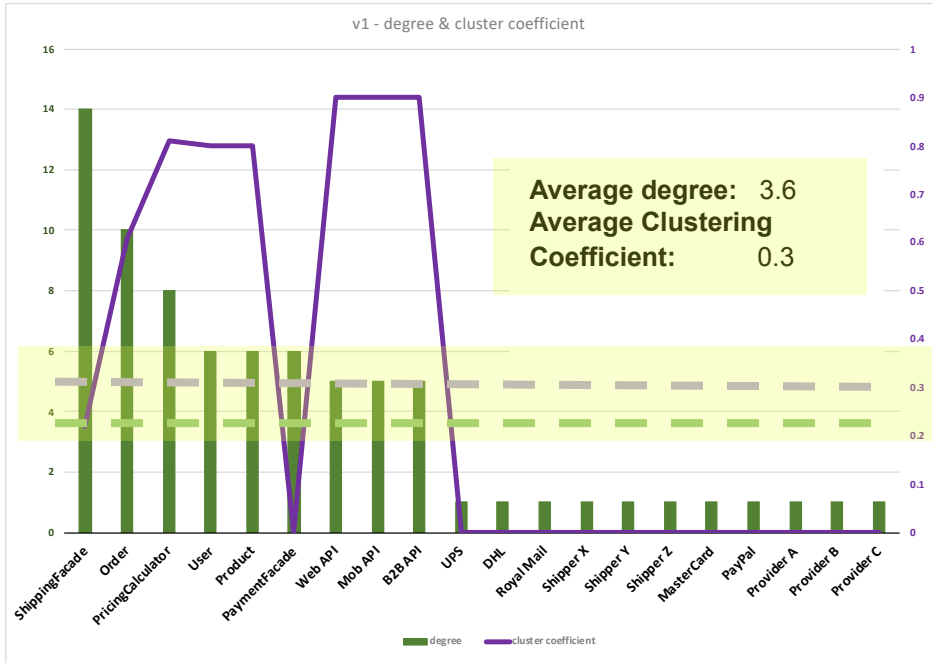
Front End

Back End

External Integration



V1.0 ANALYSIS (AVERAGES)

