OPUS No-Code Fundamentals

Reference
Link: https://www.tracelink.com/resources/tracelink-university/opus-solution-designer-foundations-presented-futurelink

Jerry Meyer: Hi, everybody. My name is Jerry Meyer. I've been with TraceLink for just around 10 years as well. If you know anything about our company history I also worked with Shabbir and Lucy at the previous company called SuppyScape.

I want to just go a little further here. In addition to supply chain and compliance backgrounds, I've also worked in the discrete manufacturing space for over two decades, building CAD and product life cycle management, no-code solutions, and applications. It's very appropriate for our no-code fundamentals that I could share a little bit of that background with you guys.

Let's start off. We're going to start off with four personas. These are going to be consistent throughout the day. We're talking a lot about the solution designer, the open solution environment. The first three personas on the left, they're all solution designer personas, but they have different goals and they're very different people.

On the far left, we've got the OPUS developer. Now their goal as a OPUS developer is to develop multi enterprise applications and standard solutions for those applications such that they can be used out of the box to exchange data with your trade partners, and they can be integrated with enterprise systems and trade partners using transforms and link actions. That's the goal of our TraceLink's developers these days.

Moving over to the second persona, it's all about solution designers who are solution partners. Our partners that want to go out and build solutions that are targeted for a specific market segment within our industry.

These solution designers, their goal in life is to build marketplace solutions that they can then make available to our OPUS customers using the marketplace solution catalog. Bob talked about that this morning a little bit about all the different catalogs we have.

The third solution designer persona we'll also cover is your company, if you're a customer of TraceLink. In this case, the solution designer wants to configure their company solutions to any unique requirements that their users have, that their company needs. They want to do this in such a way that they still maintain TraceLink's, Integrate Once, interoperate with everyone, core tenet.

That's a fundamental tenet of our company and of our system, and we never violate that.

Any solution developed in any of these three ways, standard solutions, marketplace solutions, or company solutions, they all adhere to that, and they're consistent in how they behave. That's because of the system that we put together.

The fourth persona, the last one now is a little different. This is the end users. End users have a goal of, in some cases, they want to use our standard solutions. This is important in the cases where you want to stay up to date with the latest and greatest releases of software.

Very necessary in, say, a compliance application where you want to get into validation as soon as it's available in the validation environment, go through your testing, and be ready for when that standard solution hits the production environment. You don't want to do any customization if you don't have to. You can do that because all these solutions work out of the box the way they're intended.

These same business users also want to use company solutions that have been specifically tailored to their business needs. Every company has some unique business needs and they that need to configure different things, have different behaviors specific to their company and how they interact with their trade partners.

Because all these OPUS solutions what you're going to learn about is this opinionated user interface. We we'll use that term fairly often throughout the day probably. It's easy to use across any of these solutions because it has a standard pattern for how screens flow with each other, and all solutions, regardless of whether it's the standard, the marketplace, or the company solutions, they all follow those standard patterns.

This is important because your enterprise users as well as your partners will require less training than you would otherwise potentially need if you had just random applications that were built from different companies in different ways.

With that said, let's go ahead and talk about some of these fundamentals.

Now, no-code, there's a lot of no-code solutions out there in different industries and they all have a couple of things in common. One of the things that they have in common is they tend to make the no-code behavior something they can use to drive the user interface and that makes it easier for you to build screens and make things available to users. That's one aspect of it.

The other aspect of these no-code systems in the different industries is they will fine tune or tailor the features and functions that they make available in that no-code environment that target specific challenges and problems of an industry.

As an example, let me take the customer relationship management industry where you've got Salesforce which probably most people in this room know about Salesforce.

They have fine-tuned and optimized their system, their no-code system for the sales pipeline and making sure that they can go from opportunity to sale in a very predictable manner with very high confidence so that the management of the company knows, hey, how much money can we count on next quarter or over the next year?

