Enterprise-Level Technology Architecture Models

As we turn our attention to *user story 3*, we see that all of its activities involve technology, and our primary audience is the **Chief Technology Officer** (**CTO**). This might lead us to believe that we need some highly technical models. But that is not true!

While the subject matter is technology, as with other models, the level of detail in the technology layer model should be tailored to the audience. Unlike previous user stories, this one requires that we have an enterprise view of technology. That means we need to examine all of the existing technologies at *ABC Trading*. We need to understand where they are used and for what purposes.

We will first look at the behavioral elements of the **ArchiMate® 3.1** specification and how they are used. We will then turn our attention to building the catalogs we will need to identify duplicate technologies. The sections covered are as follows:

- Using technology behavioral elements
- *ABC Trading*'s technology background
- Building the technology components catalog
- Modeling technology services
- Reporting our findings

Using technology behavioral elements

This section covers the **behavioral elements** within the **ArchiMate technology layer**. The following diagram illustrates the elements covered:

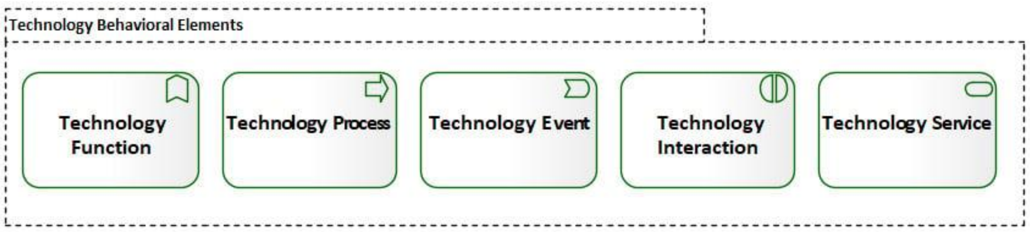


Figure – The technology behavioral elements

The primary difference between behavioral elements of the technology layer and those of the application layer is the scope of the elements to which they are applied. A technology layer behavioral element works only within the technology layer domain. Except for the Technology Service element, all behavioral elements in the technology layer represent behavior that is internal to technology structural elements.

However, unlike the application layer, we don't often build or modify technology layer components. Most of our technology is purchased from technology vendors. You may ask yourself why we need to describe how a technology component performs a function when that behavior is hidden from us. The short answer is that, unless your organization is in the business of developing technology to sell, you do not need to do this often.

However, there is one important exception to this fact. Sometimes, describing *how* something is done provides clarity to *what* is being done. It's important to note that behavioral elements are abstracted from the actual implementation. After all, we really don't know how most technologies are implemented, but we can describe how a function is performed in abstract terms. Let's look at an example.

At *ABC Trading*, we need to propose a new inventory management system. Our current system maintains data in several different flat-file formats. Our current application is doing a great deal of extra work just to piece together an existing inventory item. Adding new items is slow and error-prone. Our proposed new system places all data in a **Relational Database Management System** (**RDBMS**) and will be able to add an inventory item by executing a single **Structured Query Language** (**SQL**) statement. We need to describe what behavior is happening behind that single SQL statement. This becomes especially helpful for stakeholders who are familiar with how the existing system currently works.

Understanding the responsibilities of a relational database, such as key field lookups, index references, index maintenance, cluster selection, transaction management, and failover scenarios, can go a long way toward justifying our proposed project. We need to consider our audience. Our CFO must approve all such projects, but they are not a technology person. The benefits of an RDBMS may not be as obvious to them as it is to developers at *ABC Trading*. Even for technical folks, you may need to describe the specific behavior you are counting on from the single SQL statement.

The technology layer behavioral element that you will likely use most often is the Technology Service element. That's because this is the only element that exposes the internal behavior of technology components. Let's take a look at this element first.

Using the Technology Service element

According to ArchiMate®, *"A technology service represents an explicitly defined exposed technology behavior. A technology service exposes the functionality of a node to its environment. This functionality is accessed through one or more technology interfaces. It may require, use, and produce artifacts. A technology service should be meaningful from the point of view of the environment; it should provide a unit of behavior that is, in itself, useful to its users, such as application components and nodes"*.

The following diagram serves as our focused metamodel for the **Technology Service** element:

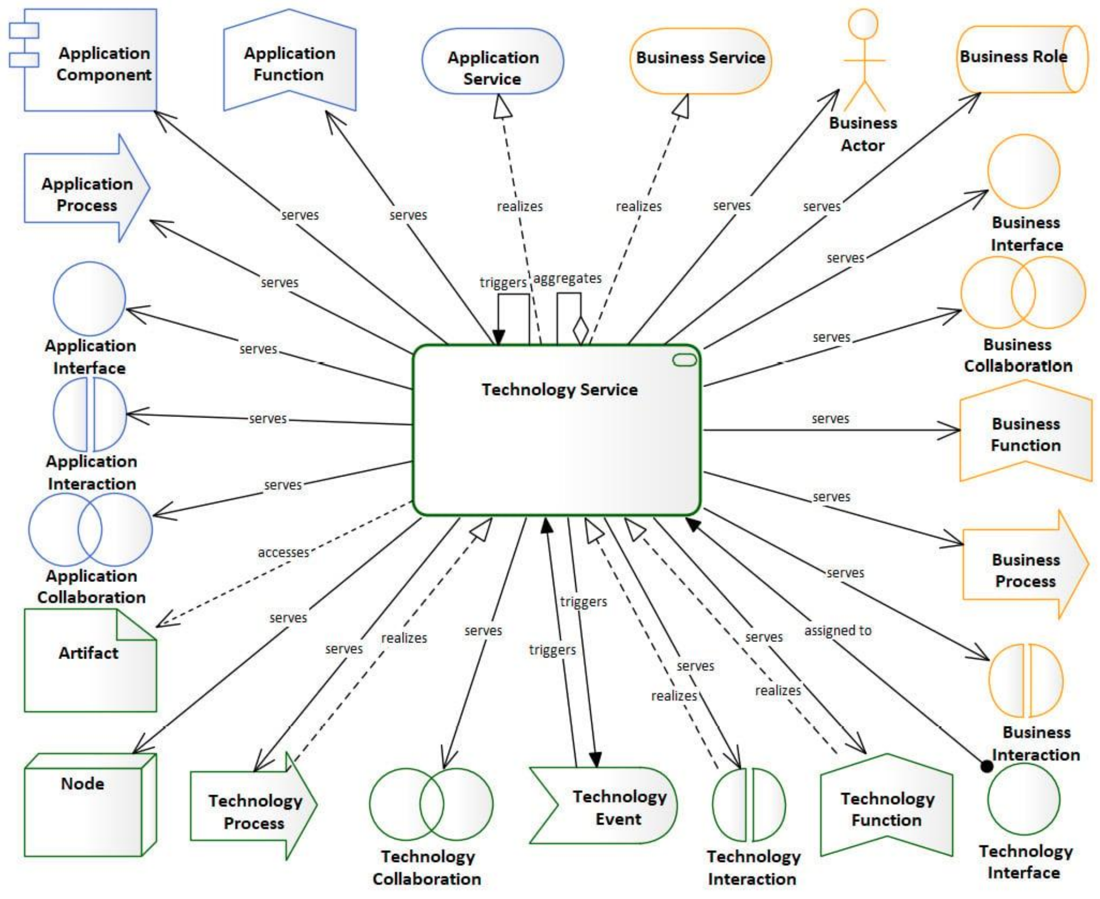


Figure – The Technology Service-focused metamodel

As you can see, a technology service can serve just about every other element in ArchiMate®. Technology services can also help to realize an application service and a business service. Examples of technology services include reading data, storing data, connecting to a resource, sending messages, receiving messages, and much more.

Probably the next most common technology layer behavioral element we will use is the technology function. Let's take a look at it next.

Using the Technology Function element

According to ArchiMate®, *"[a] technology function represents a collection of technology behavior that can be performed by a node"*.

The following focused metamodel shows the possible relationships of the **Technology Function** element:

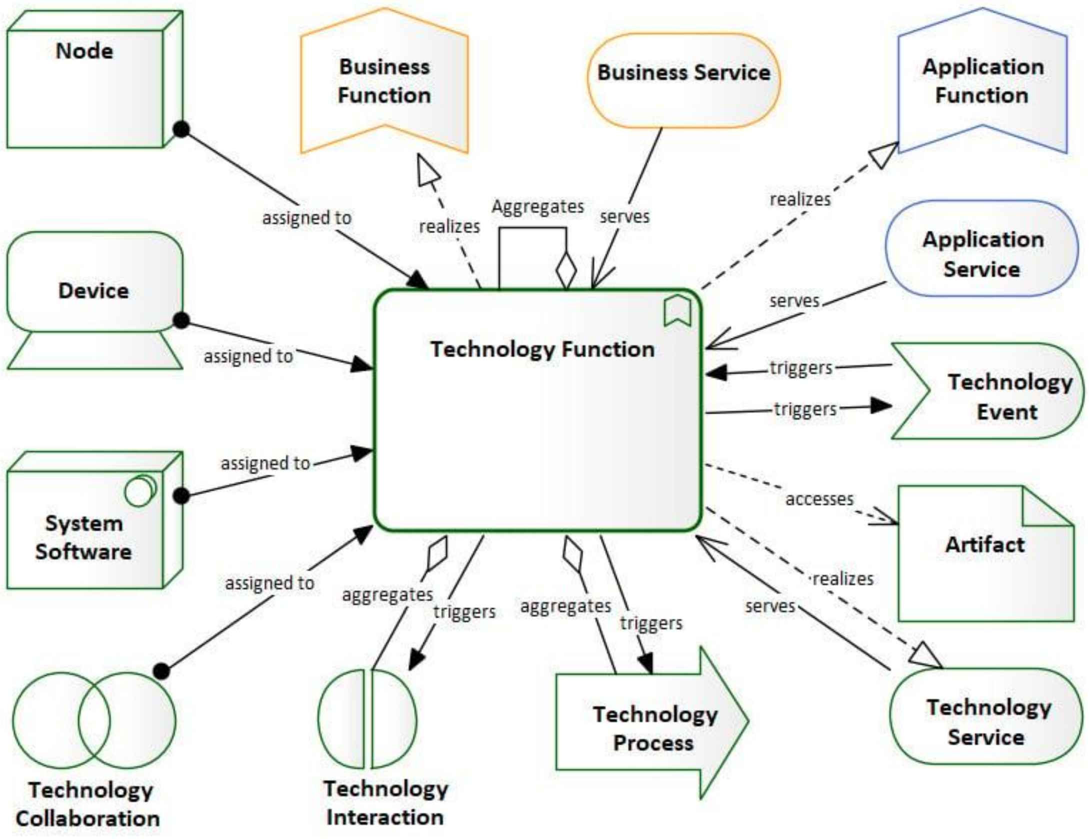


Figure – The Technology Function-focused metamodel

The technology function is internal to a node. It can be exposed only through a technology service. In the following diagram, we have added Technology Function elements to describe the roles of the network firewall and the **Router** elements of the LAN:

