**Slide 1**

Good afternoon, I’m Julie Fisher. Welcome to Not All Visualizations are Created Equal.

I’m a Neo4j Certified Developer, have been in the insurance industry for just over a decade, and have spent the last two years working on incorporating the benefits of graph and network theory into the processes and procedures for Asurion.

Asurion’s main product is cell phone insurance, which means we have claims and adjusters that make decisions about those claims. Our *mission* is more expansive: to help people protect, connect, and enjoy the latest tech.

A major component of my work has been figuring out how to visually represent data for analysts and adjusters so that they can identify patterns and trends in linked entities.

**Slide 2**

The prerequisites for this talk are pretty basic. If you’ve been involved with networks for long, these terms should be familiar. If not, welcome to the community.

Nodes, also known as vertices or entities, are the circles. A node usually represents some kind of noun: a person, place, or thing. For this example, we’ll call them people: Natasha, Clint, Wanda, Tony, Pepper, Steve, and Peggy.

The relationships, or edges, are the verb that connects the nodes: inspires, manages, tolerates, knows, friends; that kind of thing.

Direction is used to indicate when a relationship specifically moves from one node to another and not necessarily in the reverse direction. In our example, Natasha may keep tabs on Wanda, but Wanda has other interests.

Now to get down to the details of visualization.

**Slide 3**

I don’t know about you, but when I first start a project my mind goes to the biggest, coolest, most fanciest implementation ever. “I’m gonna chart the stars and see the cosmos!” Like in this

interactive chart from Barabási Lab called The Cosmic Web, where each node is a star and they’re connected to their nearest neighbors.

The the link to the project’s website is right here. Just a note, all the links are interactive in the slide deck if you download it.

**Slide 4**

So, instead of a comprehensive view of the cosmos, this is what we usually get, something that’s super basic and really very plain.

**Slide 5**

Just to reiterate, the unrealistic expectation is to throw data into a visualization and immediately have a beautiful, colorful, clearly grouped visualization.

The visualization on the left is from an article by Albert-László Barabási and his co-authors as published in Nature magazine.

Mr. Barabási is a network scientist, Professor of Network Science at Northeastern University, and author. His website generously posts many of his published articles. I’ll be referencing him and his lab periodically throughout the presentation.

The one on the right is from The Graph People blog. This blog covers many network topics, specifically how to implement them in Neo4j.

The titles are interactive links to the original sources, as is true throughout the presentation and slide deck.

**Slide 6**

Back to reality, what we get are things like this: visualizations that are hard to interpret, or where the colors representing group membership are randomly distributed throughout the network.

These problems are especially common when first starting out. They’re so common in fact, that indecipherable networks like these two have a nickname: hairball networks.

**Slide 7**

Where does that leave us? In diving into the topic of visualizations, it’s important to distinguish between publication ready visualizations and those suited for functional reality.

**Slide 8**

Publication ready visualizations are usually done by or in collaboration with a professional designer. For example, the Cosmic Web example from earlier in the presentation was done by Kim Albrecht, not Mr. Barabási.

Publication ready visualizations come *after* the analysis of data, when the findings are known and meaning already derived. At that stage, concentration can be focused on creating the prettiest and easiest to interpret display, not sifting through the data for meaning.

**Slide 9**

What I’ll be focusing on in this talk is the functional reality of creating graph visualizations.

It’s helpful to think of this as two categories: Investigation and Exploration.

**~~Slide 10~~**

**Slide 11**

The investigation category is very targeted and starts with an extremely limited subgraph.

In this example we have a single claim with the methods of interaction: phone, IP, email, etc.

From this initial, very limited, starting point, the end user can then branch out and look for connections.

Here I expanded this IP address, which is connected to 70 other claims, of which 32% have been positively identified as fraud. After a little finagling, the result is this network of organized fraud.

**Slide 12**

Exploration comes at the problem from the opposite direction. It starts with a larger subgraph, like the one on the left, then looks for patterns and trends within the targeted population.

This example shows dense clusters of nodes in the bottom left with smaller clusters throughout, but with the rest of the network more loosely connected.

The visualization on the right takes a representation of a subset of the IP addresses from the larger graph. Each node represents an IP address and the lines connecting them are claims where both IPs were used. This is an example of a chained network.

**Slide 13**

To get the most out of our functional graph visualizations, there are 5 best practices to be aware of.

1. Develop iteratively
2. Determine use case
3. Limit data
4. Structure purposefully
5. Explore layouts

We’ll go over each one individually…

**Slide 14**

…starting with Develop Iteratively.