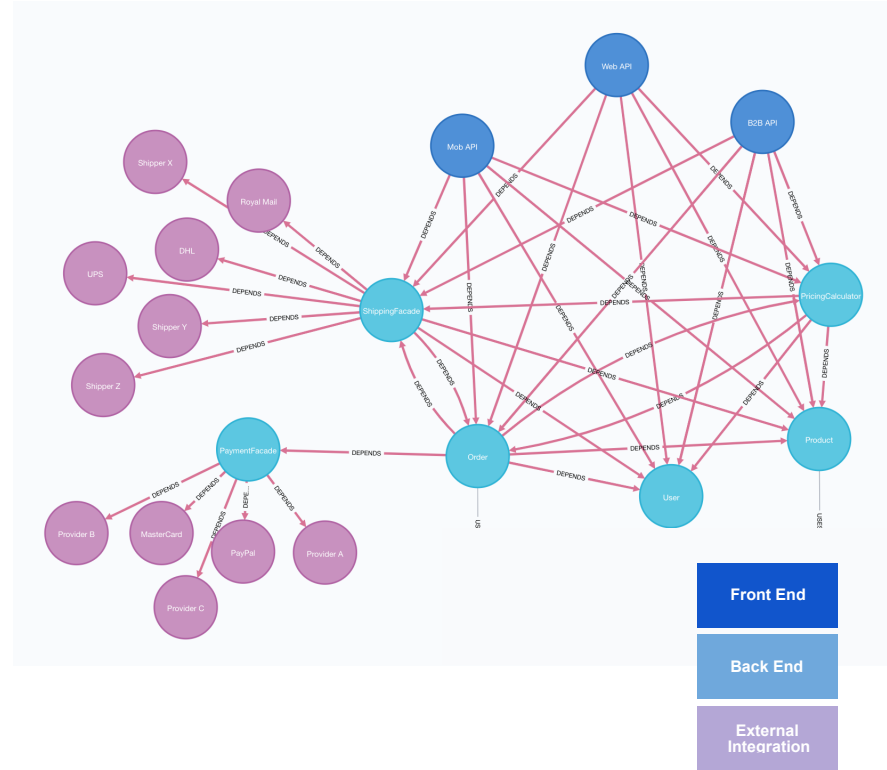




V1.0 INVESTIGATION



- Entity Services Anti-Pattern
- Distributed Monolith

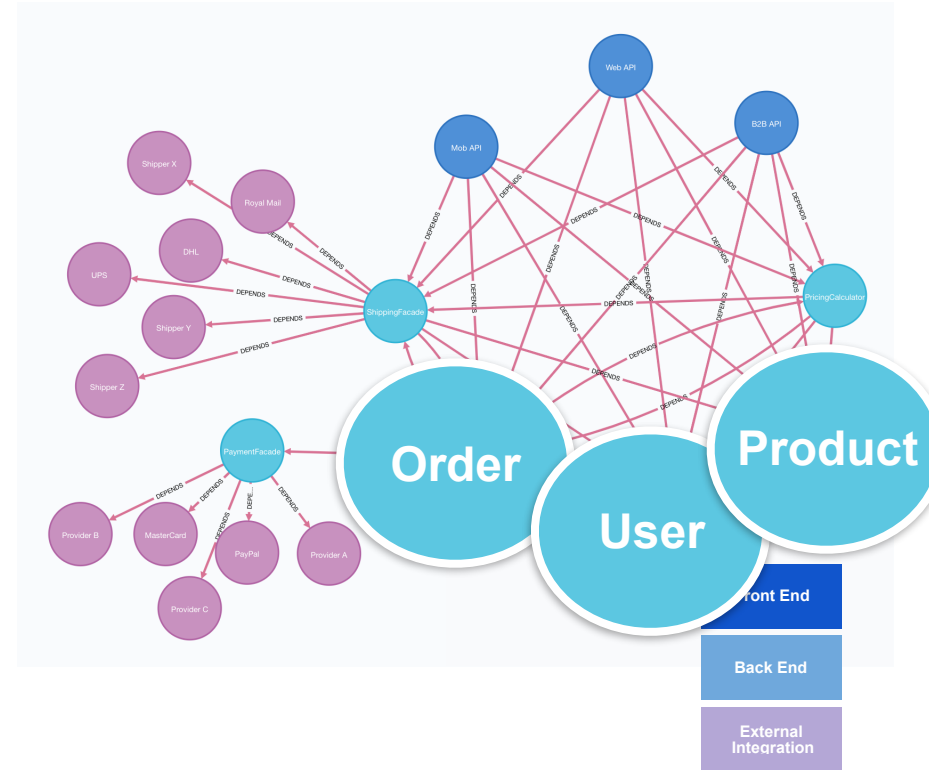




V1.0 INVESTIGATION



- Entity Services Anti-Pattern
- Distributed Monolith

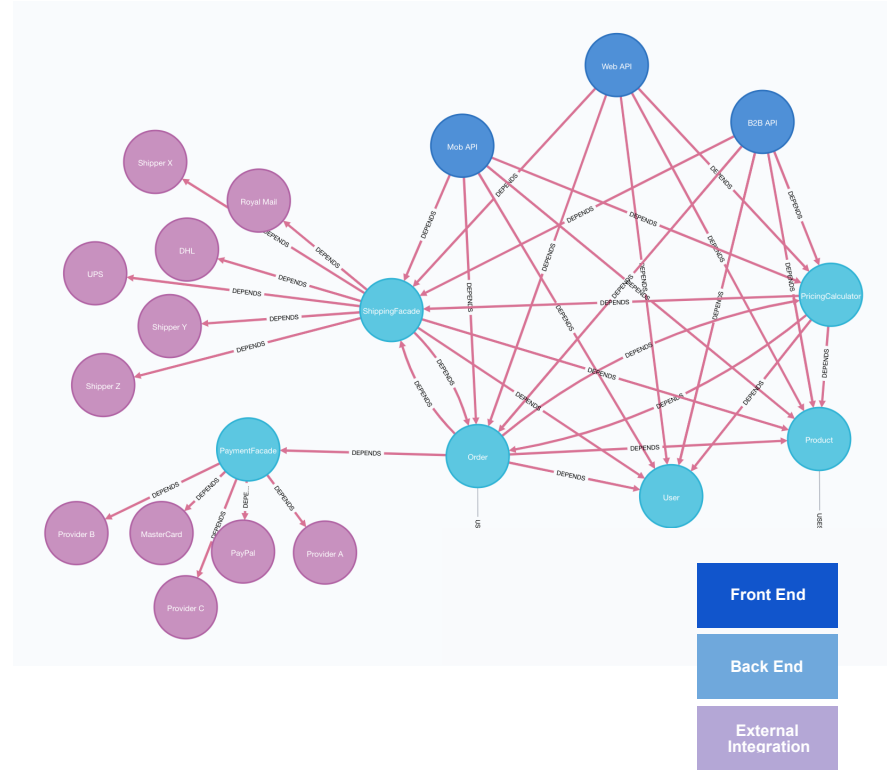




V1.0 INVESTIGATION

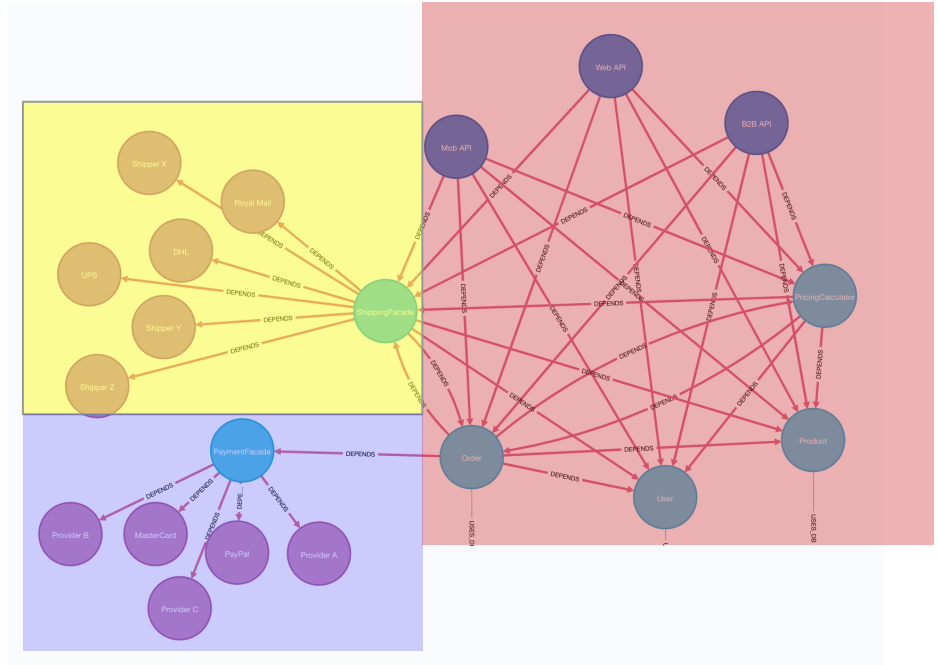
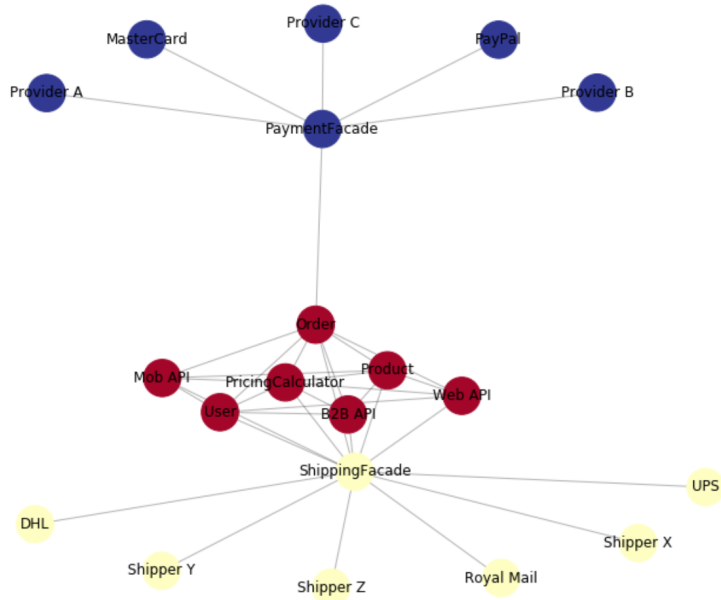


- Entity Services Anti-Pattern
- Distributed Monolith





V1.0 - COMMUNITY DETECTION

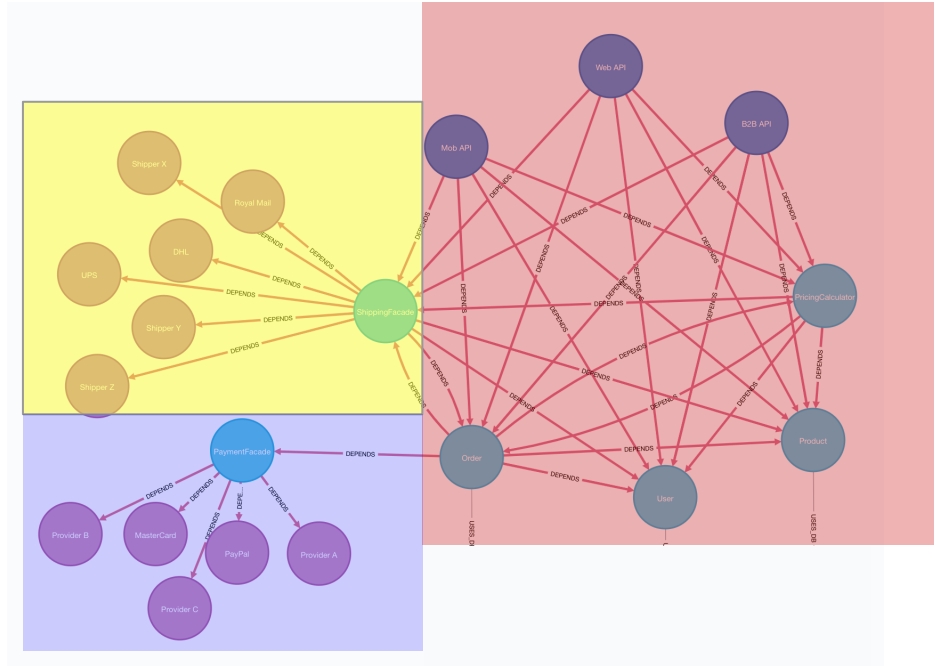




V1.0 - OBSERVATIONS CONTINUED

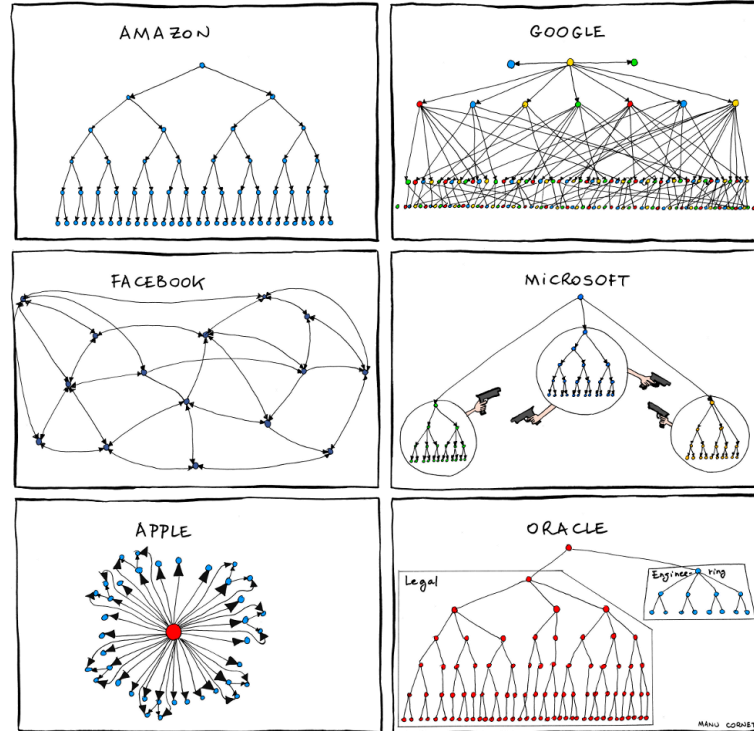


- Help detect tightly coupled areas
- Not always DDD aligned





COMMUNITIES NOT ALWAYS DDD ALIGNED



Conway's Law ?



COMMUNITIES NOT ALWAYS DDD ALIGNED



***Process &
Data
Criteria***

Organisation

Operations

Erich Eichinger Blog: "Heuristics for Identifying Service Boundaries"
<https://opencredo.com/blogs/identify-service-boundary-heuristics/>



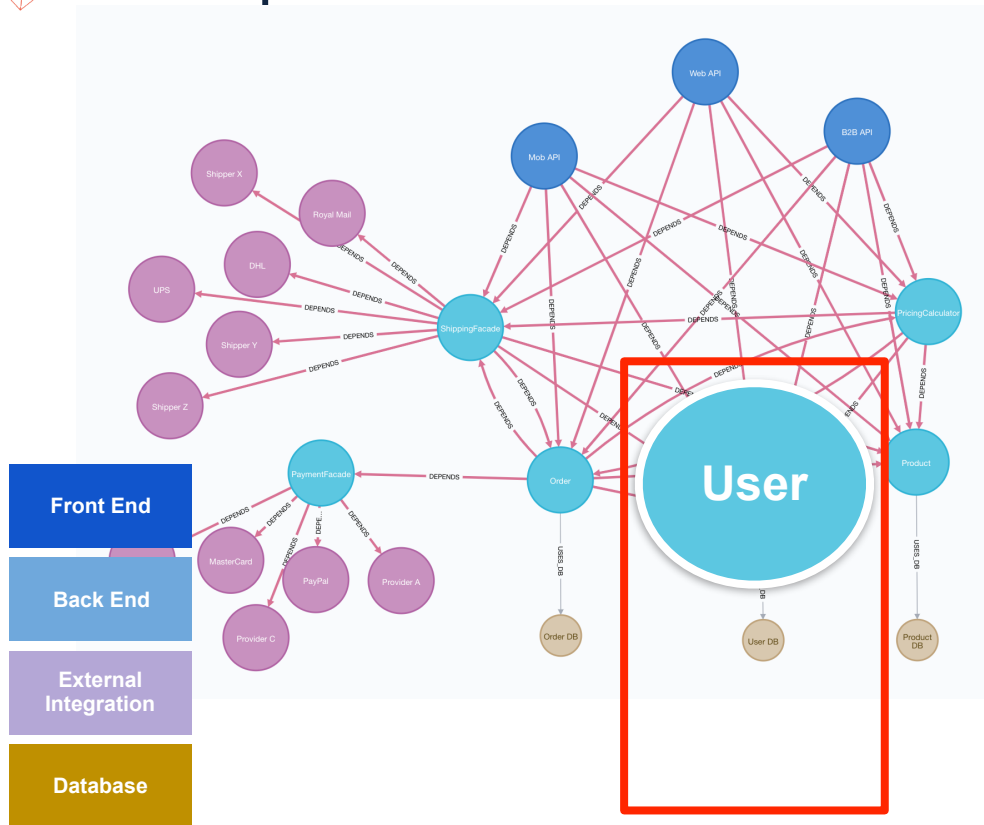
#2

The Revised Microservice Arch

User Domain Decoupling (v2.0)