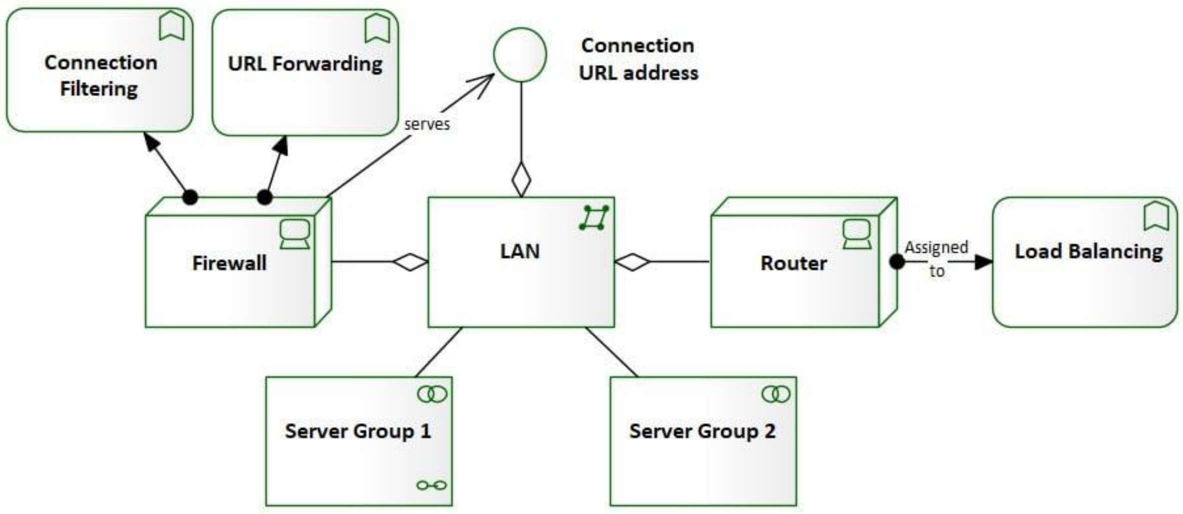


Figure – The firewall and router behaviors

From the preceding diagram, we see that the network firewall is assigned to provide **Connection Filtering** and **URL Forwarding**. The network router provides **Load Balancing**. These functions are internal to the **Firewall** and **Router** elements. In these examples, we don't know how the firewall performs connection filtering or URL forwarding, nor in what order these functions are performed. Those are implementation details that we don't need and are not privy to.

In ArchiMate®, most of the relationships that cross the level boundaries follow a distinct pattern. Upper levels tend to depend on lower levels and lower levels tend to serve upper levels, as depicted in the following diagram:

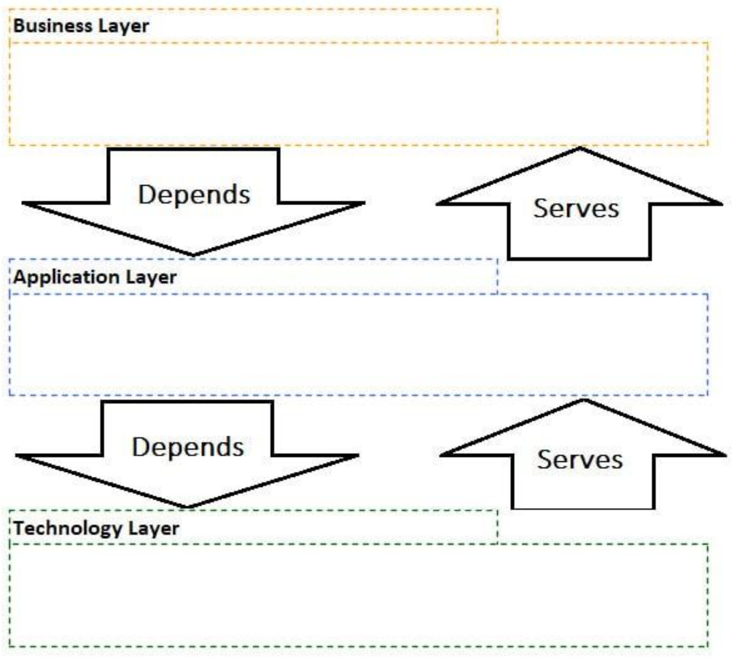


Figure – ArchiMate® inter-layer relationships

Next, we will look at another behavioral element – the technology process.

Using the technology process

"A technology process represents a sequence of technology behaviors that achieves a specific result".

The following focused metamodel shows the possible relationships of the **Technology Process** element:

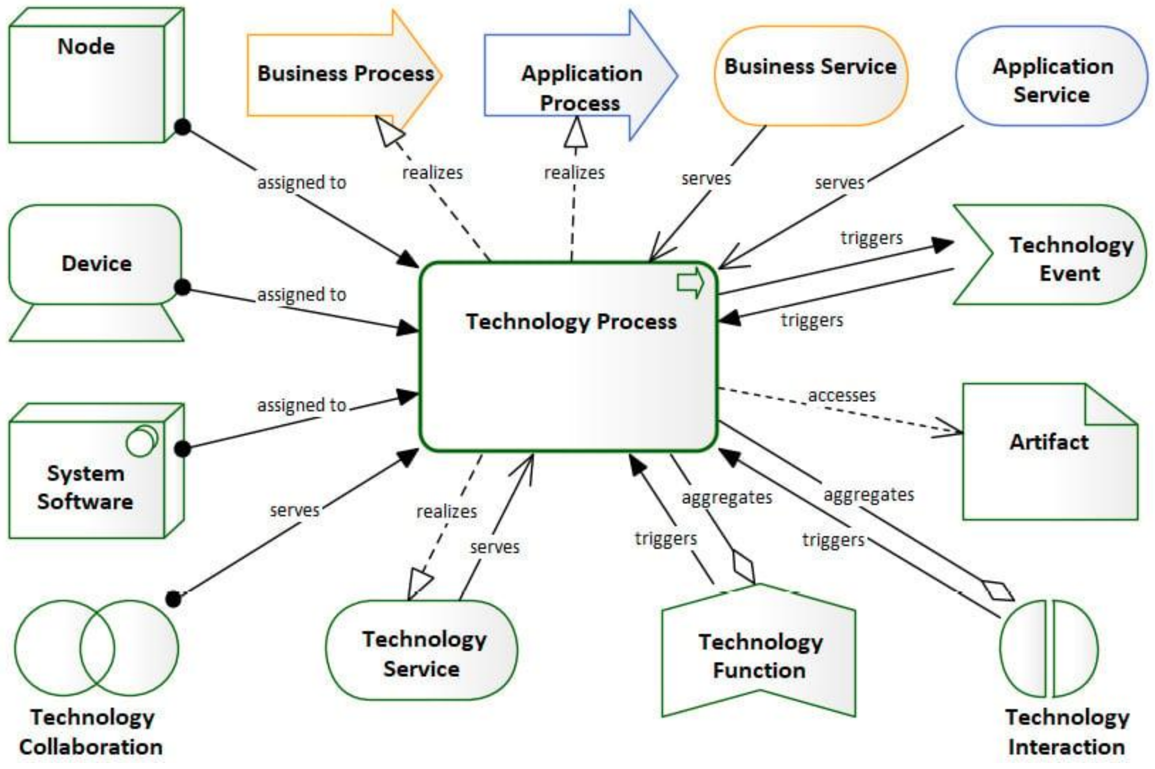


Figure – The Technology Process-focused metamodel

The key distinction between the technology process and other behavioral elements is the notion of a *sequence* of behaviors. The technology process is a behavior that requires the execution of multiple functions or steps to be completed, often in some order. The following diagram illustrates one usage of the process element:

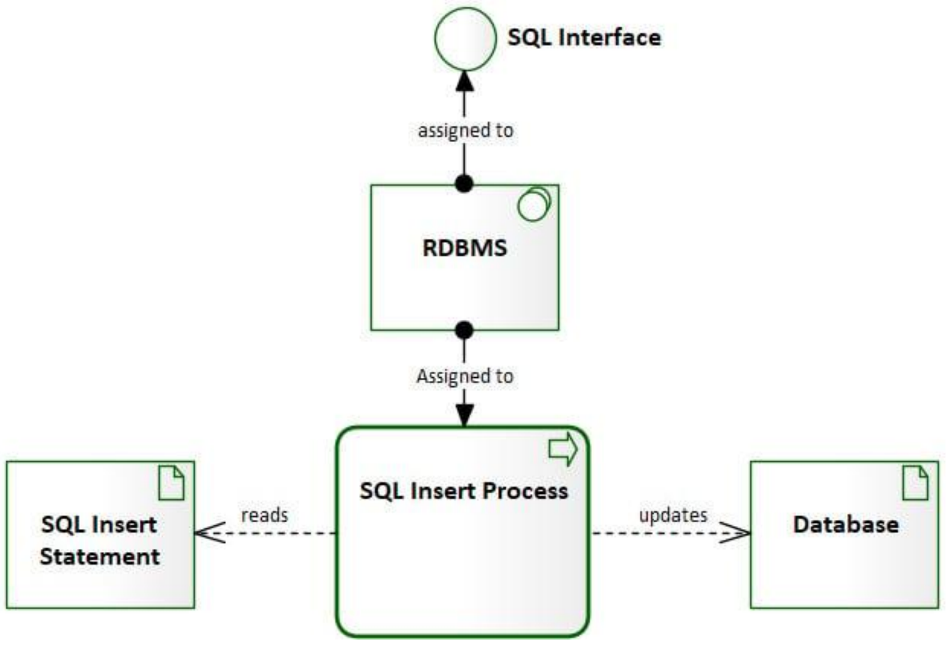
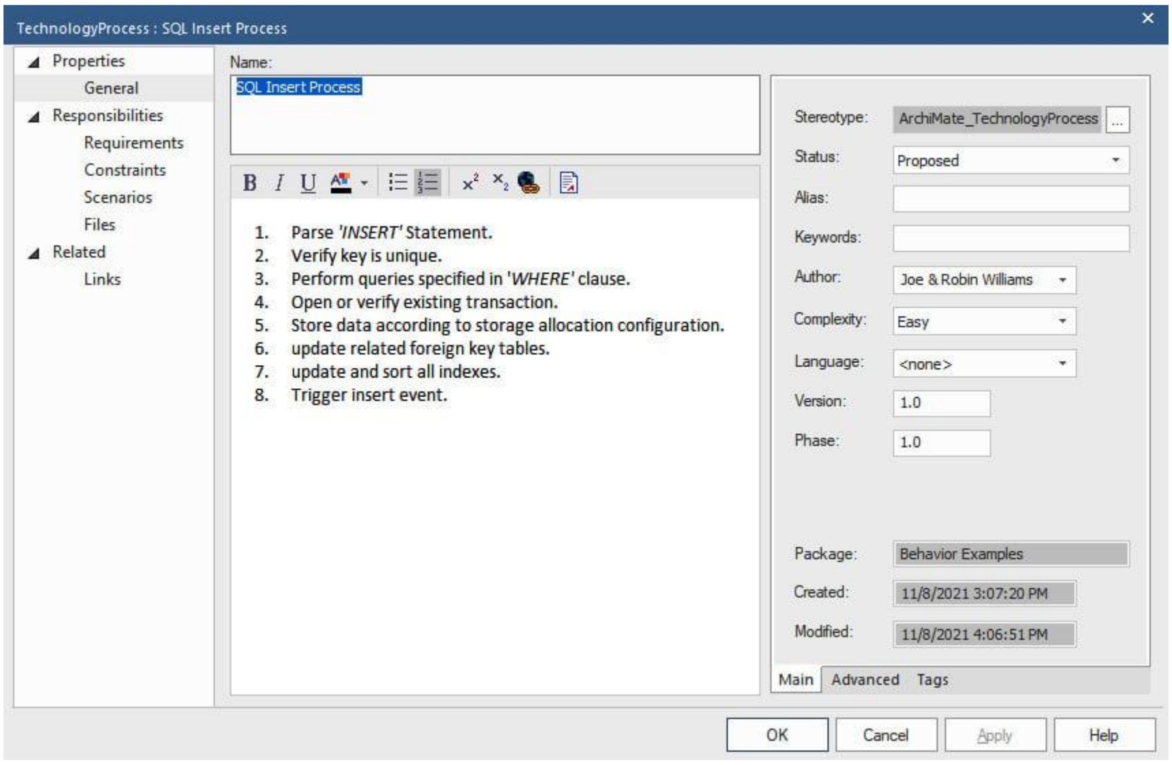


Figure – A technical process element example

In this example, **SQL Insert Process** is performed internally by the RDBMS software node. The process reads a SQL `INSERT` statement and updates a database accordingly. Relational databases perform several operations when processing such statements. If it's important to document those steps, Sparx provides an easy way to do so. The following diagram shows the **Properties** dialog for the **SQL Insert Process** element:



Name: SQL Insert Process

1. Parse *'INSERT'* Statement.
2. Verify key is unique.
3. Perform queries specified in '*WHERE*' clause.
4. Open or verify existing transaction.
5. Store data according to storage allocation configuration.
6. update related foreign key tables.
7. update and sort all indexes.
8. Trigger insert event.