As they're budgeting and figuring out where to where to put money, where to spend, they have a confidence that they're putting in the right places and they know how much they have to spend. Likewise, with doing any kind of, you know, public reporting and also they want to know that information. So Salesforce is tailored and fine-tuned for that industry in that way.

Another one, if there's probably a fair number of IT professionals in this room and you guys work with software, it's pretty likely you've run into Atlassian, Jira, bug tracking and project

management tool used in the software industry.

Now, those guys, what they've done is they've tailored their software package to optimize around source control systems, test automation systems, and other software development tools such that developers are very efficient at doing their work. They optimize for that industry.

TraceLink, on the other hand, what we've optimized for is in the B2B space, exchanging data with trade partners, multi enterprise applications, where you have trade partners exchanging data with very disparate systems, with varying levels of technology and sophistication with regards to how they've integrated with them and what technologies and capabilities those systems have. That's where we've optimized and focused our attention.

What TraceLink also is doing is we are pushing...We're going to talk about a few layers of the system around how we do this with metadata management. What TraceLink is doing is we push the definition, the metadata definition that drives our no-code environments as far down the stack as possible.

We do have three layers. We start off at the very bottom with the metadata driven business objects. There, we have things called objects, business objects. We have fields operations on those objects. Then we move up a layer and we talk about the metadata-driven user interface. Here, we have things like search pages, new pages, view edit pages, dashboards, and reports.

Then as we move up even another layer, the solution itself, we now can really tie all this stuff together and really tailor it for your enterprise using menus, roles, and permissions. Those are the different layers of our no-code system and we're going to be talking about each of them throughout the day and throughout this presentation as well.

Let's talk about that lowest layer, the business object layer first. We have something called business objects and when you think about a business object, you think about what are the nouns? What are the things you work with every day?

Business objects represents sort of the domain of an application. There are things that the users of that application are going to be familiar with. Examples would be invoice, purchase order, quality documents. These are things that people work with purchase orders on a daily basis, they're going to understand what those objects are. They're going to know. They're going to see them in the left nav menu.

We also have something called fields. Fields, when you think about those, they're just characteristics about the business objects. It might be the supplier name, the customer name, product code, problem description, or what are the reference transactions for this transaction I might have in my hand?

Then there's operations. Here you want to think about the verbs. What are the actions a user can perform on these business objects? Some actions might be I want to create a new PO, I want to edit a PO, or I want to change the state from drafted to submitted when I want to put a transaction through the system. Those are the kind of operations you can perform.

Now, I mentioned earlier that the various low code systems out there or no-code systems out there will sort of fine-tune their system for the domain, the industry that they're targeting. I'm going to go into a couple of little topics here on the fields.

We have two kinds of special fields in our system. One's called LBV and the other one's LBR. Look up by reference and look up by value. Now look up by reference, you're going to be able to relate to. Most systems have something like this. You can think of, in the relational database world, you might have a table of data and it relates to another table of data.

Say, you have a customer record and you want to know what products that customer licenses. So you're going to have a reference from that customer table to the products table. It's a foreign key. It's a way you point at a specific record in that other table. Very tight integration there. You know, when you click on something you go to that exact record.

Lookup by value is very different and it's very unique and necessary in the B2B domain where that record that you're trying to get to or that you're interested in may or may not be in the system that you're interacting with. As an example, let's say you get an invoice, and that invoice references some purchase order 123 and advanced ship notice 456.

Maybe you've integrated your invoice with your partners, maybe only some of the partners. Others partners you haven't done integrations with. Maybe those purchase orders show up at your front doorstep via an email attachment. Maybe those purchase orders haven't gotten through the clerical data entry that got them into the system yet. They are sitting on someone's desk waiting to be processed.

Or maybe you don't integrate them at all. Maybe you don't enter them at all. Maybe you [indecipherable] at them and you just go look it up when you want to figure out is this invoice really authorized.

ASNs may have another kind of problem. They may be stuck in some B2B translation layer along the way and it just hasn't shown up yet. It's very important to be able to reference transactions that may or may not be there and still function properly.