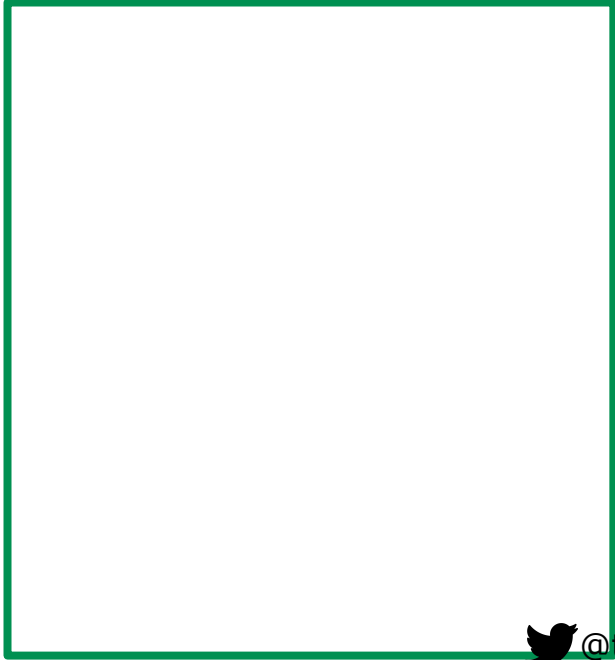
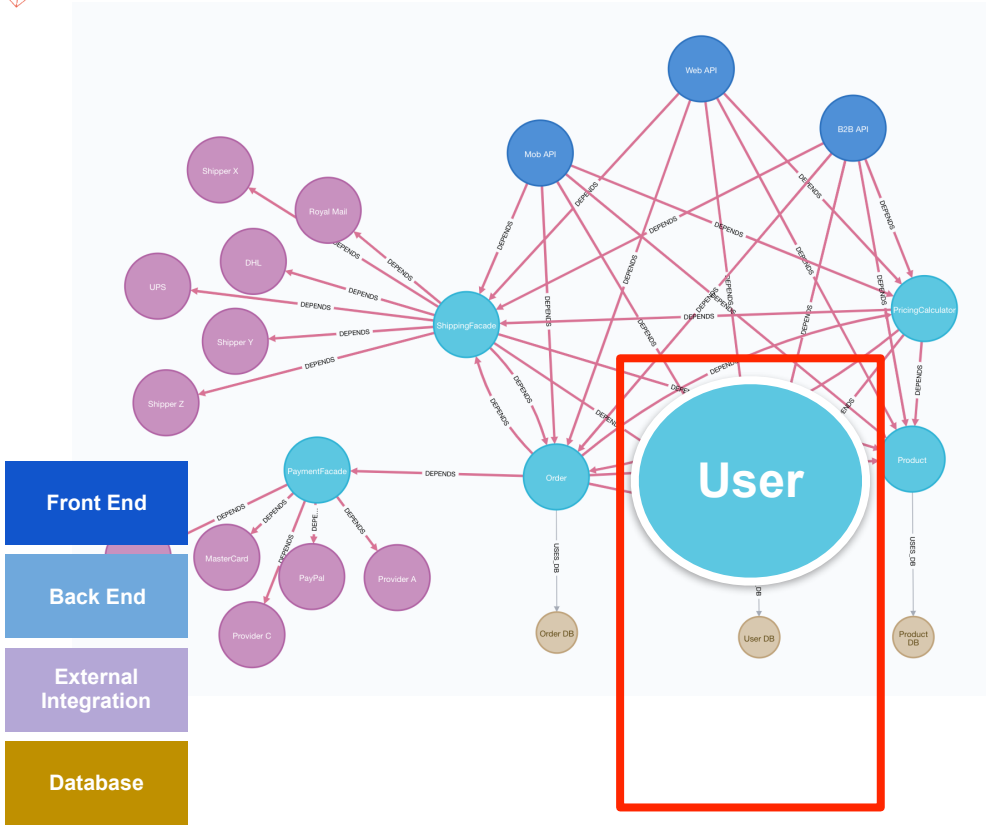


Recap ... V1.0



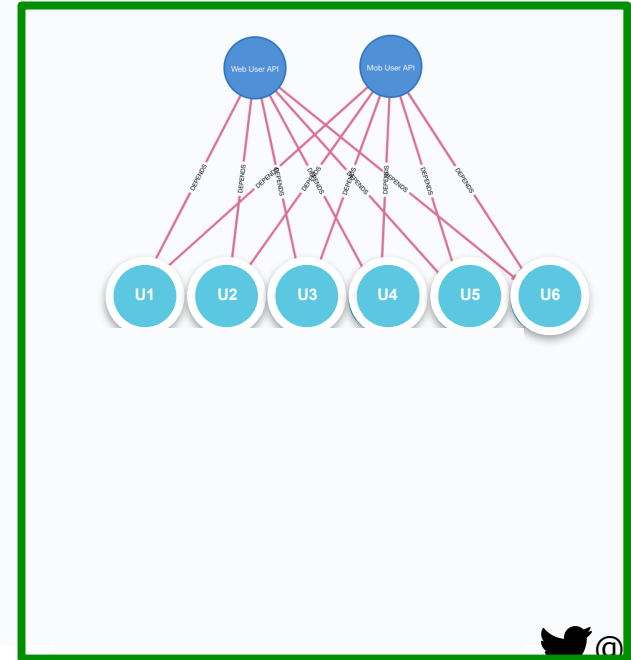
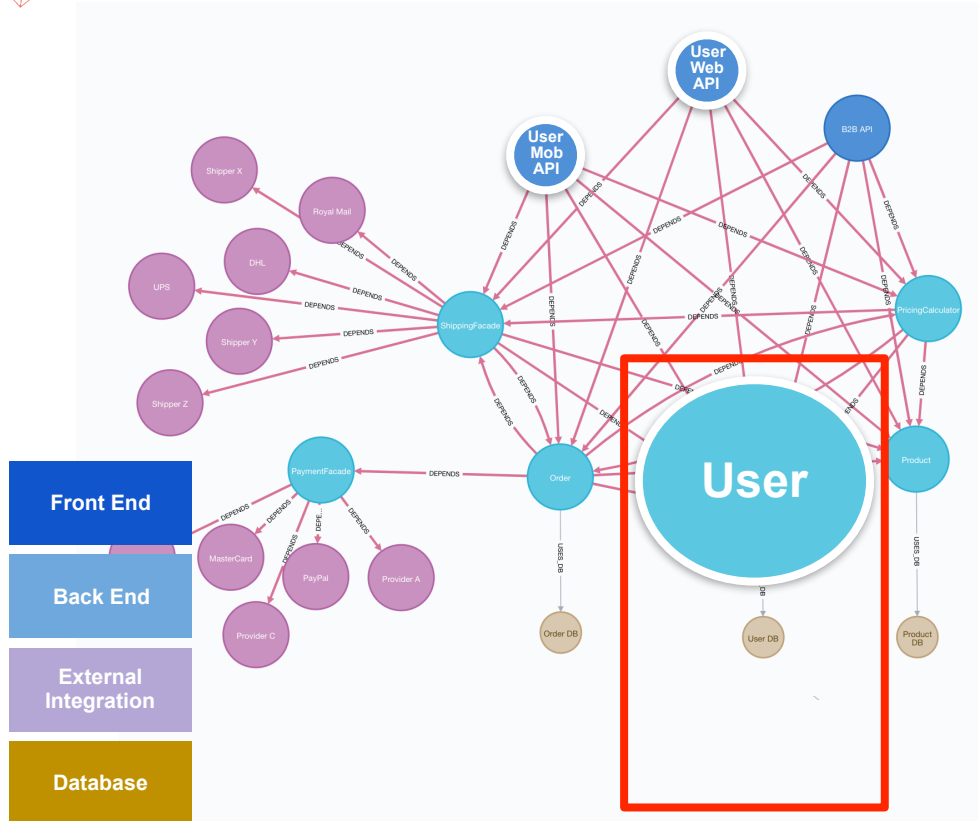


V1.0 —> V2.0



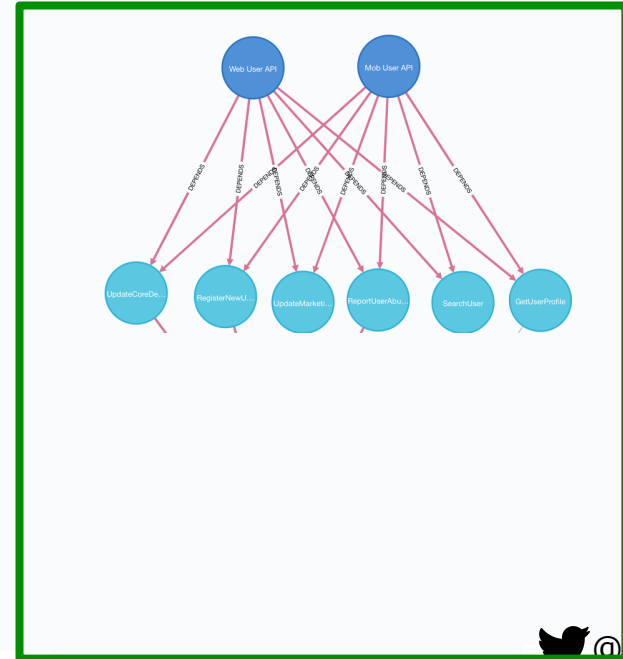
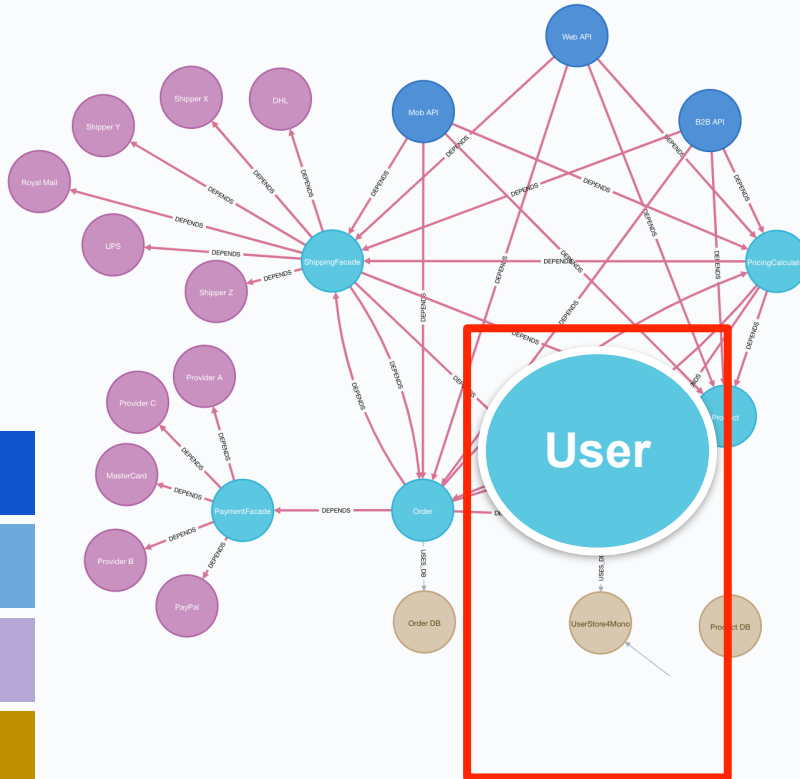