Stereotype: ArchiMate_TechnologyProcess
Status: Proposed
Alias:
Keywords:
Author: Joe & Robin Williams
Complexity: Easy
Language: <none>
Version: 1.0
Phase: 1.0
Package: Behavior Examples
Created: 11/8/2021 3:07:20 PM
Modified: 11/8/2021 4:06:51 PM

Figure – The SQL Insert Process element Properties dialog

In the **Notes** section of the **Properties** dialog, we have described a series of actions performed by the database software when performing the insert process. The actions are described in very general terms. Any given database engine may perform these or other steps. Also, they are not necessarily performed in this precise order. Most likely, you will not be describing the actions of a process at all. Knowing that it is a process in itself is often detailed enough.

The next technology behavioral element we will look at is used to describe a more involved relationship – it's the technology interaction element.

Using the technology interaction element

"A technology interaction represents a unit of collective technology behavior performed by (a collaboration of) two or more nodes".

A **technology interaction** occurs between two or more node elements to achieve a specific function or behavior, as shown here:

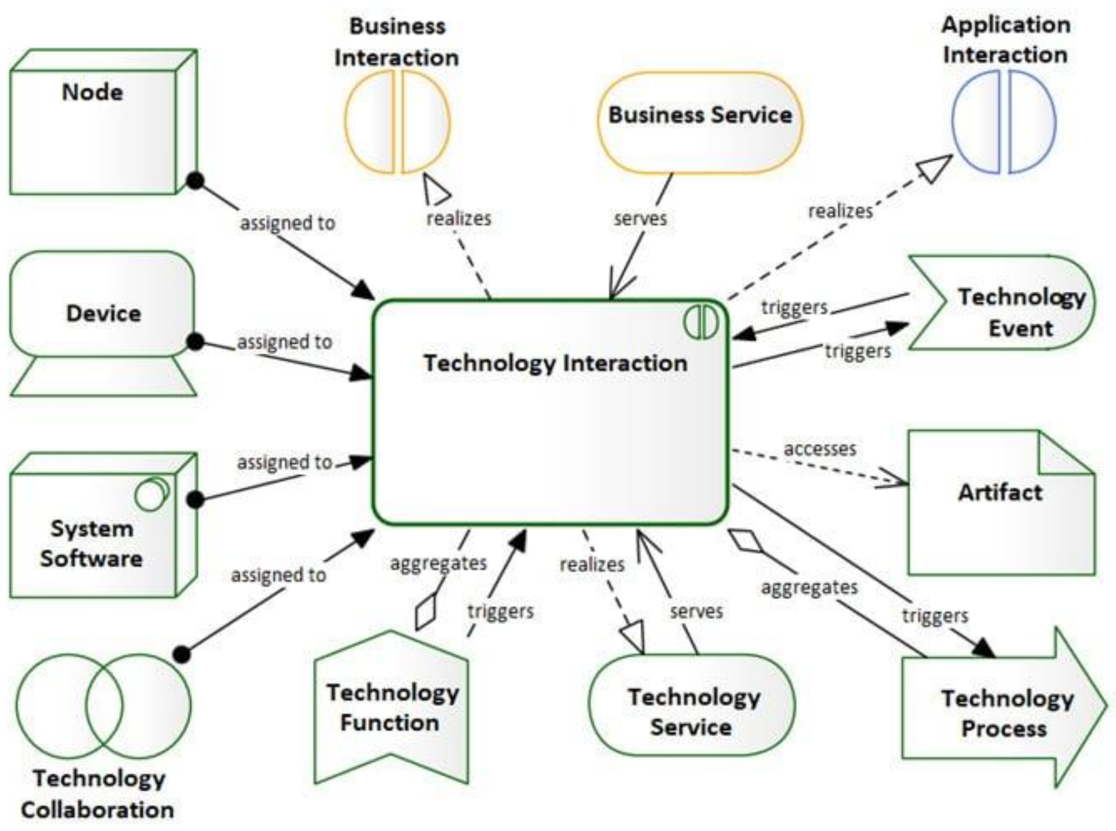


Figure – The Technology Interaction-focused metamodel

Technology Interaction elements are logically paired with Technology Collaboration elements. In this sense, the technology collaboration identifies the nodes that participate in the interaction, whereas the technology interaction describes the function that the collective nodes are to perform. For example, a database engine node and an application server node may collaborate to manage database update transactions, as shown in the following figure:

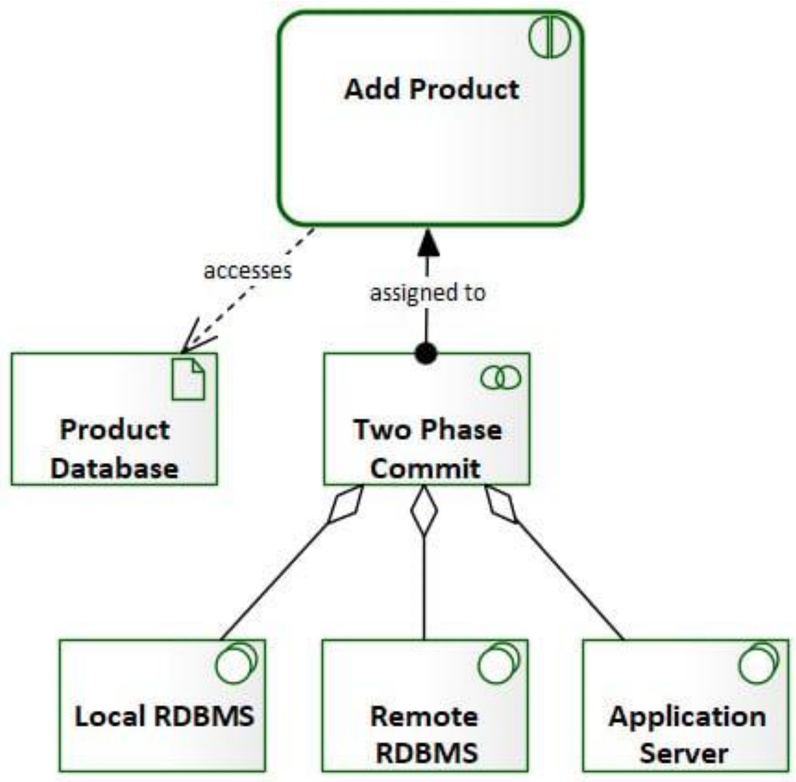


Figure – A technology interaction example

In the preceding example, the **Add Product** interaction accesses the product database for an update and performs a two-phase commit, which includes a local database engine, a remote database engine, and an application server. The **Add Product** interaction may also serve to realize the add product service (not depicted).

The final technology behavioral element we will look at is the Technology Event element.

Using the Technology Event element

A **technology event** represents a change of state within the technology environment. In short, something has happened in the environment that is important for some reason. Behavioral elements may trigger an event or may be invoked by an event. Structural elements such as nodes may be assigned to an event, implying that they will respond to the event in some way. In event processing systems, we say the nodes may *subscribe* to an event.

"Technology functions and other technology behavior may be triggered or interrupted by a technology event. Also, technology functions may raise events that trigger other infrastructure behavior. Unlike processes, functions, and interactions, an event is

instantaneous: it does not have duration. Events may originate from the environment of the organization, but also internal events may occur generated by, for example, other devices within the organization".

Technology events can have the relationships depicted in the following focused metamodel:

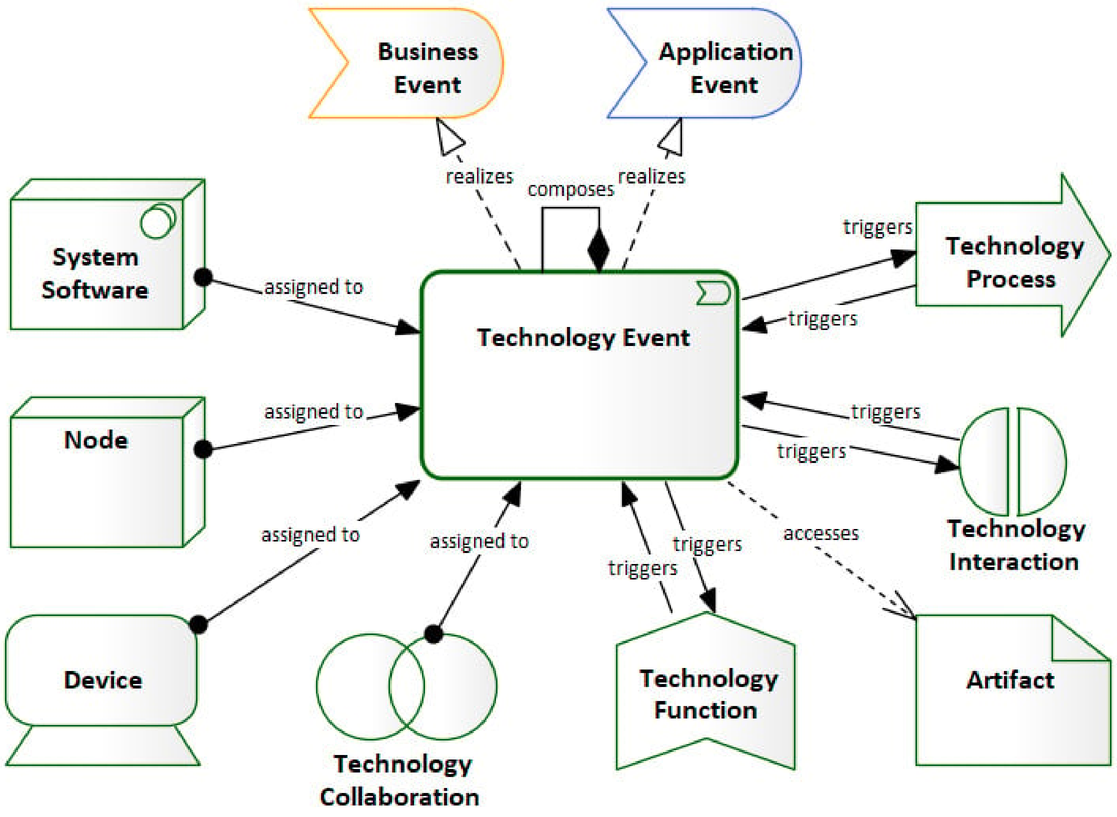


Figure – The Technology Event-focused metamodel

Examples of events include opening or closing a file, connecting to a resource, starting or completing a process, starting or stopping a device, or updating a database table.