That's what this lookup by value is all about. It's a very specialized B2B supply chain need and it's important when you think about all the disparate systems, all the various trade partners you're interacting with, and the various levels of sophistication they have with respect to integration.

Jeff, this afternoon, will talk about a lot of the technologies and techniques that we have for integrating with people. At the end of the day, if none of those work because they're so low in sophistication or so small of a shop, we still have a user interface that automatically gets populated, which is cool. And we're going to talk more about that.

I'm going to move on. This clicker is not...here we go. I have to point it, I guess. That's what it is. OK. That was that lowest level. I'm going to come back at that low level again, business object layer. But I'm going to switch gears and get to something more tangible, GUT screens and UI, people can relate to UI little bit better than business objects and fields.

The business objects in our system all show up in the left navigation menu. You'll see things like purchase orders, invoices, ASNs.

When you click on one of these menu items over in that left nav menu, it's going to take you to a certain kind of screen. I'm showing here a search screen. There's other things that it might take you to like dashboards or reports, but I'm going to focus right now on the search screen as far as from the left nav menu, and we'll come back at the reports a little bit later.

Then, once you have that search results on the screen, you can click on any one of these rows, and it's got a PO number in it, and it knows how to take you into the view edit page. It's pretty common, pretty intuitive behavior.

Also, you'll see a toolbar across the top which has a new button in it. You click the new button, it's going to take you to create a new purchase order page. Very obvious, very intuitive. And then from there, you can edit and stuff. That navigation, you don't have to wire that up yourselves.

Again, I talked about our opinionated user interface. When you define your business objects, the fields that they have and the operations that they support, this all happens. These pages

are written. They're written once and they're never written again.

There's metadata that as you drag things on the screen, as you say, want to create a search page for a purchase order, whether it's a sent purchase order or received purchase order or whatever other business object type you might be thinking about, it understands this kind of navigation and a lot of other navigations.

I'm not going to go into them all right now. You're going to learn those throughout the day. But that's what's going on here. This is when we say opinionated user interface, we're referring to this type of behaviors.

Now subtyping is a very...I'm back in the business object layer again. Subtyping is a very important tool in the toolbox for our solution developers. They are going to leverage this like crazy when you start talking about MINT. What I'm going to do before I get into exactly what we did with MINT and what you'll be able to leverage, I'm just going to read the theory because it's a sort of cool thing here.

Now, when I talk about subtyping, you're going to hear supertype, base type, class, you know, depending on what programming language you're familiar with or what system you're familiar with. Any of those terms might apply.

In theory, subtyping is a notion in programming language theory where a subtype, which is a data type, is related to a supertype based on the notion of substitutability. That's the key word here, substitutability, where programming elements such as functions and subroutines that are written for that supertype will still operate if they've been handed the subtype instead.

That's a very powerful concept. Again, IT professionals in the room, you've probably dealt with subtyping in one way or another, subclassing, whatever it might be.

Now, in practice, what do we do with MINT? In MINT, we created something called a business transaction object. That's our base type. That base type knows everything there is to know about B2B, data exchange, and pipeline processing in the generic sense.

It knows how to take a file that lands on our front doorstep via some transfer protocol, AS2, SFTP, HTTP, SMTP, doesn't matter. Something lands on our front doorstep or gets entered in the UI, doesn't really care. It lands on our front doorstep. That BTO object, it knows how to transform that data into our internal canonical. Canonicals are TraceLink's internal representation and view of a given transaction.

Now, once it's done that, it also knows how to figure out who is the target recipient for this message, whether it's one of your enterprise systems or one of your partners'.

Once it's completed that little piece of processing, it then takes that canonical that's been transformed and it makes it available to the MINT application. I'll come back to why it does that in a minute.

It also transforms the data one more time into a more flattened user interface friendly view of that data. Canonicals tend to be very deep in structures, very complex, not something the user wants to click through on the UI and expand out and fill in fields. It's way too complicated. This BTO logic knows how to call and get that information flattened out in a more user interface friendly way.