V1.0 —> V2.0



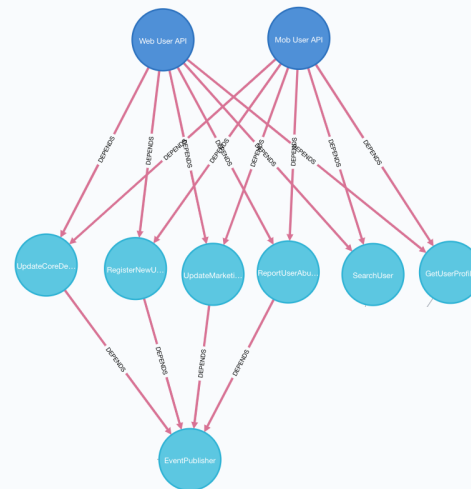
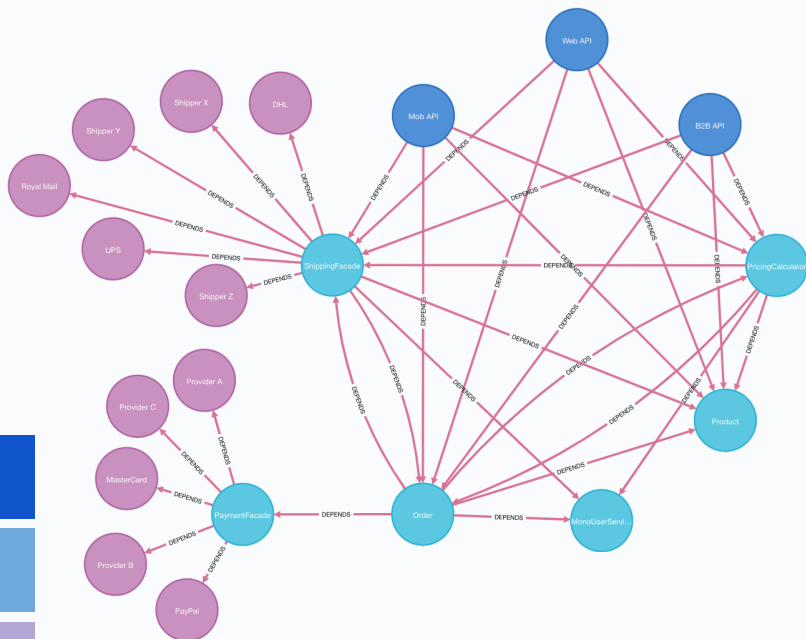


V1.0 → V2.0





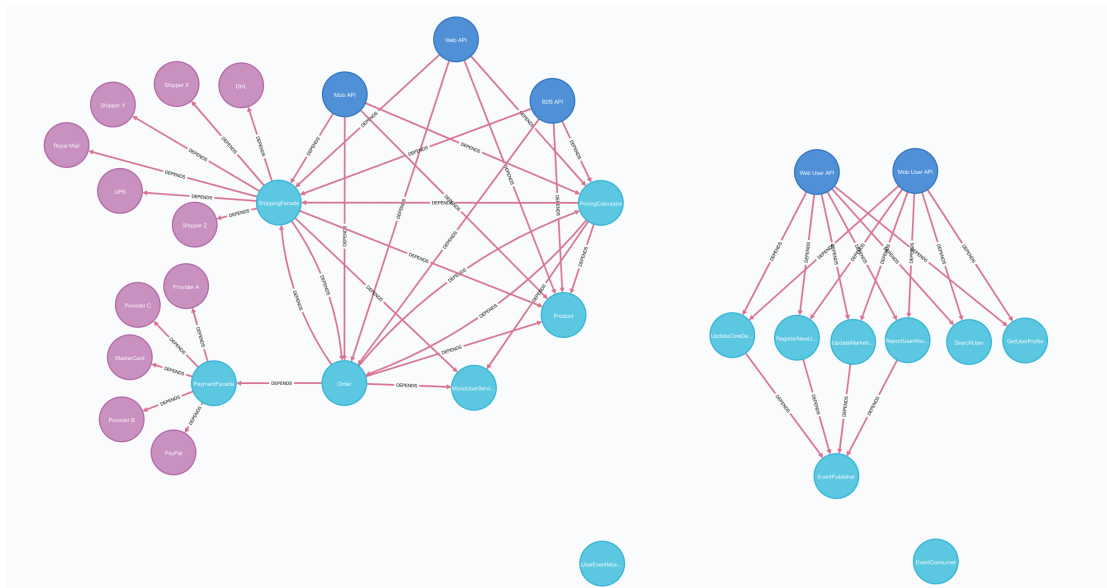
V2.0





V2.0 - NETWORK STATS

name	degree	degree_in	degree_out	triangles	cluster_coeff
ShippingFacade	14.0	5.0	9.0	14	0.1794871794871795
Order	10.0	5.0	5.0	14	0.5
PricingCalculator	8.0	4.0	4.0	14	0.6666666666666666
Web User API	6.0	0.0	6.0	0	0.0
Mob User API	6.0	0.0	6.0	0	0.0
Product	6.0	6.0	0.0	12	0.8
PaymentFacade	6.0	1.0	5.0	0	0.0
Web API	4.0	0.0	4.0	6	1.0
Mob API	4.0	0.0	4.0	6	1.0
B2B API	4.0	0.0	4.0	6	1.0
EventPublisher	4.0	4.0	0.0	0	0.0
RegisterNewUser	3.0	2.0	1.0	0	0.0
UpdateCoreDetails	3.0	2.0	1.0	0	0.0
UpdateMarketingPrefs	3.0	2.0	1.0	0	0.0
ReportUserAbuse	3.0	2.0	1.0	0	0.0
MonoUserService	3.0	3.0	0.0	3	1.0
GetUserProfile	2.0	2.0	0.0	0	0.0
SearchUser	2.0	2.0	0.0	0	0.0
UPS	1.0	1.0	0.0	0	0.0
DHL	1.0	1.0	0.0	0	0.0
Royal Mail	1.0	1.0	0.0	0	0.0
Shipper X	1.0	1.0	0.0	0	0.0
Shipper Y	1.0	1.0	0.0	0	0.0
Shipper Z	1.0	1.0	0.0	0	0.0
MasterCard	1.0	1.0	0.0	0	0.0
PayPal	1.0	1.0	0.0	0	0.0
Provider A	1.0	1.0	0.0	0	0.0
Provider B	1.0	1.0	0.0	0	0.0
Provider C	1.0	1.0	0.0	0	0.0
EventConsumer	0.0	0.0	0.0	0	0.0
UserEventMonoConsumer	0.0	0.0	0.0	0	0.0

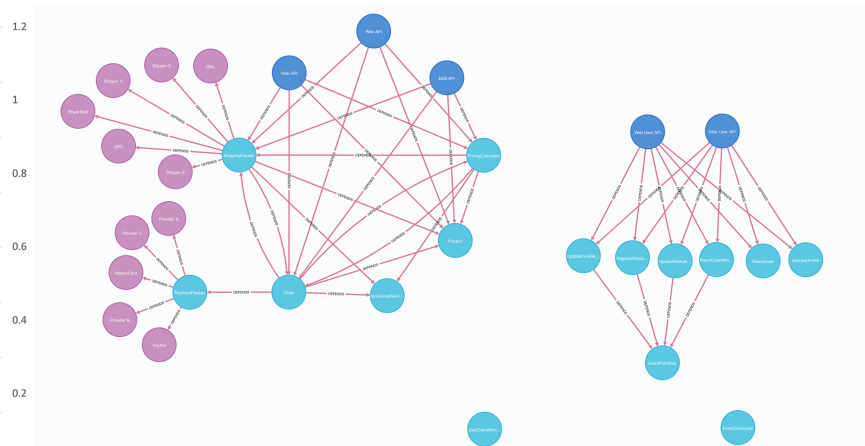


Density: 0.11
Number of nodes: 29
Number of edges: 46
Average degree: 3.1724
Average Clustering Co-eff: 0.20



Plot Area

Class	degree	cluster coefficient
Web API	4	1.0
Mob API	4	1.0
Edx API	4	1.0
MonoUserService	3	0.8
Product	6	0.7
PricingCalculator	8	0.6
Order	10	0.5
ShippingCode	14	0.3
Web User API	6	0.1
Mob User API	6	0.0
PaymentScale	6	0.0
EventPublisher	6	0.0
RegisterNewUser	4	0.0
UpdateCoreDetails	3	0.0
MarketingPrefs	3	0.0
ReportUserBase	3	0.0
GetUserProfile	3	0.0
SearchUser	2	0.0
UPS	2	0.0
DHL	1	0.0
Royal Mail	1	0.0
Shipper X	1	0.0
Shipper Y	1	0.0
Shipper Z	1	0.0
MasterCard	1	0.0
Paypal	1	0.0
Provider A	1	0.0
Provider B	1	0.0
Provider C	1	0.0
EventConsumer	1	0.0
UserEventMonoConsumer	0	0.0

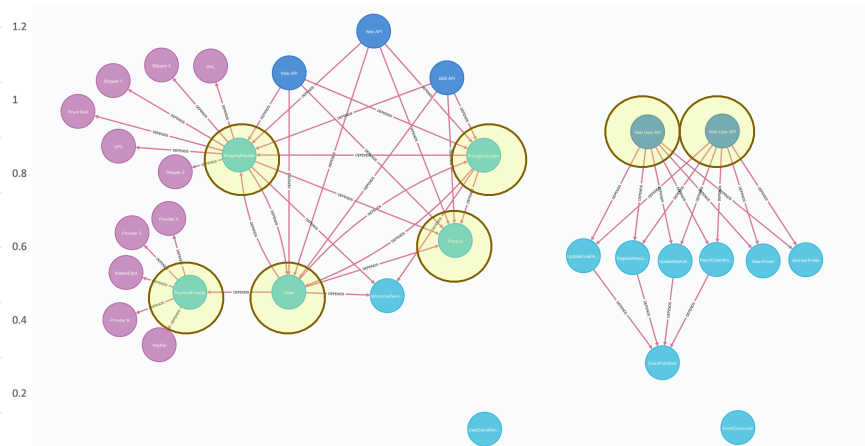
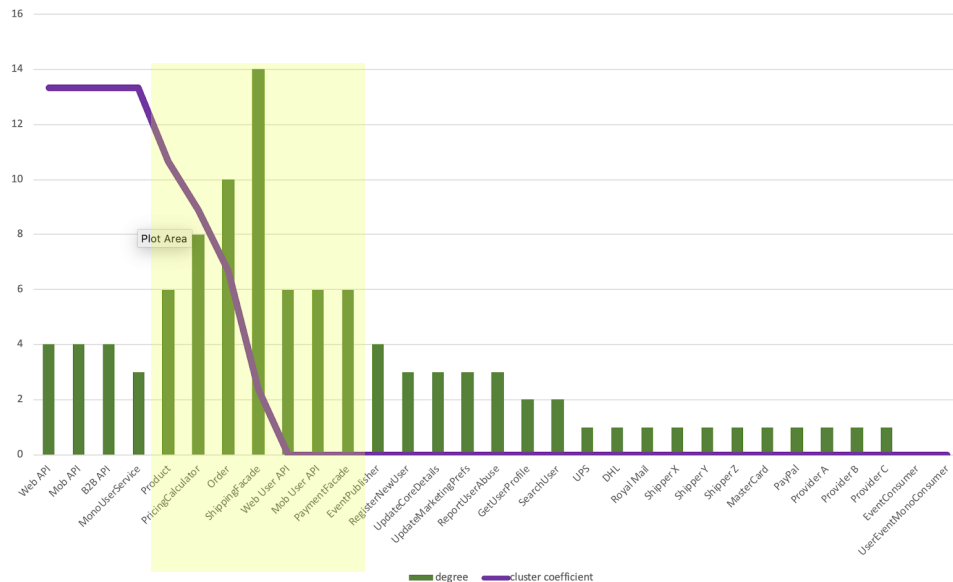