**Slide 15**

Remember the super basic, boring visualization from earlier?

Starting with something basic and plain is okay. Not only okay, it’s expected.

I can pretty much guarantee that any of the visualizations you find appealing throughout this presentation didn’t start that way.

**Slide 16**

The progression on this slide represents the journey one of our visualizations went through at Asurion.

1. It started with the default view available through Neo4j. This is just the basic browser view.
2. Then it got dropped into Tom Sawyer Software. Tom Sawyer is commercial graph visualization software solution. I used default graphics and very minimal styling. You can see a first attempt at visual differentiation of the nodes with the little TV and cell phone looking pictures.
3. Finally, with the help of a lot of user feedback, I iteratively added features and functionality until I ended up with the view on the right.

User Feedback is one of the two most important things to consider when developing iteratively.

**Slide 17**

In fact, when done thoughtfully and in collaboration with your end users, incorporating user feedback can be the single most impactful thing you’ll do.

The example shown here revolves around phone numbers. I thought our adjusters would just need to know that a phone number connected two claims.

What they actually needed was to know was not just if they were connected, but also if the different types of phone number on the same claim matched: is the number on file the same as the one that the customer provided as a contact number?

This is only one example of where users improved the product. Even after months of collecting user requirements and working out how to translate those requirements into an end product, the contributions of the end users still resulted major upgrades. These improvements helped ensure the success of the project.

In fact, most of the best practices I’ll be covering in this talk were developed based on feedback provided by my users *in* *combination* with the visualization and graph theory research I conducted.

**Slide 18**

Another piece of user feedback brings us to our second iterative development: to visually distinguish between types of nodes and relationships. They hated when everything looked the same.

Like here, there’s no visual differentiation between nodes. Even the relationships are the same color.

If your graph consists of a single type of node and relationship, you may not need this step: I.E. node differentiation isn’t visually pertinent to you.

However, whenever more than one node or relationship type is present, visual differentiation should be considered.

The middle example is the same one from earlier. It was a first stab at node differentiation, but they were still too similar to allow for quick and easy reference. And it didn’t address any need for relationship differentiation.

The last example is the final product. The dissimilarity of the icons allows for quick and easy reference, while the number and colors of the relationships provide information the adjusters need to do their job.

**Slide 19**

The examples so far have been very focused on a claims perspective. There’s a reason for this, and it’s not just because that’s the project I’ve been working on.

It’s because the second Best Practice is to carefully consider use case.

**Slide 20**

Graph databases and relational databases are the same in that the use case determines how data is stored, how it’s indexed, and how it’s structured.

In graph databases this structure is called the Graph Model, and it affects what’s available for visualization.

I’ll cover graph structure more under Best Practice number 4, but first let’s discuss a particular use case using a Case Study.

**Slide 21**

Cell phone insurance tends to have a little faster turnaround time than other types of property & casualty insurance.

If I remember correctly, state mandated timelines usually allow for a couple of days to make a liability determination for a car accident.

Asurion, with cell phone insurance on the other hand, has been offering next day replacement for 8 years and started offering limited same day replacement last year.

That’s not a lot of time between receipt of a claim and completing it.

**~~Slide 22~~**

**Slide 23**

A lot of times, when looking for suspicious behavior or fraudulent patterns our adjusters are looking through something like this. Rows upon rows and column after column of data.

This particular dataset is from the Panama Papers, but the concept is the same. Trying to find suspiciously linked entities is difficult in this format.

To give you a feel for just *how* difficult…

**Slide 24**

…I put together this little 4 row example.

One of the major strengths of tables is the ability to sort by the values in a column. This allows us to see things like, row 1 and row 4 share an email address because they’d be listed together after being sorted on the email column.

That feels scalable to large datasets.

But what about sorting values across columns? Can you see the two rows where the cell phone number from one row matches the other row’s home phone number?

It’s rows 1 and 2.

How about how strongly two claims are connected? Can you find the two rows that share two connections that aren’t the phone numbers?

It’s rows 1 and 4.

With internal audiences at Asurion, this exercise generally takes about 30 seconds to a minute for each question. With just four rows.

Think back to the Panama Papers example, there were hundreds of rows, of which I included just a small sample.

**Slide 25**

When we switch to a visual representation of the data, the answers to both of those questions becomes much clearer.

We can quickly see that these two are connected via IP and email, and these two share two phone numbers.

Something you may not have noticed in the table view, one of these claims isn’t connected to any other claim, and one of them is connected to two claims.

This, by the way, is one of the early conceptualizations for the claim visualizations I’ve been using as demos. It’s a manually created Gliffy diagram.