It handed it off to the MINT application. Because it handed it off to MINT and because as customers or as solution partners who might be tailoring these subtypes...this is where you can insert logic and stuff.

We hand it off to MINT. MINT does whatever it does based on configurations that people have made. Then, once it's done, it takes it back and it goes through another transformation

process on the outbound side, and it delivers that message via whatever again transfer protocol your partner might be interacting with, AS2, SFTP, etc., and delivers it to the partner.

All of that is programmed into the BTO object, that base object. It doesn't matter what kind of transaction we're talking about. That processing is the same and it's very repeatable, and we do it millions of times a day.

Now, how does that help us in MINT? Because MINT has real transactions, purchase orders, invoices, etc. Well, every transaction in MINT has been subtyped below this BTO. We have sent purchase orders, received purchase orders for your external manufacturing. We have sent and received purchase orders for your commerce, etc., we've got invoices. We've got ASNs.

All of these objects are subtyped below the BTO object and they know all that transaction processing. They know how to do that. They don't have to do anything special. It's already coded up.

Now, what does a MINT developer have to do then? Jeez, it sounds like you did everything. No. There's still work. [laughs] Whenever we come up with a new type of transaction, purchase order versus invoice, there's a lot of research that goes into developing what that canonical looks like.

We do a lot of analysis across the industry, across different suppliers that are out there, vendors that you're all very familiar with. We look at a lot of standards, and we come up with a superset of all the fields that are going to be necessary to properly process that transaction.

Once we've done that canonical, we've got it. The MINT team, all they have to do after we've defined that canonical is define a few transforms, drag and drop a few screens, and they're done. We were talking about minutes to create screens, and it's interesting. I know this. I can't believe how fast they did this.

In the past month, they created 60 subtypes and built over 300 screens for those subtypes. We're just a small handful of people dragging and dropping. It's crazy cool stuff. I know they have on the road map another almost 400 to get to. It's like they're cranking through it. Anyway.

I'm going to move on now. It's important to note, this is another one of those features that we've built into the OPUS platform capabilities, tools in the toolkit for any solution designer that is tailored for B2B data exchange. We got all these disparate systems, all these different levels of technology, and those sophistication with integration. It's a very important feature for us.

A little bit more on business objects. I'm going to stay at that later for just a little longer, and then we'll go back to the UI again because that gets more fun.

We talked about primary business objects and the fact that we have fields that we can add to them, characteristics of those business objects. We didn't talk too much about the details of those fields though.

Well, fields might be simple things. Due dates, problem descriptions, LBR, LBV fields. We talked about those a little bit. There's all kinds of relatively singular fields that we might be talking about here. Some are simple, some are less simple like workflow. We have a workflow field type and that's used to show you the state of an object. You'll see that in a moment. It's pretty cool.

We also have a concept of groups. Remember, our data is complex. It's B2B data. It's not

just simple fields that you can drag in and drop on like a Salesforce can do with sales deals and opportunities. It's much more complex than that.

So groups, what are groups? Well, think of ship to address, ship from address. Those are sets of fields that you want moving around together. When you drag a ship to address on the screen, you want to get street 1, street 2, city, state, region, postal code, whatever it might be.

Same with ship from, same with carrier, and what business to, business from, there's all that stuff. You just drag the whole group on and you get all that behavior built in. Also, we have something called collections. When you talk about collections, you want to think about arrays.

I've got a collection of related transactions. Your invoice talks about what PO's authorized. It talks about what ASN's might have been there to deliver it, etc.

You can also have collections of groups. Think about your purchase order. It has order item details. There's all kinds of information inside that order item details, quantities, unit measures, geez, I don't know unit cost, all kinds of things that might be in there, product descriptions, product codes, all that.

You can have fields, groups, which are groups of fields, or collections, which should be collections of fields, or collections of groups. All this capability is built into that low level metadata layer. Once it's defined at that layer, everything else in the stack, the UI knows how to behave, whether it's a search page, a view edit page, a new page, or as you're going to see in a little bit the reports and dashboard pages. It's all pretty cool.