In the following example, when a new product is added, it triggers a **Propagate Changes** interaction to propagate the change across all database instances:

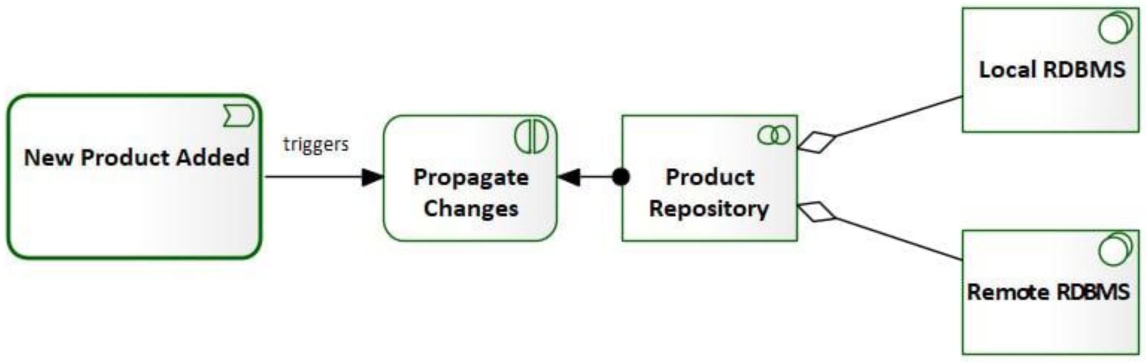


Figure – A technology event example

The elements discussed in this section represent the core of the technology behavioral elements. Before we move on to building our technology catalogs, let's take a look at why we're doing this in the first place and gain some context around *ABC Trading*'s technology background.

ABC Trading's technology background

In our scenario, the CTO is new to the organization. They are under pressure to control technology costs. Like many organizations, *ABC Trading* has dozens of applications that have been developed or purchased at various points in its history. Along with each application, new technology components have been implemented to support those applications and various other projects.

The CTO suspects that, along the way, some applications and technology components that are no longer needed have been left in place. It's also possible that the use of some applications or components can be eliminated by making relatively small changes. Unlike the previous user stories, this one requires us to analyze information across a broad spectrum of the enterprise rather than looking at the details of a single application or technology. The problem is that the CTO does not have this information.

But why not? It's not as if systems are implemented haphazardly. *ABC Trading* has a robust change management process in place. This process requires that new applications come with detailed design information in the form of Microsoft Word documents. Changes to the data center and network environment go through the same process. The information is generated at implementation time. Since this process has been in place for several years, you would think that all of the necessary information is available.

The simple answer is that the information does, indeed, exist. The applications development team keeps scores of design documents from years of changes. Database administrators maintain data models in conceptual, logical, and physical forms. The operations department maintains a spreadsheet of all servers and physical equipment in place in the data centers. The network folks are constantly updating their network infrastructure diagram. So, what's the problem?

One problem is that this information is rarely reused. When the application team designs a change to an application, they make their own reference to the database or network environments, rather than using the information generated by the operations or network folks. It's just easier that way.

When the operations team renames or replaces a server, they never go back and change views that rely on them. Each department maintains its independent perspective of the enterprise. Rarely are any of the views completely accurate. From the enterprise perspective, this information is unreliable. It's simply not trusted. This is especially frustrating, given the amount of time and work devoted to creating and maintaining documentation. It's our job as **Enterprise Architects** to integrate this information into an enterprise repository from an enterprise perspective. We need enterprise-level information. To keep it accurate and reliable, we need to build trust in our enterprise information.

Confidence in enterprise information will not be realized quickly but will come gradually. We will look at ways to ensure the integration of information across the enterprise. The following section describes one convenient way to establish the core of the technology layer, the **technology components catalog**.

Building the technology components catalog

To address the CTO's concerns about redundant and unused technology, we need information in our repository for all equipment at *ABC Trading*. From the repository, we will identify the services provided by each technology. Where two or more technologies provide the same service, we have a potential duplicate. We can then analyze the usage of each technology to determine whether we might be able to eliminate one of them. First, let's look at what the technology components catalog is.

Defining the technology components catalog

Simply put, the technology components catalog is a list of all hardware, system software, and equipment in use at *ABC Trading*. Examples of hardware include servers, storage devices, networking devices, and so on. Examples of equipment include server racks, coolers, humidifiers, sensors, and much more.

System software includes such things as operating systems, database software, storage or backup systems, application containers, transaction monitors, and security software. For our purposes, the technology components catalog must reside in our repository in Sparx. This is where we will be performing our analysis and reporting our results.

Collecting the information

One possible way to build this catalog in Sparx is to fully document each application in the enterprise, including all technology components used by the application. Doing this will give us a complete list of each technology component in use. While it would be wonderful to have such information in Sparx, there are at least two problems with this approach:

- Firstly, reporting from an application perspective will not reveal technologies not being used by any application.
- Secondly, documenting each application would take an enormous amount of time and effort.

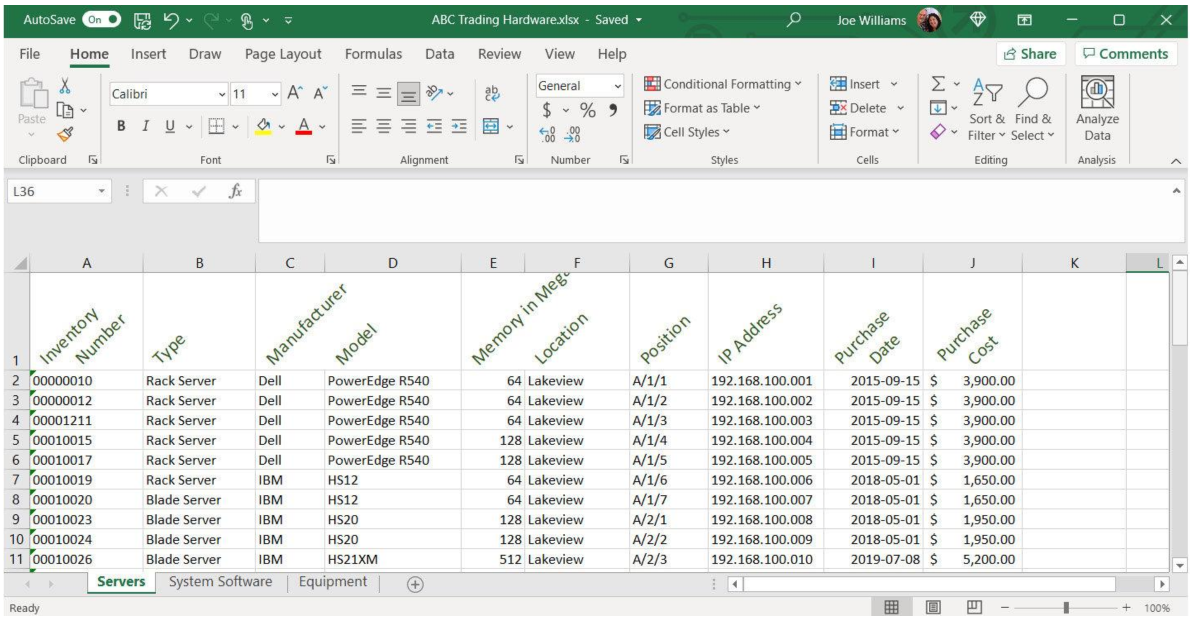
Certainly, the CTO would like this information before they retire. So, we will need another, more practical, approach.

It is common for organizations to maintain information in spreadsheets. We need to reuse this information where practical. Fortunately for us, the data center manager maintains an inventory of equipment housed in the data center. Each of the warehouse managers has a small list of equipment in use at their respective locations. After meeting with these folks, we have most of the information we need, but it's in the form of a set of Microsoft Excel spreadsheets.

Spreadsheets are great for crunching numbers, and knowledge of Microsoft Excel is ubiquitous in the industry. However, when knowledge of complex relationships needs to be communicated, spreadsheets often fall short of the task. This is where a modeling tool such as Sparx Systems Enterprise Architect shines.

So, we need to do some formatting to the sheets and save them in the `.csv` format. We will then build an import specification in Sparx and, finally, import the files into Sparx.

By far, the data center equipment list is the biggest. The warehouse equipment list was small enough that we simply added the warehouse equipment to the data center list. The first worksheet contains a list of the server hardware used at the data center, which looks like the following screenshot:


AutoSave On
ABC Trading Hardware.xlsx - Saved
Joe Williams
File Home Insert Draw Page Layout Formulas Data Review View Help
Share
Comments
Conditional Formatting
Format as Table
Cell Styles
Insert
Delete
Format
Sort & Filter
Find & Select
Analyze Data
Clipboard Font Alignment Number Styles Cells Editing Analysis
L36
Servers System Software Equipment
Ready
100%

	A	B	C	D	E	F	G	H	I	J
1	Inventory Number	Type	Manufacturer	Model	Memory in Meg	Location	Position	IP Address	Purchase Date	Purchase Cost
2	00000010	Rack Server	Dell	PowerEdge R540	64	Lakeview	A/1/1	192.168.100.001	2015-09-15	$ 3,900.00
3	00000012	Rack Server	Dell	PowerEdge R540	64	Lakeview	A/1/2	192.168.100.002	2015-09-15	$ 3,900.00
4	00001211	Rack Server	Dell	PowerEdge R540	64	Lakeview	A/1/3	192.168.100.003	2015-09-15	$ 3,900.00
5	00010015	Rack Server	Dell	PowerEdge R540	128	Lakeview	A/1/4	192.168.100.004	2015-09-15	$ 3,900.00
6	00010017	Rack Server	Dell	PowerEdge R540	128	Lakeview	A/1/5	192.168.100.005	2015-09-15	$ 3,900.00
7	00010019	Rack Server	IBM	HS12	64	Lakeview	A/1/6	192.168.100.006	2018-05-01	$ 1,650.00
8	00010020	Blade Server	IBM	HS12	64	Lakeview	A/1/7	192.168.100.007	2018-05-01	$ 1,650.00
9	00010023	Blade Server	IBM	HS20	128	Lakeview	A/2/1	192.168.100.008	2018-05-01	$ 1,950.00
10	00010024	Blade Server	IBM	HS20	128	Lakeview	A/2/2	192.168.100.009	2018-05-01	$ 1,950.00
11	00010026	Blade Server	IBM	HS21XM	512	Lakeview	A/2/3	192.168.100.010	2019-07-08	$ 5,200.00

Figure – The original hardware inventory Excel spreadsheet

Important Note

If you're following along with this exercise, you'll need to create a backup of the worksheet before you start.

We need to modify this list to prepare it to be imported into Sparx. Columns in the spreadsheet need to align with fields in Sparx. The following subsections detail this process.

Formatting the CSV file

First, we need to split any merged cells before attempting the import process because merged cells can confuse Sparx. The element fields we need to create are as follows:

- Name
- Notes
- Type
- Stereotype
- Tag-value field – `Location`
- Tag-value field – `PurchaseDate`
- Tag-value field – `PurchaseCost`

Tag value fields are user-defined fields that can be added to any element type without being constrained by the field data type. You can add as many tag-value fields as needed, and you need to provide a name and a value pair for each.