This just goes to show: start where you need to. If you keep iterating, a finished product will emerge eventually.

**Slide 26**

Next we move to the third Best Practice. This one was a close contender for the first and most important Best Practice. If you only remember one thing from this presentation, remember this: Limit Your Data.

**Slide 27**

When data is limited, it’s much easier to determine if there’s a there there.

Here we have the chain of IP addresses again, as well as a string of claims where the user is attempting to replace the replacement device under a new account.

It’s super tempting and if you’ve been doing this for any length of time, we both know you’ve tried it: …

**Slide 28**

…dumping everything into the same visualization.

Unfortunately, the results of this temptation almost always turn out the same: unusable.

The visualization on the left is a single week’s worth of claims. Do you see anything useful in there?

Yeah, me neither.

The visualization on the right is every graphable data point we had associated with an organized fraud ring. Can you see the connection that distinguishes these as fraud?

Neither could the adjusters.

There are quite a few options when it comes to limiting data.

**Slide 29**

The first is based on the investigation approach. Start with a single entity and expand from there.

Our adjusters found it much easier to start with the single claim they were working, then determine if there were any suspicious points they wanted to investigate.

For example, the text box looking thing represents free text provided by a customer. Free text is when a customer can type whatever they want, however they want to type it. If that free text is exactly the same on 63 other claims where 70% of them have been positively identified as fraud, that’s something an adjuster may want to investigate.

If there are no other links between the claims, it’s probably what we call a hub or celebrity node. Something simple that lots of customers just happen to type exactly the same. An example for an auto claim may be something like “I hit the garage.”

You can see, that’s not the case in this particular grouping, there are several dense clusters.

**Slide 30**

Once the data gets larger, like that expanded subgroup from the last slide, we have to switch tactics. Now we need something that works with the Exploration method. This tactic is filtering.

I took the data from the subgraph on the left and limited the nodes to just claims and emails. I filtered out phone numbers, IP addresses, that kind of thing.

By limiting the nodes, it became clear that most of these claims are connected via a string of email addresses.

This kind of entity linking is not readily apparent in tabular data. Sorting columns will give you some of these emails, like this one and this one, but it won’t reveal the connected network.

**Slide 31**

In addition to filtering nodes, we can also filter relationships. Chapter 3 of Networks, Crowds, and Markets has a great example of this. They started with Facebook data, then created multiple visualizations progressively limiting the relationships.

The first establishes is the normal network. An edge is created as long as there is an acknowledged connection. Basically, anyone that you’re “friends” with.

The second limits the relationships to those that are actively maintained. I.E. the people you interact with regularly.

The third calls out relationships where communication is one-side, one party has messaged the other without response.

The last is the most restrictive. It only connects people that have mutual communication. I.E. both parties respond to each other.

As you can see, the number of relationships decreases as we get more restrictive in our definition of a relationship.

**Slide 32**

The last category for limiting data may seem like it was already covered under filtering. However, hiding unwanted data can’t necessarily be done with a filter.

A filter implies that all of one or more categories is removed holistically. Unwanted data may or may not fall into such neat, all encompassing categories.

For example, here’re those claims where the customer re-registers a device under a new account and makes another claim.

In the top example, there are multiple claim entities connected to several of the devices. This extra information makes the picture harder to interpret.

To make it easier, the duplicative claim nodes have been hidden in the second visualization. Now, it can easily be seen that the customer is activating the replacement device under a different account, then making a new claim on that replacement device.

But it isn’t just what we show that makes a visualization usable, it’s also how we show it.

**Slide 33**

All of the examples we’ve seen have purposefully used structure to their advantage. In Best Practice 4 we’ll discuss three aspects of structure: nodes, relationships, and direction.

**Slide 34**

Nodes can be structured several ways. The examples I’ve shown so far fall into two categories: Unimodal and Bimodal.

Unimodal is like our social network with Natasha, Clint, and company. Each node is a person and only a person, which is why they’re all blue.

The bimodal structure is used to represent graphs that have two kinds of nodes. So if our people belong to clubs, we could represent this as a bimodal network like this.

Bimodal networks tend to only connect unlike nodes. This can be seen here, where the blue, people nodes only connect to tan, club nodes and the tan, club nodes only connect to blue, people nodes.

It makes sense, as a club can’t be part of a club. And a person can’t be a club, that another person can join.

We can also see the impacts of layouts here, in that it is very difficult to discern that nodes 3 and 10 aren’t connected to any other nodes.

**Slide 35**

How does this translate to our real-world examples?