V2.0 - COMBINED STATS (DEGREE)



V2 - Degree & Cluster Coefficient

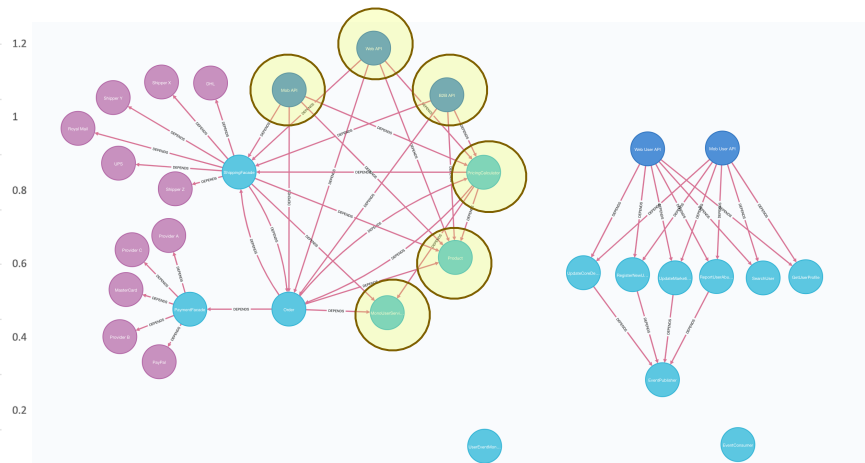
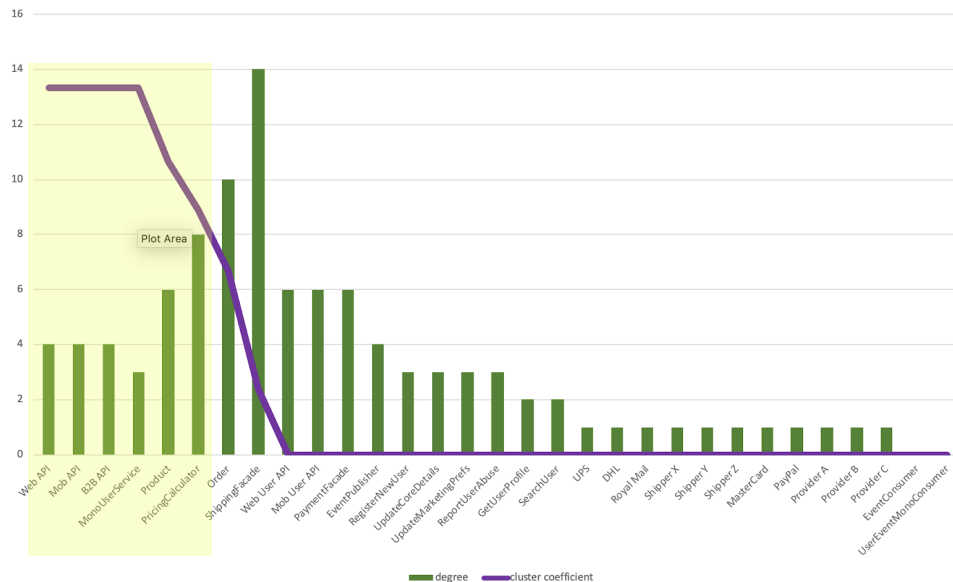




V2.0 - COMBINED STATS (CLUSTER COEFF)

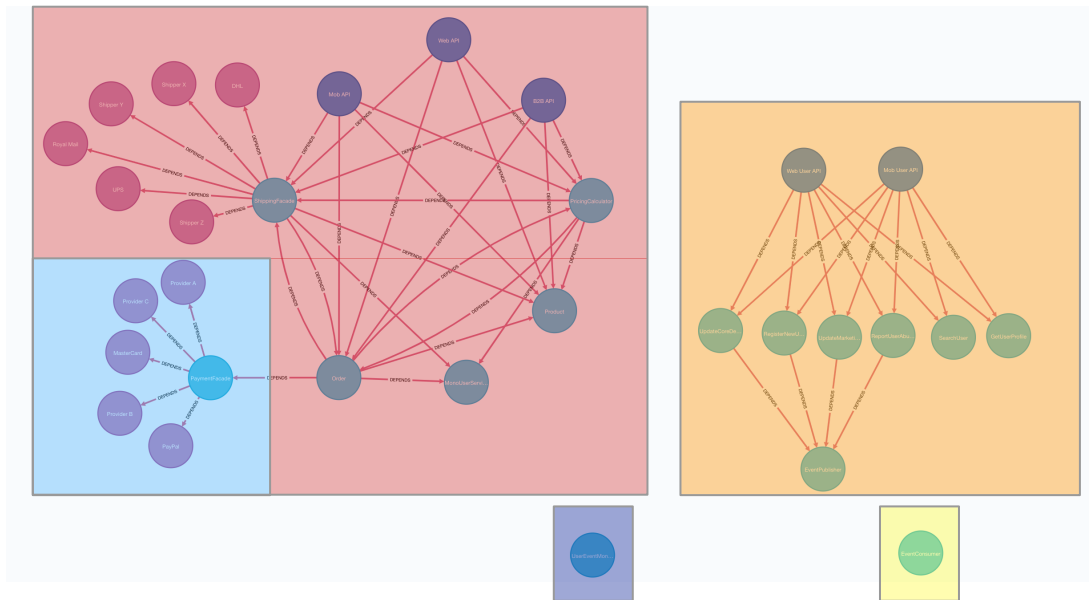
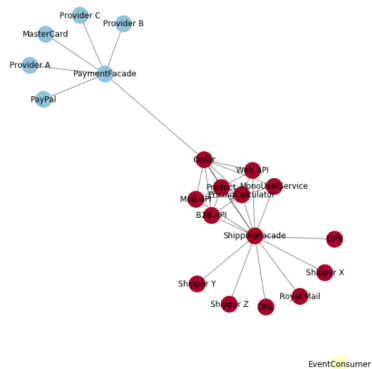