Formatting the element name

The first model element field value we need to create is the **element name**. The worksheet fields we will use to create this value include the following:

- `Manufacturer`
- `Type`
- `Inventory number`

Using the inventory number will make the element name unique. Sparx does not require the `name` field to have a unique value, but for readability, we would like the name to be unique – for example, the first server on the list will have the name `Dell Rack Server 00000010`. To create this new `name` column, follow these steps:

1. Insert a new column to the left of column **A**. Give the new column **A** the heading `Name`.
2. In the second row of the new **Name** column, enter the `=$C2&" "&$B2&" "&$A2` formula. This instructs the spreadsheet to set the value of the current cell to the value in cell `C2` and then append a space and the contents of cell `B2`,

followed by a space and the content of cell `A2`. The result should be the name **Dell Rack Server 00000010** displayed in row **2**, column **A**.

3. Now, simply copy this new formula and paste it into all the remaining rows of column **A**. This column should now show the names of each of the servers in the data center.
4. While Excel now displays the name, the actual value of each cell is still a formula. Replace the formula in each cell with the actual text resulting from the formula. Copy the entire column **A** and paste it back in the same location using the **paste-as-text** option.

This is a recommended way to construct a meaningful name that is both readable and understandable by any user. You may decide to follow a different naming approach according to your environment and stakeholders' requirements. The next element field we need to create is the `Notes` field.

Formatting the Notes field

The `Notes` field is used to add descriptive text to any element type. The steps for formatting the `Notes` field are similar to the steps used for formatting the `Name` field. You need to replace the formula with one that better fits your needs:

1. Create a new column to the right of column **A**. Give it the name `Notes`.
2. In the second row, enter the formula to concatenate the **Model**, **Memory**, **Position**, and **IP Address** fields. We include labels for the position and IP address – `=$F2&" "$G2&"mb Position"&$H2&" IP address: "&$I2`.
3. Copy the formula and paste it into the remaining rows.
4. Convert the formulae to text by repeating *step 3* of the previous section.
5. Remove the original columns by repeating *step 4* of the previous section.

You are free to add any text in the `Notes` field, but you need to make sure that your notes are sufficiently descriptive yet not too long. We suggest limiting your notes to fewer than three lines, but it is totally up to you. Next, we need the element `Type` field.

Identifying the element types

In Sparx, every element is identified by a type and an optional stereotype. The way that Sparx works is by extending the basic UML types, such as **class** and **component,** with stereotypes. If you are modeling with ArchiMate® notations, Sparx requires you to provide the stereotypes as well. Providing these two fields is required by Sparx for importing spreadsheets as

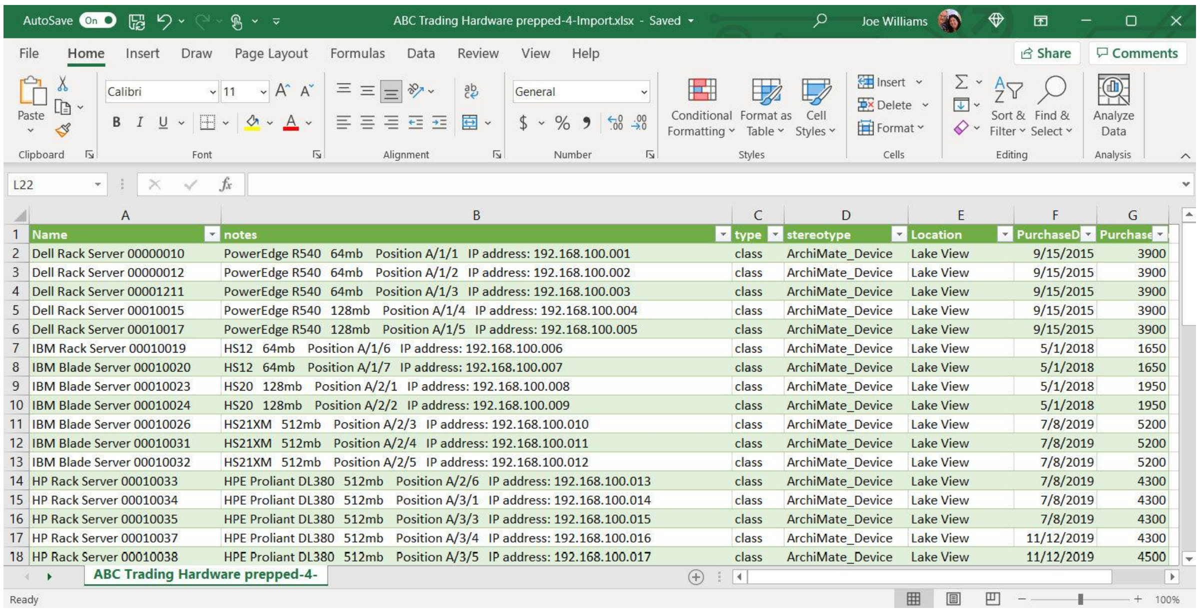
ArchiMate® elements. In the sample spreadsheet that we are using, all the elements have a basic element `class` type and the `ArchiMate_Device` stereotype. We add these two columns as follows:

1. Create a new column to the right of the **notes** column. Give it the name `Type` and enter the `class` value in all rows.
2. In the same manner as the **type** column, add the **stereotype** column. Insert another column to the right of the **type** column and add the `ArchiMate_Device` value to each row.

This tells Sparx the type and the stereotype to assign for each element that will be imported. The last step is to format the `tag` fields.

Formatting the tag fields

Leave the `Location` and `PurchaseDate` fields as they are, but format the `PurchaseCost` field as `Number` with no commas to mark the *thousands* position. The import specification will indicate that these fields go into tag-value fields. The final spreadsheet should look like this:



	A	B	C	D	E	F	G
1	Name	notes	type	stereotype	Location	PurchaseD	Purchase
2	Dell Rack Server 00000010	PowerEdge R540 64mb Position A/1/1 IP address: 192.168.100.001	class	ArchiMate_Device	Lake View	9/15/2015	3900
3	Dell Rack Server 00000012	PowerEdge R540 64mb Position A/1/2 IP address: 192.168.100.002	class	ArchiMate_Device	Lake View	9/15/2015	3900
4	Dell Rack Server 00001211	PowerEdge R540 64mb Position A/1/3 IP address: 192.168.100.003	class	ArchiMate_Device	Lake View	9/15/2015	3900
5	Dell Rack Server 00010015	PowerEdge R540 128mb Position A/1/4 IP address: 192.168.100.004	class	ArchiMate_Device	Lake View	9/15/2015	3900
6	Dell Rack Server 00010017	PowerEdge R540 128mb Position A/1/5 IP address: 192.168.100.005	class	ArchiMate_Device	Lake View	9/15/2015	3900
7	IBM Rack Server 00010019	HS12 64mb Position A/1/6 IP address: 192.168.100.006	class	ArchiMate_Device	Lake View	5/1/2018	1650
8	IBM Blade Server 00010020	HS12 64mb Position A/1/7 IP address: 192.168.100.007	class	ArchiMate_Device	Lake View	5/1/2018	1650
9	IBM Blade Server 00010023	HS20 128mb Position A/2/1 IP address: 192.168.100.008	class	ArchiMate_Device	Lake View	5/1/2018	1950
10	IBM Blade Server 00010024	HS20 128mb Position A/2/2 IP address: 192.168.100.009	class	ArchiMate_Device	Lake View	5/1/2018	1950
11	IBM Blade Server 00010026	HS21XM 512mb Position A/2/3 IP address: 192.168.100.010	class	ArchiMate_Device	Lake View	7/8/2019	5200
12	IBM Blade Server 00010031	HS21XM 512mb Position A/2/4 IP address: 192.168.100.011	class	ArchiMate_Device	Lake View	7/8/2019	5200
13	IBM Blade Server 00010032	HS21XM 512mb Position A/2/5 IP address: 192.168.100.012	class	ArchiMate_Device	Lake View	7/8/2019	5200
14	HP Rack Server 00010033	HPE Proliant DL380 512mb Position A/2/6 IP address: 192.168.100.013	class	ArchiMate_Device	Lake View	7/8/2019	4300
15	HP Rack Server 00010034	HPE Proliant DL380 512mb Position A/3/1 IP address: 192.168.100.014	class	ArchiMate_Device	Lake View	7/8/2019	4300
16	HP Rack Server 00010035	HPE Proliant DL380 512mb Position A/3/3 IP address: 192.168.100.015	class	ArchiMate_Device	Lake View	7/8/2019	4300
17	HP Rack Server 00010037	HPE Proliant DL380 512mb Position A/3/4 IP address: 192.168.100.016	class	ArchiMate_Device	Lake View	11/12/2019	4300
18	HP Rack Server 00010038	HPE Proliant DL380 512mb Position A/3/5 IP address: 192.168.100.017	class	ArchiMate_Device	Lake View	11/12/2019	4500

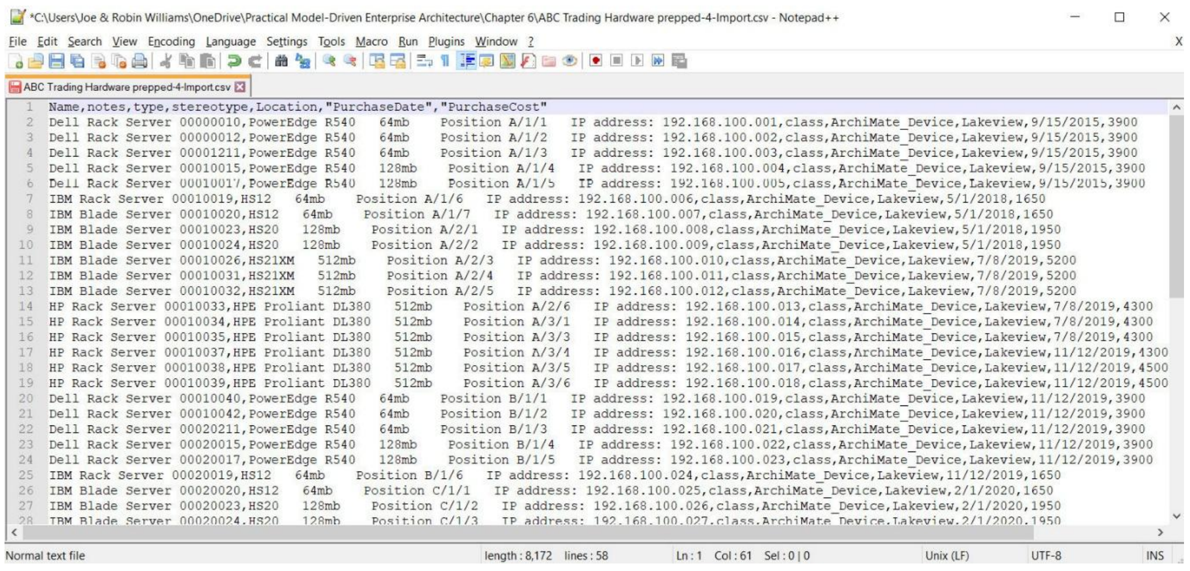
Figure – The reformatted spreadsheet

Finally, save the spreadsheet in the `.csv` format – **File** > **Export** > **Change File Type** > **CSV (Comma Delimited)**. You may get a warning about losing content when saving to this format; that's okay. Give it the name `ABC Trading Hardware prepped-4-Import.csv`.

Caution

None of the fields can contain embedded commas. This is important, as it will cause the import to fail.