Next, we talked about object operations. When you go to create a page, you're creating a view edit page for, say, a purchase order or you're creating a search page for an invoice. Well, those pages know what kind of operations they support. Obvious ones are search, do, view, edit, but then there's others too. There's copy, there's follow, there's a bunch of other standard operations that import export.

We can just make available to those pages automatically because we know what those pages are supposed to do. Now you can control who has access to them via the whole solution metadata with your roles and your permissions. You can turn it on and off for people, but it's there automatically if you want it.

We also have something called specialized operations, which we're not going to get into today. I will show you an example of some on one of the last slides because that's also really cool.

Then we have the last thing for business objects that we can tailor, that you can configure are the workflows. Every business object comes out of the box with a standard workflow. All the subtypes inherit those workflows automatically. But then customers are going to, they might have some kind of a PO approval process that they have to go through with the dollar amounts over a certain thing.

Fine. Add an approval process substate in there and detect when the dollar amount's more than some amount, then kick off an approval process. Or if you don't have such a thing, when that transaction gets submitted, it goes right through the system, spits out, and lands at your partner's front doorstep. Either way, it doesn't matter.

That's another area with the substates and the transitions between the states that you can configure and add those, again, using the whole drag and drop capabilities we have.

All right. Let's go back up to the UI now. It's a little bit more interesting.

I'm just going to go a little bit deeper, not too deep. But here I'm going to stay on search

page, view edit page, and the new page because that's consistent and easy to keep on track here. All of these pages have two general areas. They have more than that, but I'm going to just talk about two. They have a content area and they have toolbars.

The content area is where you see your fields. If you're on a search page, that content area is going to look like a table where each field is a column. Of course, when you define your search page and you drag fields like order date or state on that, it's going to know how to behave. It's going to know how to sort. It's going to know how to be filtered. All those kind of things come with it.

When you drag those same fields onto the view edit page or the new page, you're going to get a different behavior, one that's defined again at that metadata layer. If I may have a date field, Bob already mentioned when you pick it, you get a date chooser.

If you're dealing with workflow, you get this really cool workflow, the state of an object, this really cool workflow visual thing that you can say, hey, move to this state. It has to honor the rules of the workflow that have been defined for that base object type.

Then likewise, for the subtype that you've defined with your substates, it has to follow the rules. But it knows how to do that. You just drag the field on the screen and that cool life cycle shows up for you.

Other things about this, in addition to the fields then, we've got the toolbar. The toolbar is where all the operations show up. Your new operation we've already talked about. We've got import export up there. I've got copy, follow some other, you know, very easy to understand operations. But then there's a couple of other ones that are neat. I mentioned specialized operations.

Over here on the purchase order details page, I've got a PO acknowledgment, ASN, and an invoice button. What's that all about? Well, think about it.

Purchase order comes into you or you send a purchase order to your partner. It's got lots of detail, a lot of line items on it, well, they can just hit that PO acknowledgment button and it'll create a PO acknowledgment. It'll copy all the relevant fields from the purchase order, automatically fill in all the order item details, and then let the user edit the details.

Maybe you ordered a 100 of some product and you only have half of that inventory in stock. You can edit that PO acknowledgement, split the line item, and say, I can deliver 50 of those items on the date you need it, and I can deliver the other 50 two weeks later. Is that OK? Send that acknowledgment back, and it's done.

Now, this is for somebody who might have partners, let's say, who's not tightly integrated with the system. They maybe don't have a lot of sophistication. They can use the UI to do that.

If they're integrated, say, via NetSuite or Microsoft Dynamics with, say, LinkActions, they can be working in their little mid-tier ERP system, do the same PO acknowledgment there, and send it back, and it goes all the way back in through into you. It doesn't care that you were doing EDI EDIFACT. You're going to get a PO acknowledgment in EDI EDIFACT world regardless whether they use the UI or a LinkAction via their ERP system.