V2 - Degree & Cluster Coefficient





V2.0 - COMMUNITY DETECTION

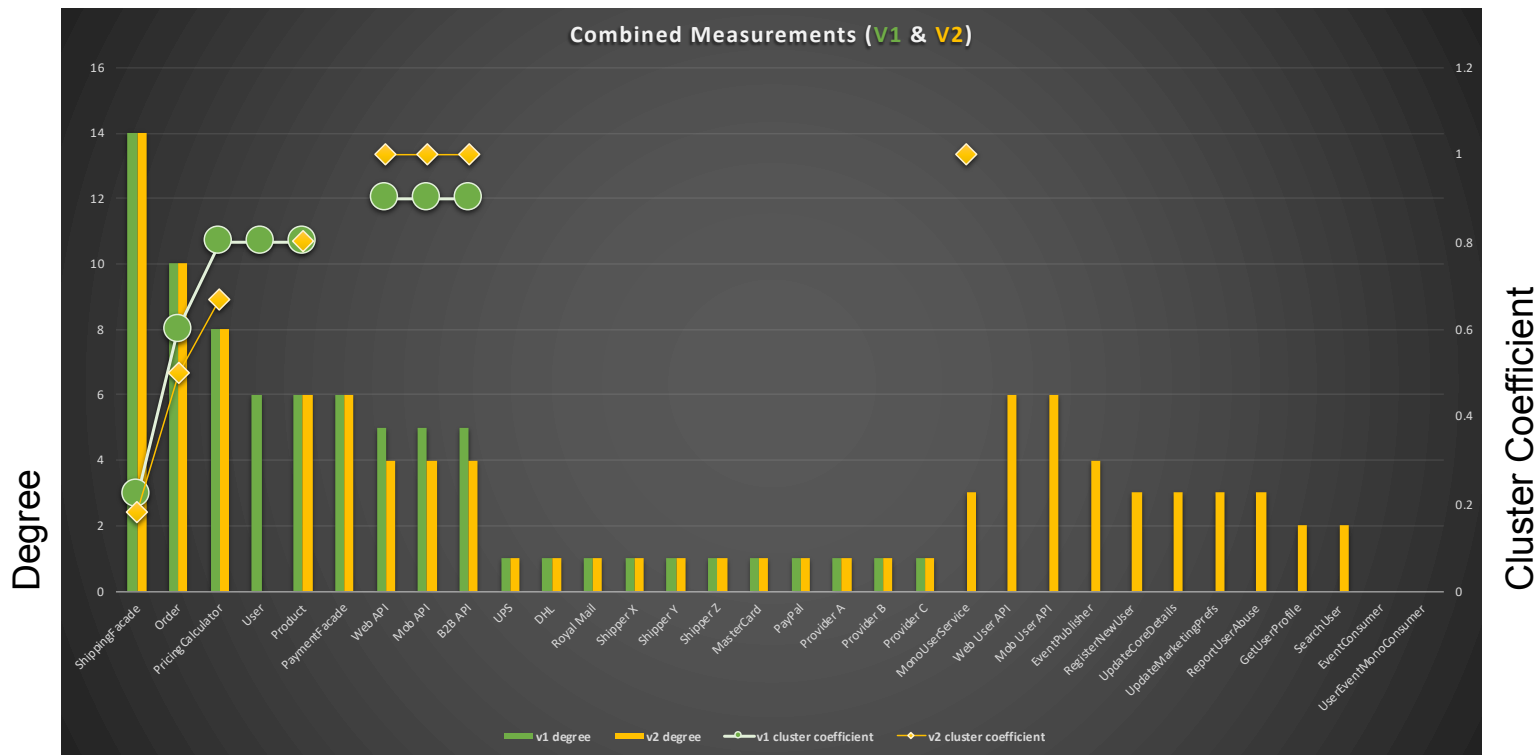




COMPARING V1 and V2

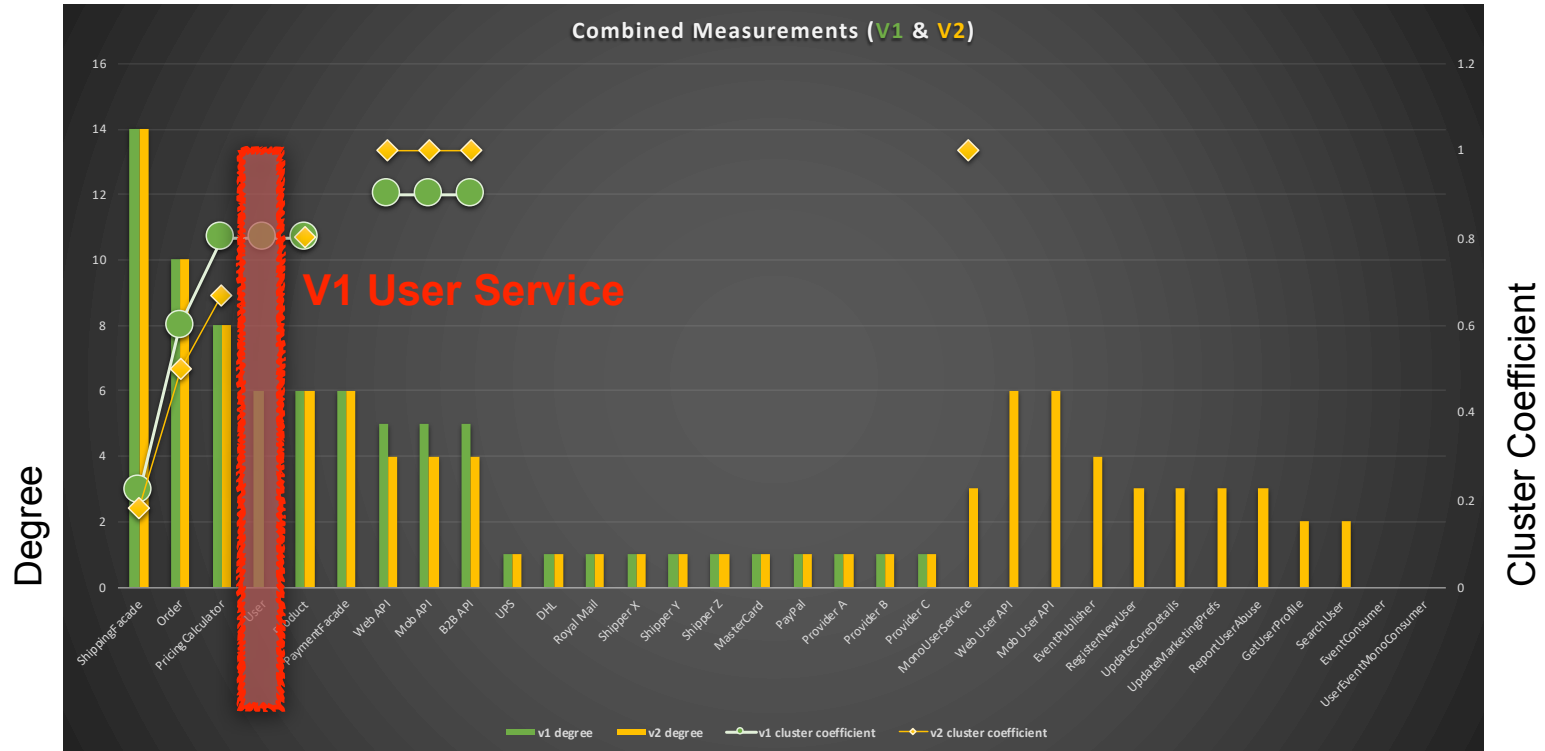


COMPARING



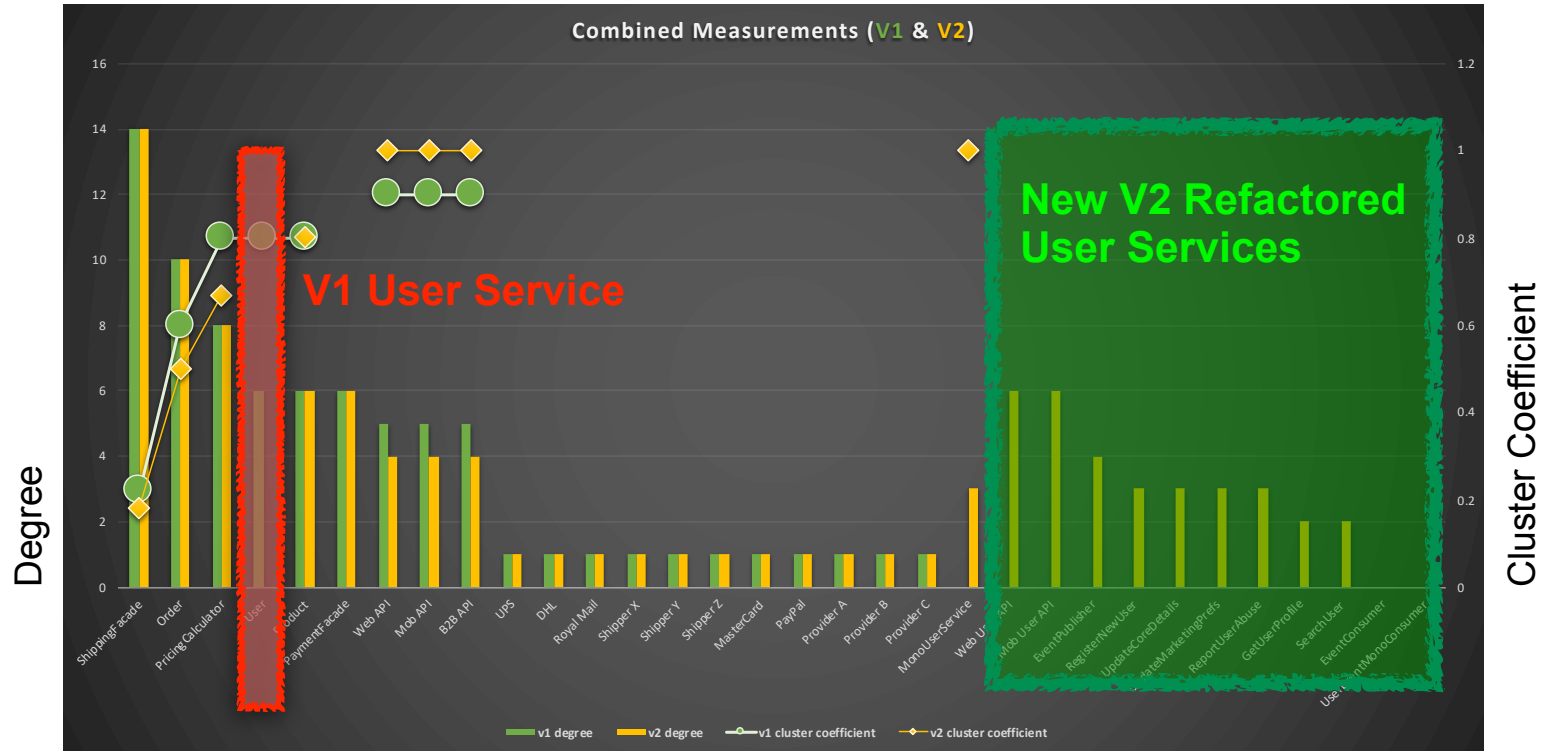


COMPARING



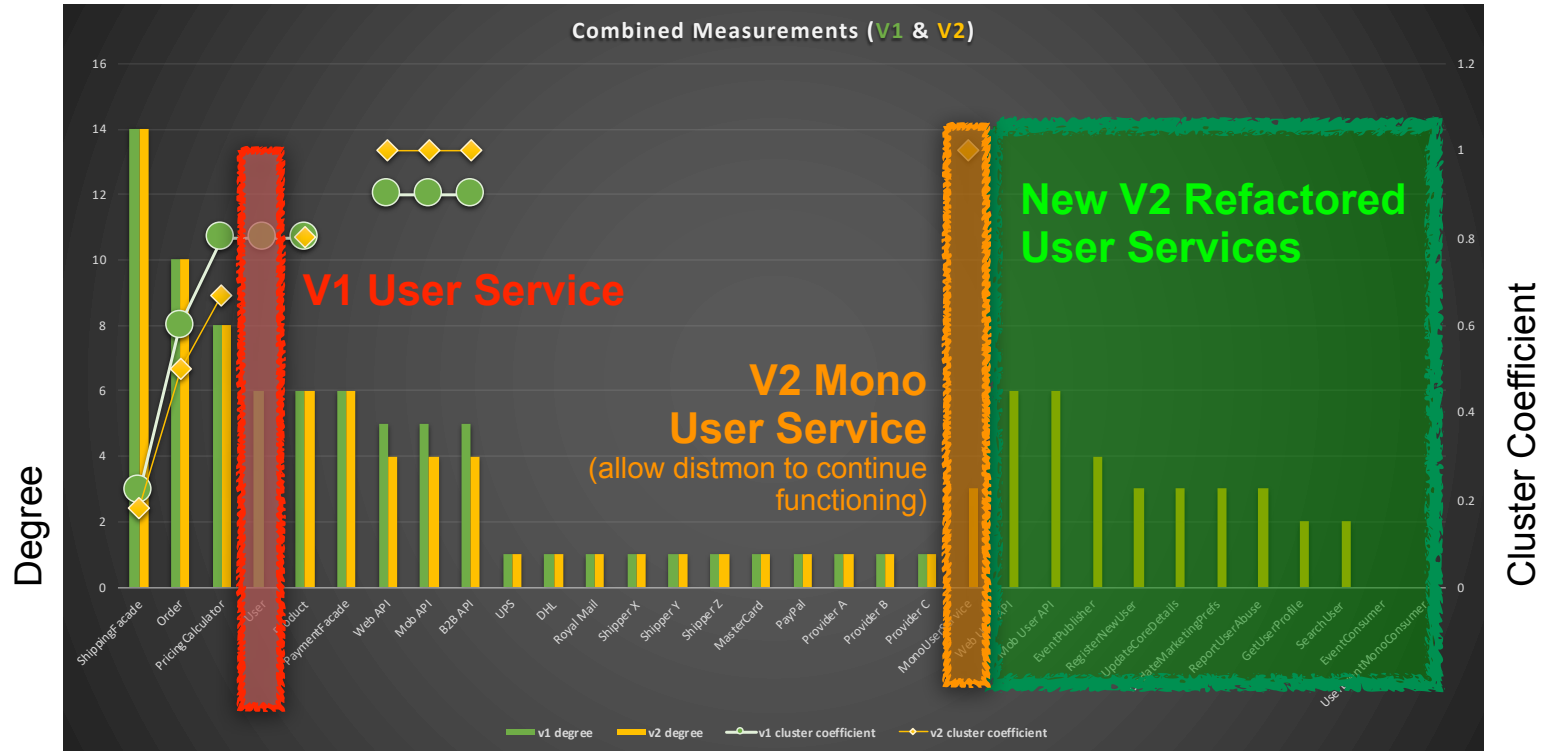


COMPARING



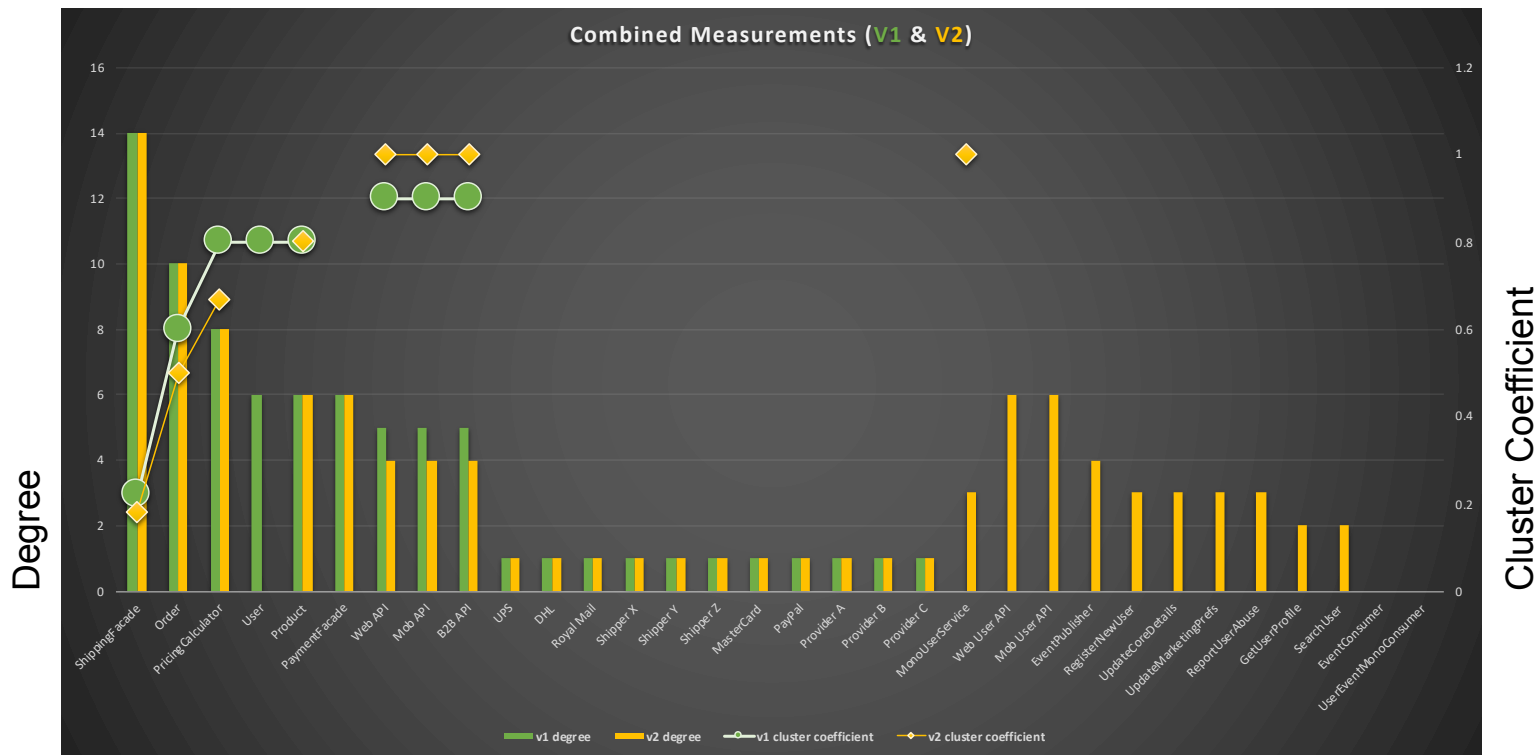


COMPARING



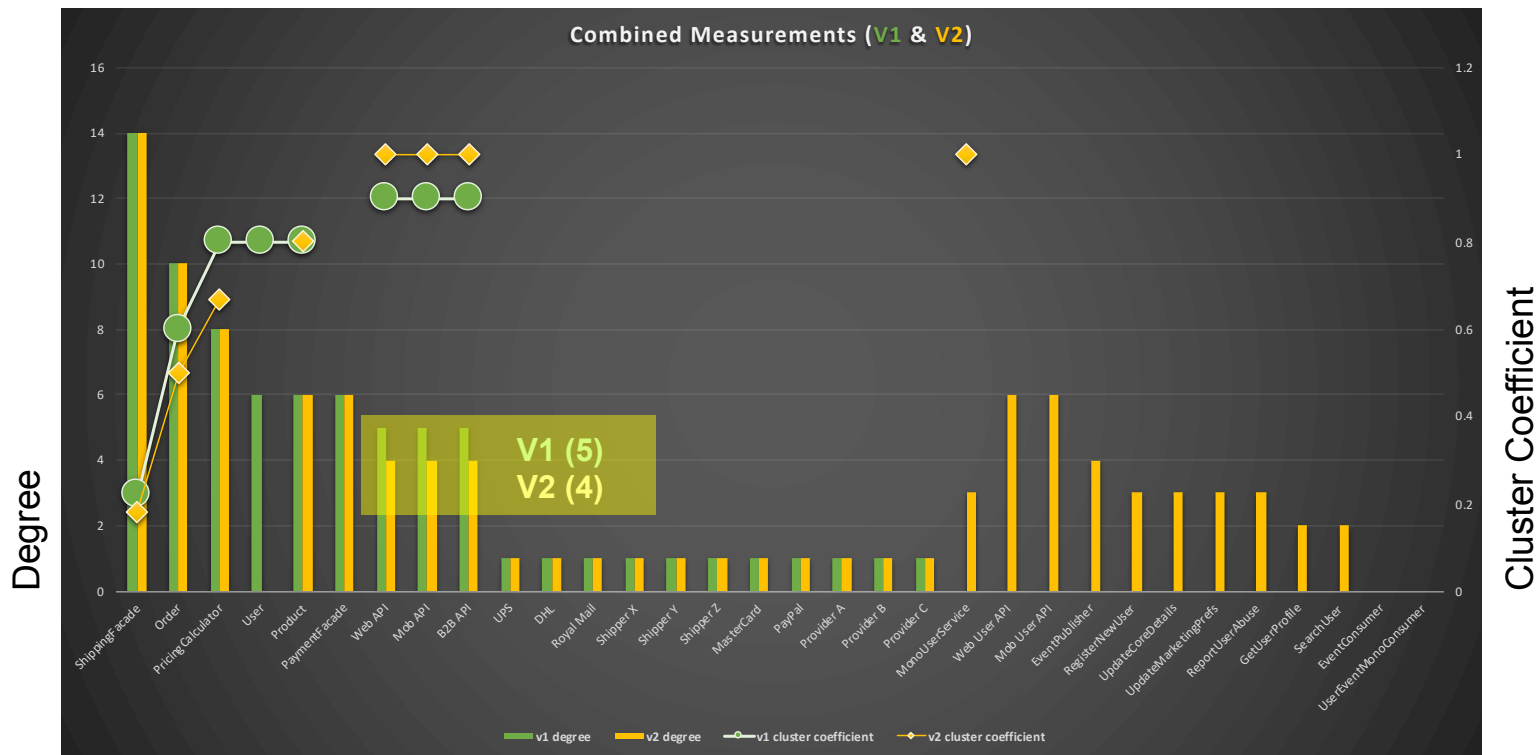


COMPARING



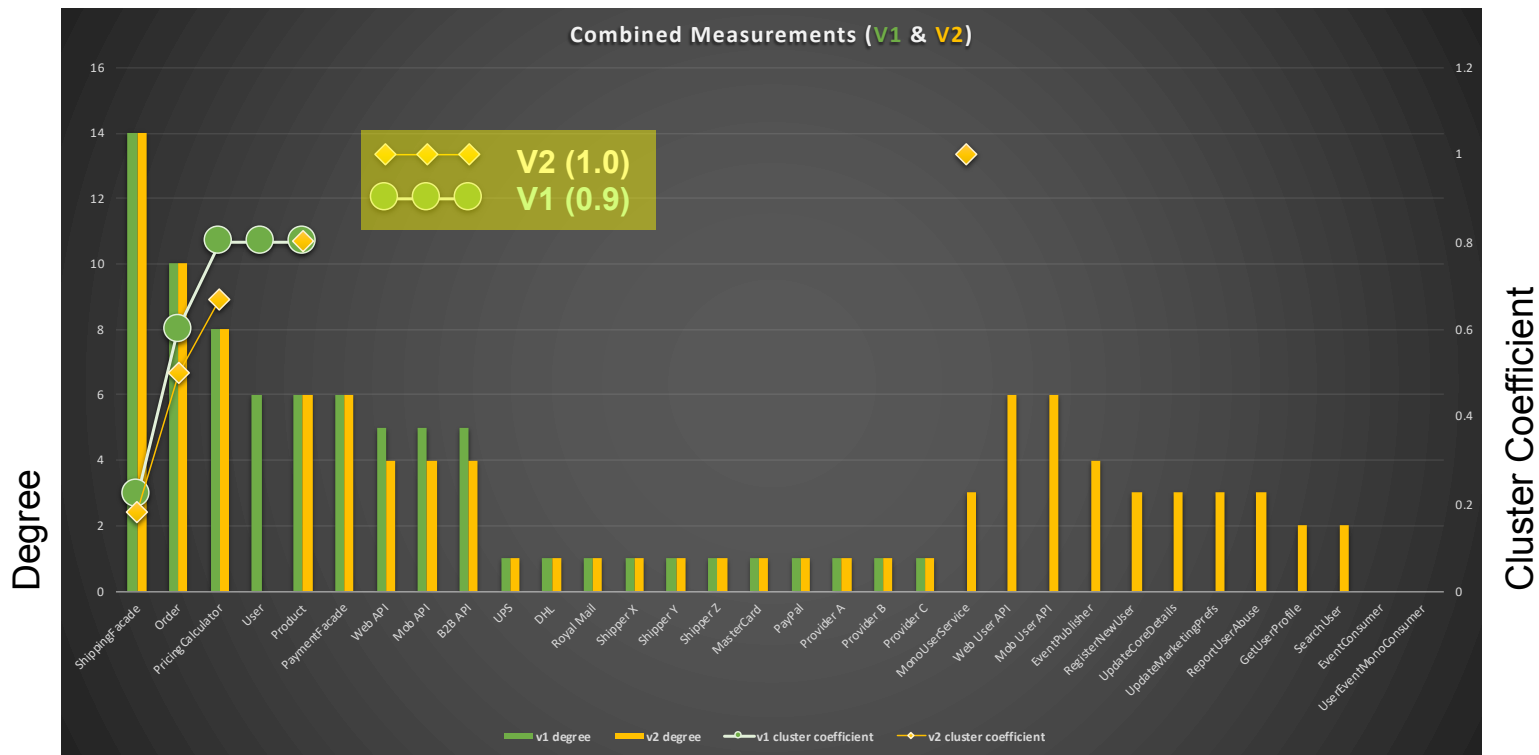


COMPARING





COMPARING





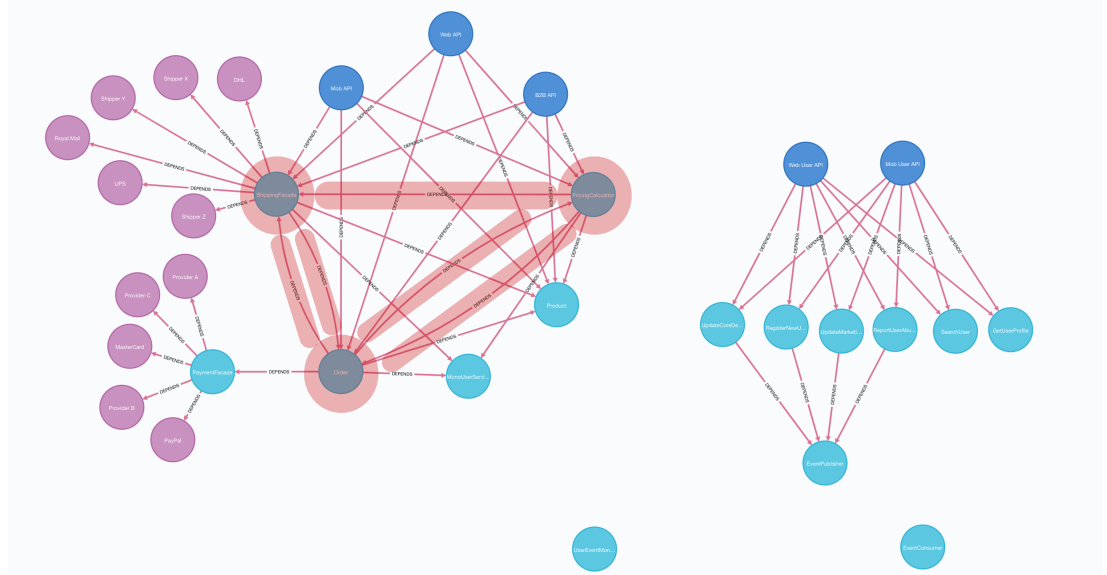
FURTHER OPTIONS



OTHER MEASURES & ALGORITHMS



- **Strongly Connected Components (SCC)**
- Help detect circular dependencies





OTHER MEASURES & ALGORITHMS