The `.csv` files are plain text files. You can view them in any text editor. Our prepped file now looks like the following:


*C:\Users\Joe & Robin Williams\OneDrive\Practical Model-Driven Enterprise Architecture\Chapter 6\ABC Trading Hardware prepped-4-Import.csv - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
ABC Trading Hardware prepped-4-Import.csv

```
Name,notes,type,stereotype,Location,"PurchaseDate","PurchaseCost"
Dell Rack Server 00000010,PowerEdge R540   64mb    Position A/1/1   IP address: 192.168.100.001,class,ArchiMate_Device,Lakeview,9/15/2015,3900
Dell Rack Server 00000012,PowerEdge R540   64mb    Position A/1/2   IP address: 192.168.100.002,class,ArchiMate_Device,Lakeview,9/15/2015,3900
Dell Rack Server 00001211,PowerEdge R540   64mb    Position A/1/3   IP address: 192.168.100.003,class,ArchiMate_Device,Lakeview,9/15/2015,3900
Dell Rack Server 00010015,PowerEdge R540   128mb    Position A/1/4   IP address: 192.168.100.004,class,ArchiMate_Device,Lakeview,9/15/2015,3900
Dell Rack Server 00010017,PowerEdge R540   128mb    Position A/1/5   IP address: 192.168.100.005,class,ArchiMate_Device,Lakeview,9/15/2015,3900
IBM Rack Server 00010019,HS12   64mb    Position A/1/6   IP address: 192.168.100.006,class,ArchiMate_Device,Lakeview,5/1/2018,1650
IBM Blade Server 00010020,HS12   64mb    Position A/1/7   IP address: 192.168.100.007,class,ArchiMate_Device,Lakeview,5/1/2018,1650
IBM Blade Server 00010023,HS20   128mb    Position A/2/1   IP address: 192.168.100.008,class,ArchiMate_Device,Lakeview,5/1/2018,1950
IBM Blade Server 00010024,HS20   128mb    Position A/2/2   IP address: 192.168.100.009,class,ArchiMate_Device,Lakeview,5/1/2018,1950
IBM Blade Server 00010026,HS21XM   512mb    Position A/2/3   IP address: 192.168.100.010,class,ArchiMate_Device,Lakeview,7/8/2019,5200
IBM Blade Server 00010031,HS21XM   512mb    Position A/2/4   IP address: 192.168.100.011,class,ArchiMate_Device,Lakeview,7/8/2019,5200
IBM Blade Server 00010032,HS21XM   512mb    Position A/2/5   IP address: 192.168.100.012,class,ArchiMate_Device,Lakeview,7/8/2019,5200
HP Rack Server 00010033,HPE Proliant DL380   512mb    Position A/2/6   IP address: 192.168.100.013,class,ArchiMate_Device,Lakeview,7/8/2019,4300
HP Rack Server 00010034,HPE Proliant DL380   512mb    Position A/3/1   IP address: 192.168.100.014,class,ArchiMate_Device,Lakeview,7/8/2019,4300
HP Rack Server 00010035,HPE Proliant DL380   512mb    Position A/3/3   IP address: 192.168.100.015,class,ArchiMate_Device,Lakeview,7/8/2019,4300
HP Rack Server 00010037,HPE Proliant DL380   512mb    Position A/3/4   IP address: 192.168.100.016,class,ArchiMate_Device,Lakeview,11/12/2019,4300
HP Rack Server 00010038,HPE Proliant DL380   512mb    Position A/3/5   IP address: 192.168.100.017,class,ArchiMate_Device,Lakeview,11/12/2019,4500
HP Rack Server 00010039,HPE Proliant DL380   512mb    Position A/3/6   IP address: 192.168.100.018,class,ArchiMate_Device,Lakeview,11/12/2019,4500
Dell Rack Server 00010040,PowerEdge R540   64mb    Position B/1/1   IP address: 192.168.100.019,class,ArchiMate_Device,Lakeview,11/12/2019,3900
Dell Rack Server 00010042,PowerEdge R540   64mb    Position B/1/2   IP address: 192.168.100.020,class,ArchiMate_Device,Lakeview,11/12/2019,3900
Dell Rack Server 00020211,PowerEdge R540   64mb    Position B/1/3   IP address: 192.168.100.021,class,ArchiMate_Device,Lakeview,11/12/2019,3900
Dell Rack Server 00020015,PowerEdge R540   128mb    Position B/1/4   IP address: 192.168.100.022,class,ArchiMate_Device,Lakeview,11/12/2019,3900
Dell Rack Server 00020017,PowerEdge R540   128mb    Position B/1/5   IP address: 192.168.100.023,class,ArchiMate_Device,Lakeview,11/12/2019,3900
IBM Rack Server 00020019,HS12   64mb    Position B/1/6   IP address: 192.168.100.024,class,ArchiMate_Device,Lakeview,11/12/2019,1650
IBM Blade Server 00020020,HS12   64mb    Position C/1/1   IP address: 192.168.100.025,class,ArchiMate_Device,Lakeview,2/1/2020,1650
IBM Blade Server 00020023,HS20   128mb    Position C/1/2   IP address: 192.168.100.026,class,ArchiMate_Device,Lakeview,2/1/2020,1950
IBM Blade Server 00020024,HS20   128mb    Position C/1/3   IP address: 192.168.100.027,class,ArchiMate_Device,Lakeview,2/1/2020,1950
```

Normal text file   length : 8,172   lines : 58   Ln : 1   Col : 61   Sel : 0 | 0   Unix (LF)   UTF-8   INS

Figure – The CSV file contents

The file is ready to be imported into Sparx, so let's see how to do it.

Importing the CSV file into Sparx

We are going to import the `.csv` file we just created into Sparx. This process consists of three steps:

1. Creating a package in the repository to hold new elements
2. Creating an import specification for the new `.csv` file
3. Executing the import using the new specification

Let's start by creating the package.

Creating the package for the enterprise view

In our enterprise repository, we already have a package for the *metamodels*, and we have other packages for specific applications or specific elements of the enterprise.

However, the content that we are about to import now does not belong to a specific element, and we can consider it global or enterprise-level content. Therefore, we will create a separate package to contain all the enterprise content, including the spreadsheet content that we are about to import, by following these steps:

1. Start Sparx and open your repository.
2. In the **Project** browser, right-click on the model's root node, **Architecture Content**.
3. Add a new view and name it `Enterprise View`.
4. Right-click on **Enterprise View** and add a new folder called `Servers`:

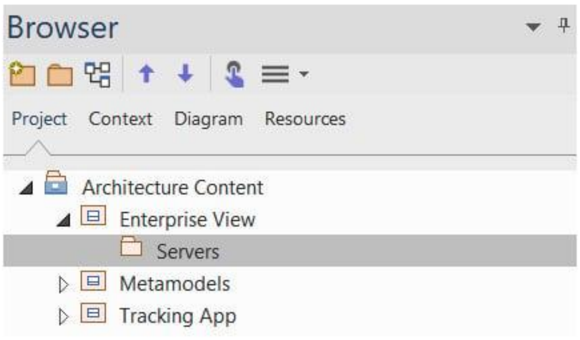


Figure – The Servers package inside the Enterprise View package

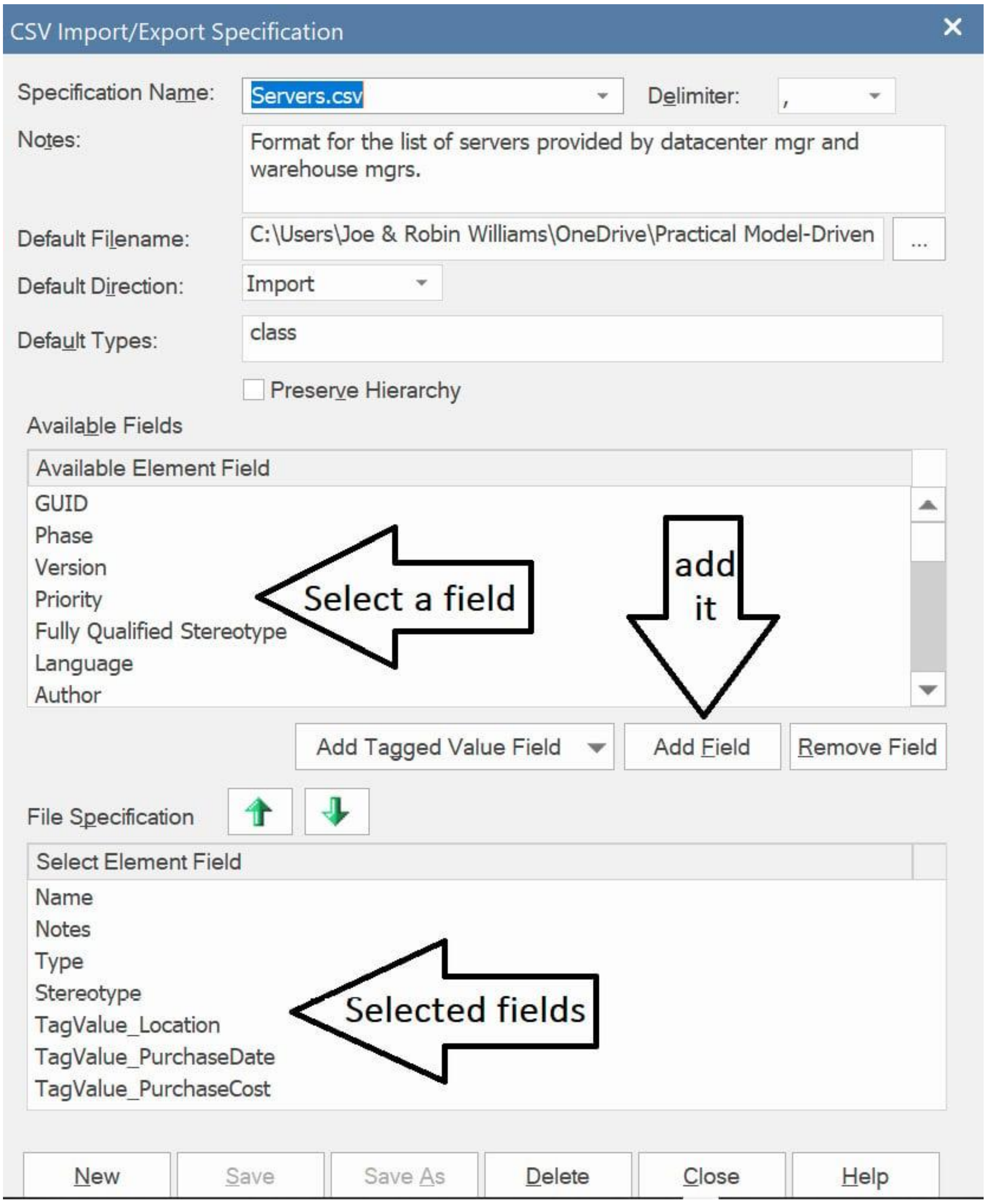
That's good enough for now. That will hold the contents of the `.csv` file.