All of our screens follow the same content, content area, toolbar, and there's a few other areas that Steve was going to cover when he goes into the next session.

Dashboards, reports and dashboards, another opinionated user interface that we've built. In this case, instead of seeing the search page when you click on one of those left menu items, what you're going to see is either a dashboard or a report. These also have an opinionated behavior or built in navigations.

For example, there's a filter bar that will come out and you can filter on how you want to filter that data. Think about this for just a minute. The way that dashboard got built, someone drug a report on those charts on the screen, defined some query objects for them and said, here, this is the data I want to see. That filter bar, they drug a few fields. Here's the kind of fields I want to filter by. Great.

Additionally, if a user ends up clicking on a specific area within a chart, it's going to go and show you a small tabular report of just that data. If you click on the view report for an entire chart, it'll bring up a tabular view of all that data.

Saurabh is going to talk about all these details in much more granular and, actually, I think show a really cool demo as well.

I can't believe when I saw the demo recently how fast the system it's like you tweak one of these filters, that whole page, all those dashboards, all of a sudden start changing. This moves like this. This goes down. That goes up. It's crazy stuff. Really fast. Really exciting. Couldn't believe what that team achieved in a really short period of time.

Anyway, fun fact here. We keep talking about purchase orders, invoices, and other business objects that business users are going to be familiar with. But if you look over here in this menu, what do we have? We've got dashboards, reports, and query objects. You think to yourself, well, that doesn't look like a purchase order or an invoice or any of these other things. No.

But they are the business objects of the Report Builder application. We're actually using OPUS to build all of our OPUS applications on top of it. The very first one, we had to go in there and insert stuff in the database raw ourselves. But once we built that first builder screen, we were able to then start dragging and dropping screens ourselves, which is why we were able to crank out so much stuff in such a short time.

This is a report builder app that your report builder people will be able to use, and their business objects are dashboards, reports, and query objects. Those are the things they're going to use. So that's what's going to show up in the menu when you're in the report builder application.

When you're in MINT, you're going to see purchase orders, invoices, all that type of stuff. Coming soon, you'll be able to take these reports and dashboards and include them in your solutions as well. So that's really neat stuff.

With that, I'm just going to do a little quick key takeaways for what we learned today, and then I'm going to hand it off to Steve.

The first thing to remember, we have a unified no-code environment regardless of whether you're talking about standard marketplace or company solutions. They all utilize that same no-code solution design environment, drag and drop, editing, this opinionated user interface.

That makes it simple, not only for your end users, but also for your solution developers. They don't have to worry about the pixels and where exactly how much padding you put around fonts. They don't have an option. Just drag the fields. That business logic is down there in the core of the business objects. They don't need to worry about those details.

Two, metadata-driven development framework. It's a no-code development design. It starts with the metadata at that lowest level, the business objects, their fields, their operations. They build up, that metadata builds up into the user interface layer, where we start defining new pages, search pages, view edit pages, reports and dashboards.

Then, finally, the entire solution and its metadata is built by putting together menus, menu

items, roles, and permissions. That's the third item there.

Fourth, configuration and subtyping and workflow extensions. That subtyping is what allows you to tailor your objects. MINT creates subtypes of purchase orders and invoices from that base BTO object. Then the workflows for all these subtypes can be extended with substates, and this is what allows customers to adapt those solutions to your business needs when necessary.

Then the fifth thing up here is page structure and consistency, that opinionated UI. Again, every page follows a very clear structure, starts with the menus, then page titles, sections, fields, and the operations and buttons, and where they're all placed. It's consistent, and that reduces all the design decisions and things that people have to worry about. Ultimately, it ends up benefiting your end users because there's very minimal training here.

I mean, everything we talked about is very intuitive. If you work with POs on a day to day basis and you see a button that says purchase order, you click on it, you see some search results, you click on one of the rows, you see some view edit pages, you're going to know what to do. It's very simple, very intuitive.