In the left-hand visualization, we can see the Kaggle competition dataset: The Marvel Universe Social Network. Each node represents a Marvel character. The characters are connected if they appear in the same comic.

The nodes have been colored based on the results of a community detection algorithm and clearly show three large character clusters: Spiderman, Captain America, and X-men.

The right-hand visualization is our claim graph again. This bimodal representation has claims as one type of node and contact points as a second type of node. It follows our general rule of each node type only connecting to the other type and never to its own type.

**Slide 36**

There is a caveat to representing the claim network using a bimodal structure. You may have been wondering why I lumped all the contact methods into a single node type.

It can also be represented as a multimodal network.

A multimodal network is when there are more than two types of nodes.

If we considered each contact category as its own node type (like phone, IP, email, etc), then this would be a multimodal network.

Using a multimodal structure can be very helpful when considering filtering tactics.

**Slide 37**

When working with bimodal or multimodal networks, it’s important to understand that they can also be represented as a unimodal network; this is called a projected unimodal network.

Here’s our bimodal claim network again. We can simplify it to connect claims directly to each other. Using this structure may help reduce noise and allow end users to more readily identify patterns or trends within the data.

**Slide 38**

An interesting structure that I won’t get much into is called multilayer. An example would be a transportation network. In this example we’ve got trains, subways, and streets.

These different transportation methods don’t interact directly, but a person can easily move from one to another. So when you arrive somewhere via train, you can walk to a taxi stand and get a ride to the subway station to continue the next leg of your journey.

**Slide 39**

Next we have relationship representation. As we’ve seen in the examples, relationships can be represented by a single line or multiple lines.

If we use our people joining clubs example, if Natasha and Wanda belong to 3 of the same clubs, they could be connected by three lines (as shown here). Alternately, we could connect them with a single line if we just want to know that they belong to one or more of the same clubs.

If connected via a single line, you can add a property called ‘weight.’ This property records how many clubs they both belong to without cluttering the visualization with too many lines.

In this view you can see that the weight is represented in the thickness of the line. Here Natasha and Wanda’s relationship has a weight of three, whereas Natasha and Pepper’s relationship has a weight of one.

**Slide 40**

Returning to our claims example, we can see how single relationships and multiple relationships can be used to convey different levels of information.

As mentioned earlier, the adjusters needed to know if types of relationships matched, not just that they were present. As such, I opted for multiple relationships to make clear different kinds of relationships between a claim and a point of contact.

**Slide 41**

Direction doesn’t play heavily into the claims examples, but I wanted to demonstrate how drastically a node’s importance can change when direction is taken into account.

These two graphs have the same structure and the same relationships. The only difference is that one has directed edges.

An example of this would be Facebook vs Twitter. In Facebook, the connection has to be acknowledged by bother parties. In Twitter, one person can follow another without the other person following back.

Now, nodes 3 and 5 have the same number of connections in both graphs: 5. The number of connections is also known as a degree, so they have a degree of 5. Because the degree is the same, they’d be given the same importance.

However, when we take direction into account, node 5 has 3 incoming relationships and node 3 has only 1.

In this case, node 5 would have more importance for any metrics considering in-degree and node 3 would have more importance for metrics considering out-degree.

These considerations will be important when you start working with centralities, like PageRank.

**Slide 42**

PageRank, the algorithm used by Google to return relevant search results, uses in-degree vs out-degree to help determine importance.

Here’s a very high-level overview of this algorithm.

Let’s say we want to rank Towards Data Science, the data science oriented section of medium.com, and my little blog, Data Science Diaries. We need to return the page that is most likely to be what the user is searching for.

For the sake of argument, let’s make the unrealistic assumption that both Towards Data Science (we’ll call them B) and my blog (we’ll call me E) have the same degree – so the sites link out to and are link referenced the same number of times.

Because Towards Data Science is so much more prominent and well known, a much larger portion of its degree are links reference its content. My little blog on the other hand, points to many other websites as references.

Because other websites link to Towards Data Science, it is considered the more trust-worthy result and will rank higher.

**~~Slide 43~~**

**Slide 44**

This brings us to our last Best Practice: layouts.

**Slide 45**

Here are several layouts from Gephi.

I took a subgraph of heroes connected to Captain America from that Marvel Universe Social Network dataset from Kaggle and just cycled through some of the available layouts.

The first visualization is the default one when importing nodes from Neo4j to Gephi: relatively evenly spaced in a square.

In the second, the nodes have been packed as tightly as possible into a circle.

The third is the Yifan Hu layout. This layout employs a variation of force direction.