Creating an import/export specification

This process creates a specification that tells Sparx what to expect while importing our data. Go to **Publish** > **CSV** > **CSV Exchange Specification**. A dialog box will appear, asking for more information to complete the import. Then, enter the following information in the dialog box shown:

1. Provide a name for the specification, such as `Servers.csv`.
2. Select the comma from the **Delimiter** dropdown.
3. The **Notes** are optional and free-form.
4. **Default Filename** is the `.csv` file created earlier. When you specify this value, you will receive a warning that the file already exists. Reply **Yes** to this dialog. Don't worry – nothing will be over-written! The warning is for exporting only; we select **Import** as **Default Direction**.
5. For each of the fields that we created in the `.csv` file, select the field name from the **Available Element Field** section and press the **Add Field** button. The fields must be in the order that they appear in the `.csv` file. The following screenshot

shows the **CSV Import/Export Specification** dialog and the process of adding fields:


CSV Import/Export Specification

Specification Name: Servers.csv
Delimiter: ,
Notes: Format for the list of servers provided by datacenter mgr and warehouse mgrs.
Default Filename: C:\Users\Joe & Robin Williams\OneDrive\Practical Model-Driven
Default Direction: Import
Default Types: class
Preserve Hierarchy

Available Fields

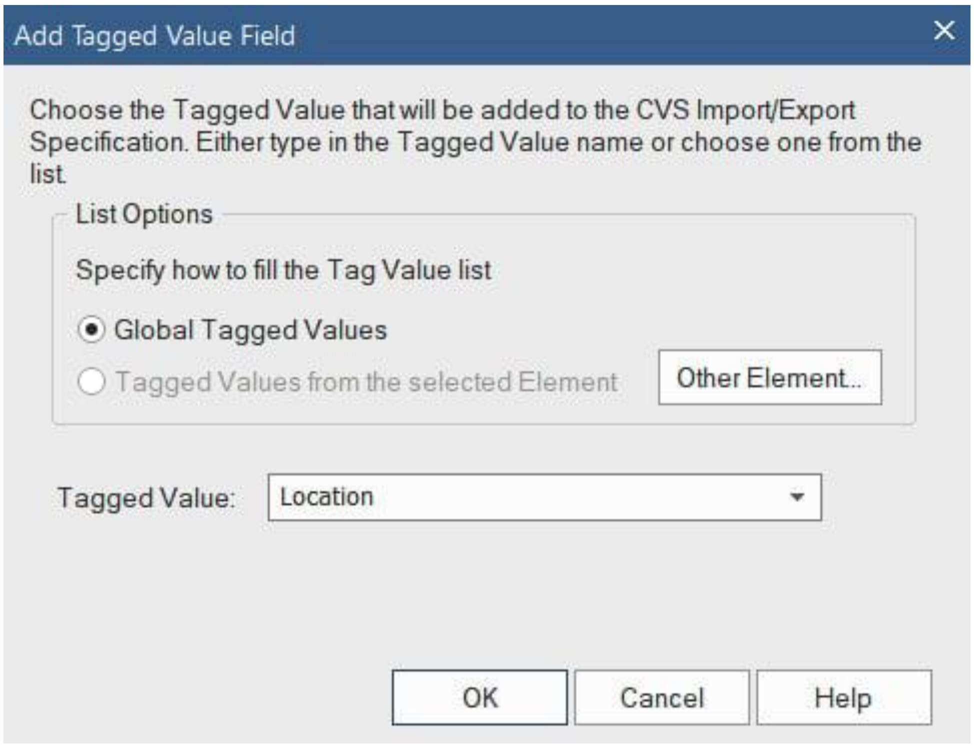
Available Element Field
GUID
Phase
Version
Priority
Fully Qualified Stereotype
Language
Author

File Specification

Select Element Field
Name
Notes
Type
Stereotype
TagValue_Location
TagValue_PurchaseDate
TagValue_PurchaseCost

Figure – The CSV Import/Export Specification dialog

6. For the last three fields, **Location**, **PurchaseDate**, and **PurchaseCost**, press the **Add Tagged Value Field** button. Then, select **Value**. Enter the field name and press **OK**, as shown in the following screenshot:


Add Tagged Value Field

Choose the Tagged Value that will be added to the CVS Import/Export Specification. Either type in the Tagged Value name or choose one from the list.

List Options

Specify how to fill the Tag Value list

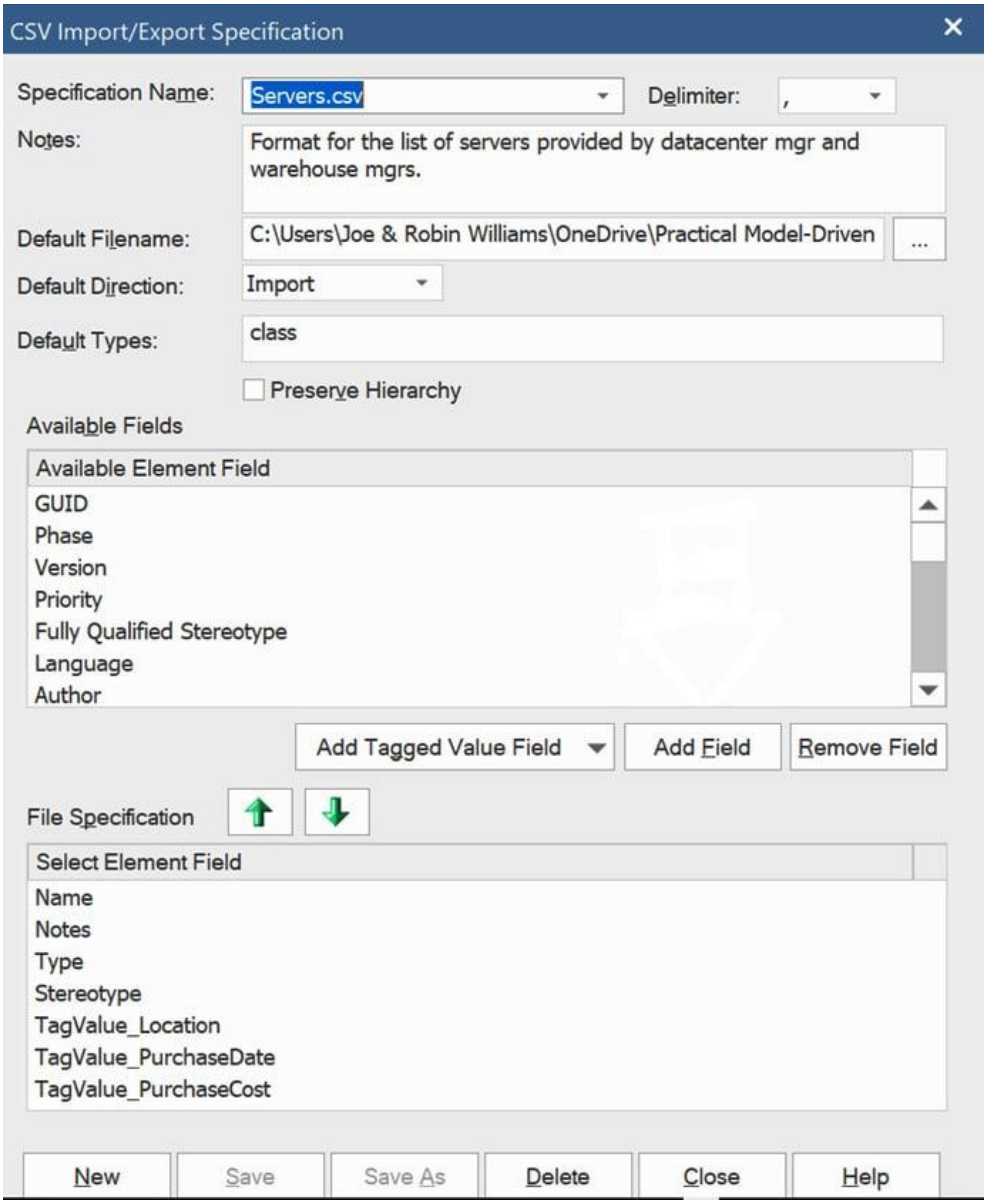
(●) Global Tagged Values

( ) Tagged Values from the selected Element

Tagged Value: Location

Figure – Add Tagged Value Field

7. When all fields have been specified, the **CSV Import/Export Specification** dialog should look like the following:


CSV Import/Export Specification

Specification Name: Servers.csv
Delimiter: ,

Notes: Format for the list of servers provided by datacenter mgr and warehouse mgrs.

Default Filename: C:\Users\Joe & Robin Williams\OneDrive\Practical Model-Driven

Default Direction: Import

Default Types: class

Preserve Hierarchy

Available Fields

Available Element Field
GUID
Phase
Version
Priority
Fully Qualified Stereotype
Language
Author

File Specification

Select Element Field
Name
Notes
Type
Stereotype
TagValue_Location
TagValue_PurchaseDate
TagValue_PurchaseCost

Figure – The completed CSV Import/Export Specification dialog

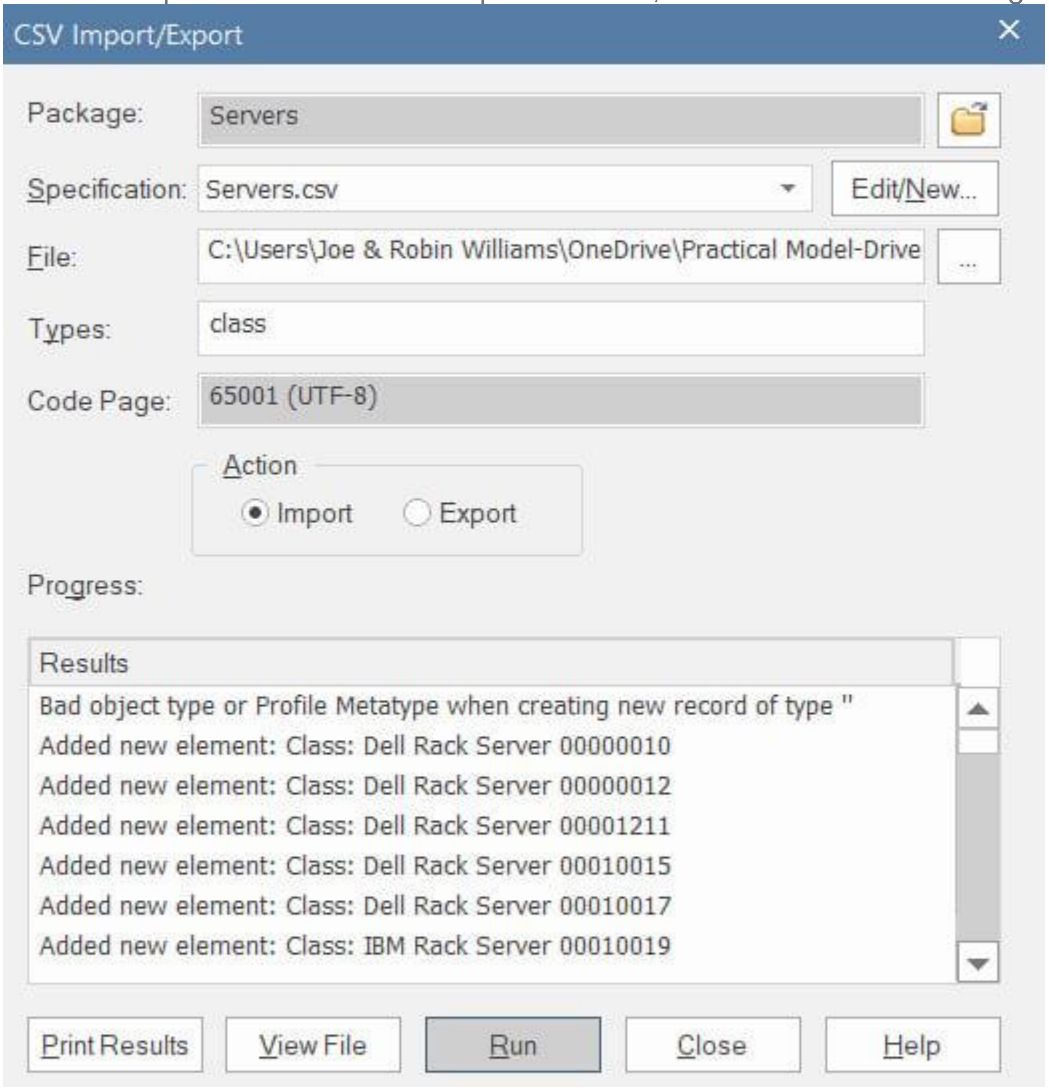
8. Press the **Save** button to save the specification data.