- **Path Finding Algorithms**

Detect who is calling deprecated services



CONCLUSION



CONCLUSION



Early days ...
Can't be applied blindly ...

Common sense and
understanding of architectural
approaches and patterns still
required!!



HYPOTHESIS



General Hypothesis: Data Driven Architectural Improvement

You can extract metrics and KPIs from a
microservices architecture using graph theory

AND

use these to gain insight into the structure
and characteristics of your microservices
architecture



HYPOTHESIS



General Hypothesis: Data Driven Architectural Improvement



You can extract metrics and KPIs from a microservices architecture using graph theory

AND



use these to gain insight into the structure and characteristics of your microservices architecture



Specifically



Can we use these metrics to detect bad microservice architectural smells and anti-patterns like a tightly coupled architecture (distributed monolith)?



CONCLUSION

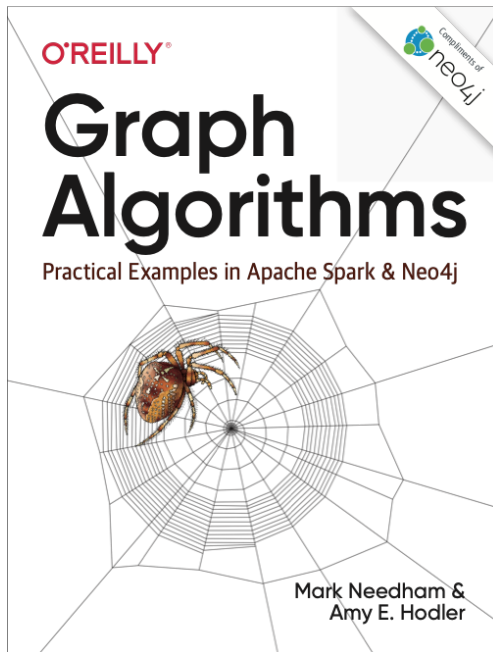


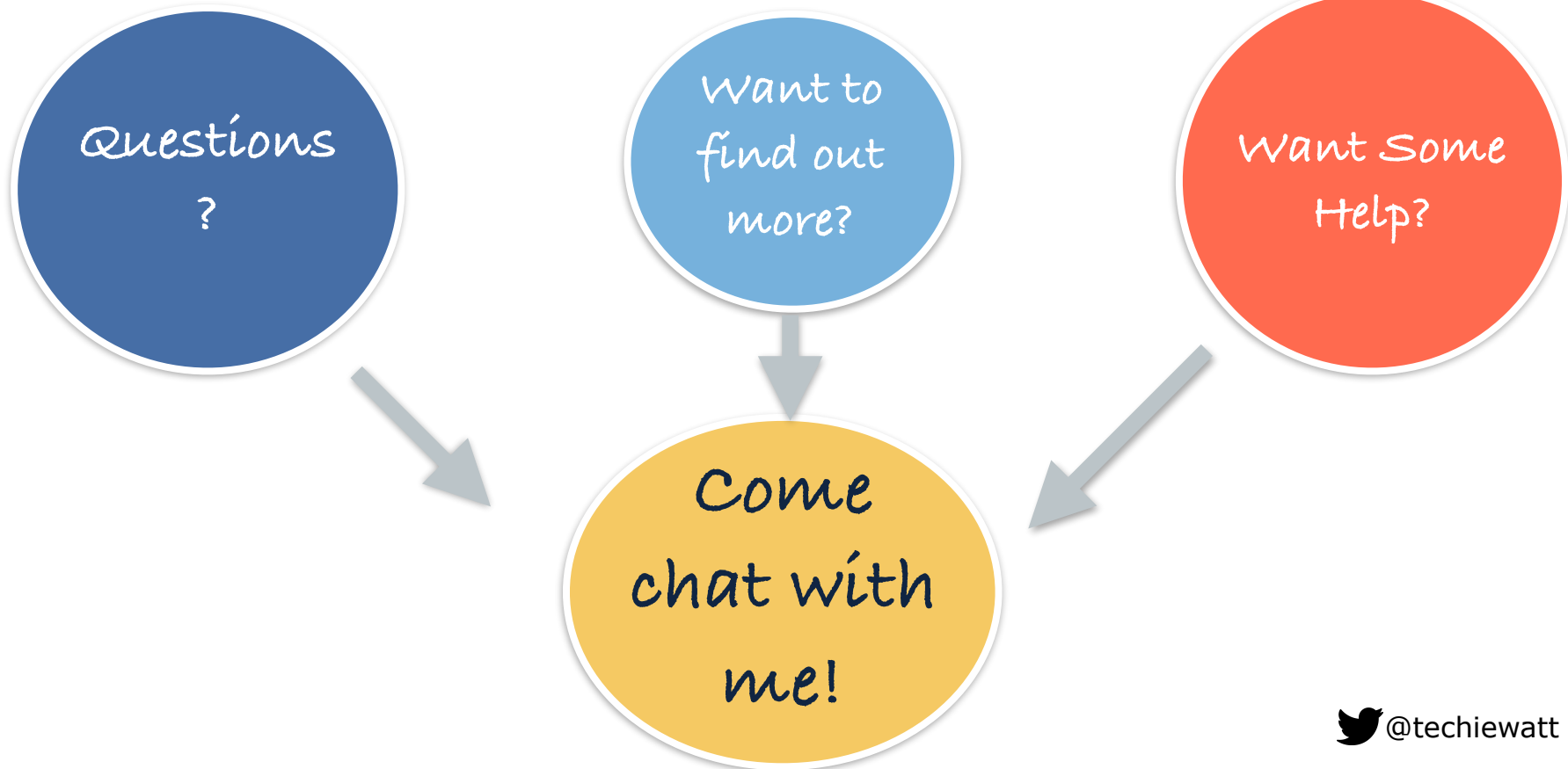
Demonstrated using **degree** and **cluster coefficient** measures to **detect tightly coupled (distributed monolith) architectures**

Demonstrated using **community detection algorithms** to **uncover related groupings (boundaries) of microservices**



RECOMMENDED LEARNING RESOURCES





Please

**Remember to
rate this session**

Thank you!