**Slide 46**

Force-directed layouts are by and far the most popular layout component to include in a visualization.

Force-directed algorithms attempt to space more similar or more densely connected nodes closer together and less similar or more weakly connected nodes farther apart.

We can see three tight clusters in this example, one of which is loosely connected to the others and thus pushed farther away.

**Slide 47**

Force-direction algorithms usually attempt to space nodes so that overlap of both nodes and relationships is minimized to the best of the algorithm’s ability.

Here we can see the implementation in both Tom Sawyer Software and Gephi.

There are several types of force-directed algorithms, the details of which are outside the scope of this talk, but if you want to know more, I’ve included some books in the Resources slide at the end of the presentation.

**Slide 48**

Once a force-direction algorithm has been applied, there are several choices of layout.

This is most common layout is where nodes are arranged in organic, circular patterns.

Another common layout is the hierarchical layout. The example shows claims and contact points, but this layout is much more common in networks that represent some kind of hierarchical structure (thus the name).

So, down here, I report to my boss, my boss and his co-workers report to their boss and so forth.

The other two layouts on the screen are Tom Sawyer’s Circular layout and what I call the Circuit layout because it reminds me of a circuit board.

Exploring different layouts is an important part of the process. I recommend trying out all the layouts available in whatever software you choose.

You never know what the end users will prefer, and looking at data from different perspectives can yield surprising and informative results.

**Slide 49**

The last concept I’d like to discuss is combining elements.

Node combos involve grouping like nodes, then limiting the relationships to just those connecting the groups.

These visualizations are both from Cambridge Intelligence; a graph visualization company that has multiple SDK options available.

The first one shows the full graph with all nodes and relationships drawn explicitly.

In the second one, the dense clusters have been grouped. In this example, the clusters were by geographical region.

**Slide 50**

Returning to our claims and devices example, we could have obtained the same results as manually hiding data if we’d been able to group claims.

The desired node combos are circled in yellow.

The grouping concept can be applied to relationships…

**Slide 51**

…this is called edge bundling.

This example come from the paper Edge Bundling in Information Visualization. The authors used flight data.

In this first view, all nodes and relationships are drawn as usual. Then someone used an overlay to depict common flight patterns.

The second visualization bundles those relationships to clearly show common air traffic patterns over the US.

**Slide 52**

These 5 Best Practices allowed me to develop graph visualizations for Asurion that have helped us identify new patterns and trends that we were unable to ferret out using tabular data.

The most important one to remember, especially when first starting out is Number 3, Limit Data. Staring with too much data to realistically show can doom a project before it even starts.

**Slide 53**

A quick word about visualization software.

The five used in this presentation were Gephi, NetworkX, Gliffy, Cambridge Intelligence, and Tom Sawyer.

Each has its own strengths and weaknesses. I’ll briefly cover my opinion based on my personal experiences.

NetworkX and Gephi are both free. Because they’re free, expectations should include bugs and limited customization. However, they’re both well established and mostly stable platforms. These are a great place to start for an individual.

Gliffy is available in Confluence. These diagrams are created entirely manually and I only used it to create small examples to demonstrate graph concepts.

Tom Sawyer Software and Cambridge Intelligence both put out quality commercial products.

Tom Sawyer has a GUI that allows for faster prototyping without the need to code. This was a major benefit when I didn’t have a developer available to assist with visualizations.

Licenses were also available in small quantities, which allowed me to prototype without having to get approval for large license fees.

Cambridge Intelligence has two products I’m familiar with: KeyLines and ReGraph. They are very similar, the differences mostly cater to compatibility with different platforms, like React. To the best of my knowledge, this solution offers the highest level of customization for the least effort when moving into a coded solution at an Enterprise level.

There are many other solutions out there.

**Slide 54**

I’ve included a list of the visualization tools and software that I’m aware of here. I haven’t used all of them, so explore at your own risk.

There are many other techniques and methods to apply to graph visualization. If I’ve whetted your appetite, and I hope I have, you can get more information from the sources listed under Books, Articles, and Blogs.

The last link is to a list of books about all manner of visualization topics, not just graph specific.

**Slide 55**

If you’d like to get in contact with my, you can find my LinkedIn page here, or my GitHub page here.

The slide deck and version of the talk are available on GitHub under the Conferences folder, which can be reached here.

If you liked my way of presenting information, you can also check out my Blog, where I dive into Data Science concepts.

Last, but not least, if you’d like to know more about my employer Asurion, you can check us out here.

I hope you enjoyed the talk and hope you enjoy the rest of the conference.