Now that we have created the specification, we can import our data.

Running the import function

This is also the easiest step:

1. In the Project browser, select the new package, **Servers**.
2. Go to **Publish** > **CSV** > **CSV Import/Export**.
3. From the dialog that displays, enter the name of the specification we just created, `Servers.csv`.
4. Make sure that the **File** field points to the `.csv` file we just created.
5. The remaining fields have default values, which we will accept.
6. Press the **Run** button. Sparx executes the `import` function and reports the status of each row processed from the import `.csv` file, as shown in the following figure:



Package: Servers
Specification: Servers.csv
File: C:\Users\Joe & Robin Williams\OneDrive\Practical Model-Drive
Types: class
Code Page: 65001 (UTF-8)
Action: [✓ Import] Export
Progress:

```
Results
Bad object type or Profile Metatype when creating new record of type ''
Added new element: Class: Dell Rack Server 00000010
Added new element: Class: Dell Rack Server 00000012
Added new element: Class: Dell Rack Server 00001211
Added new element: Class: Dell Rack Server 00010015
Added new element: Class: Dell Rack Server 00010017
Added new element: Class: IBM Rack Server 00010019
```

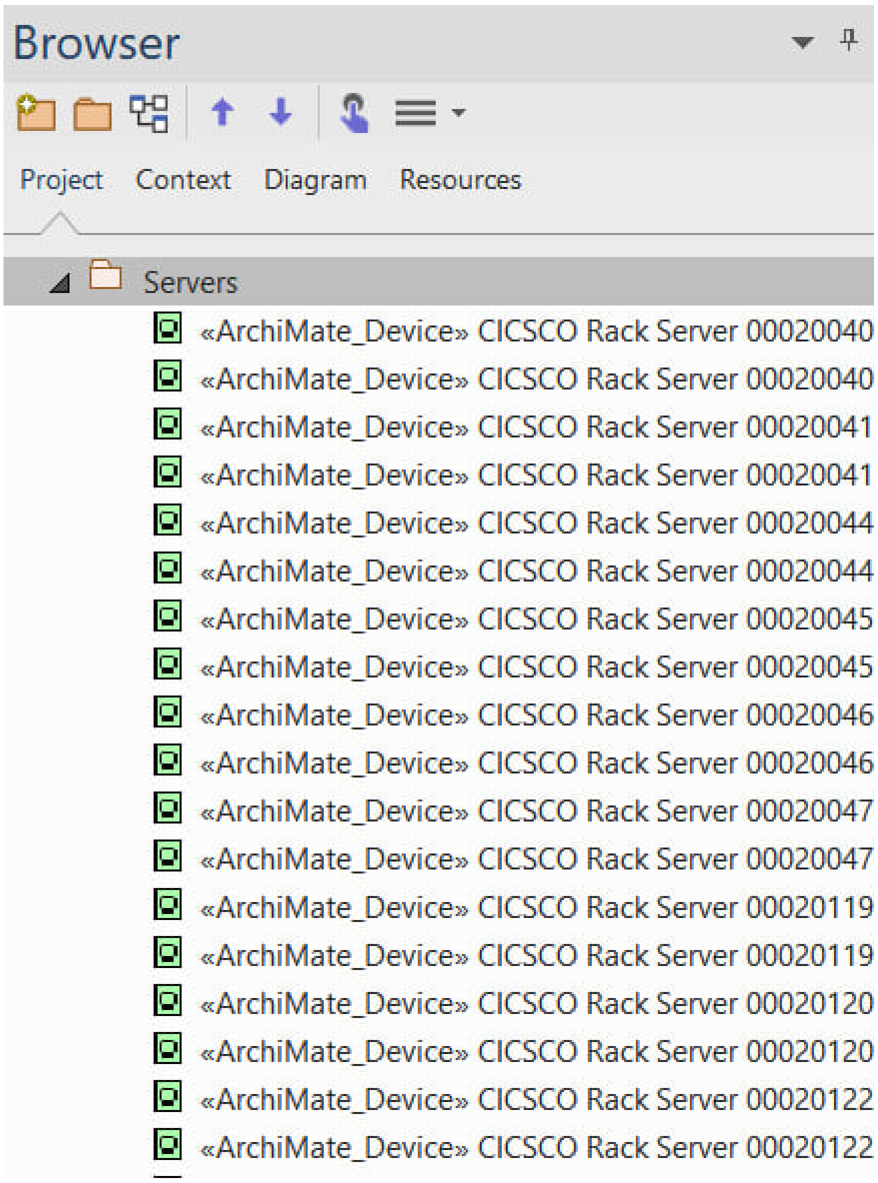
Figure – Successful import results

7. Note that the first record reported an error, **Bad object type or Profile Metatype when creating a record of type**. This is okay. The first record is the header row. Since there is no element type called `type`, an error is reported for that record. It is a small price to pay for keeping our field names straight, but if you do not like error messages, you are free to remove the header row in future imports.
8. Click the **Close** button to close the dialog.

Next, we will see how the imported elements look in Sparx and how to fine-tune them.

Reviewing the results

Now, let's look at what we have added to the repository. Expand the **Servers** package in the Project browser. You should see a list of elements, as shown here:

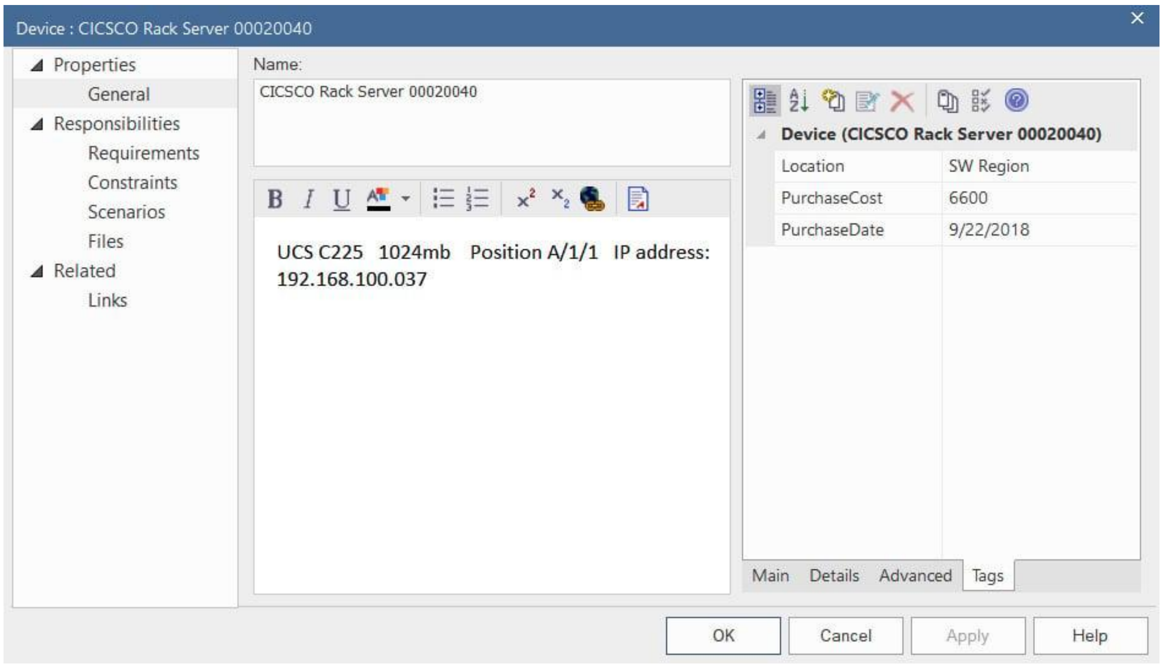


Servers

«ArchiMate_Device» CICSCO Rack Server 00020040
«ArchiMate_Device» CICSCO Rack Server 00020040
«ArchiMate_Device» CICSCO Rack Server 00020041
«ArchiMate_Device» CICSCO Rack Server 00020041
«ArchiMate_Device» CICSCO Rack Server 00020044
«ArchiMate_Device» CICSCO Rack Server 00020044
«ArchiMate_Device» CICSCO Rack Server 00020045
«ArchiMate_Device» CICSCO Rack Server 00020045
«ArchiMate_Device» CICSCO Rack Server 00020046
«ArchiMate_Device» CICSCO Rack Server 00020046
«ArchiMate_Device» CICSCO Rack Server 00020047
«ArchiMate_Device» CICSCO Rack Server 00020047
«ArchiMate_Device» CICSCO Rack Server 00020119
«ArchiMate_Device» CICSCO Rack Server 00020119
«ArchiMate_Device» CICSCO Rack Server 00020120
«ArchiMate_Device» CICSCO Rack Server 00020120
«ArchiMate_Device» CICSCO Rack Server 00020122
«ArchiMate_Device» CICSCO Rack Server 00020122

Figure – The ArchiMate® devices

Select one of the elements in the list. Right-click and select **Properties** from the context menu, and then select **Properties** again from the context sub-menu. The **Properties** dialog will appear. Select the **Tags** tab on the right side of the dialog. The results should look something like the following:


Device : CICSCO Rack Server 00020040

Name: CICSCO Rack Server 00020040

UCS C225 1024mb Position A/1/1 IP address: 192.168.100.037

Device (CICSCO Rack Server 00020040)

Location	SW Region
PurchaseCost	6600
PurchaseDate	9/22/2018

Figure – The device properties

Now that we have imported information regarding all the servers at *ABC Trading*, we need to follow the same steps with other model elements so that we can build the information around complex relationships and, later, report on it.

Equipment and system software

This section describes other technology component elements that we need to import. We will not go into detail about how to format and input this data, as the process is the same as what has already been described. Given the equipment list, we have created another `.csv` file import specification called `Equipment.csv` and loaded that information into another repository package called `Equipment`. We have done the same with `System Software`. The Sparx Project browser now looks like this:

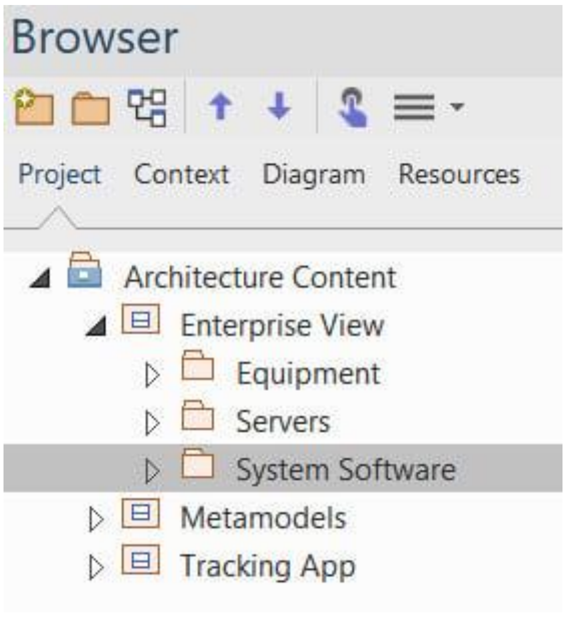
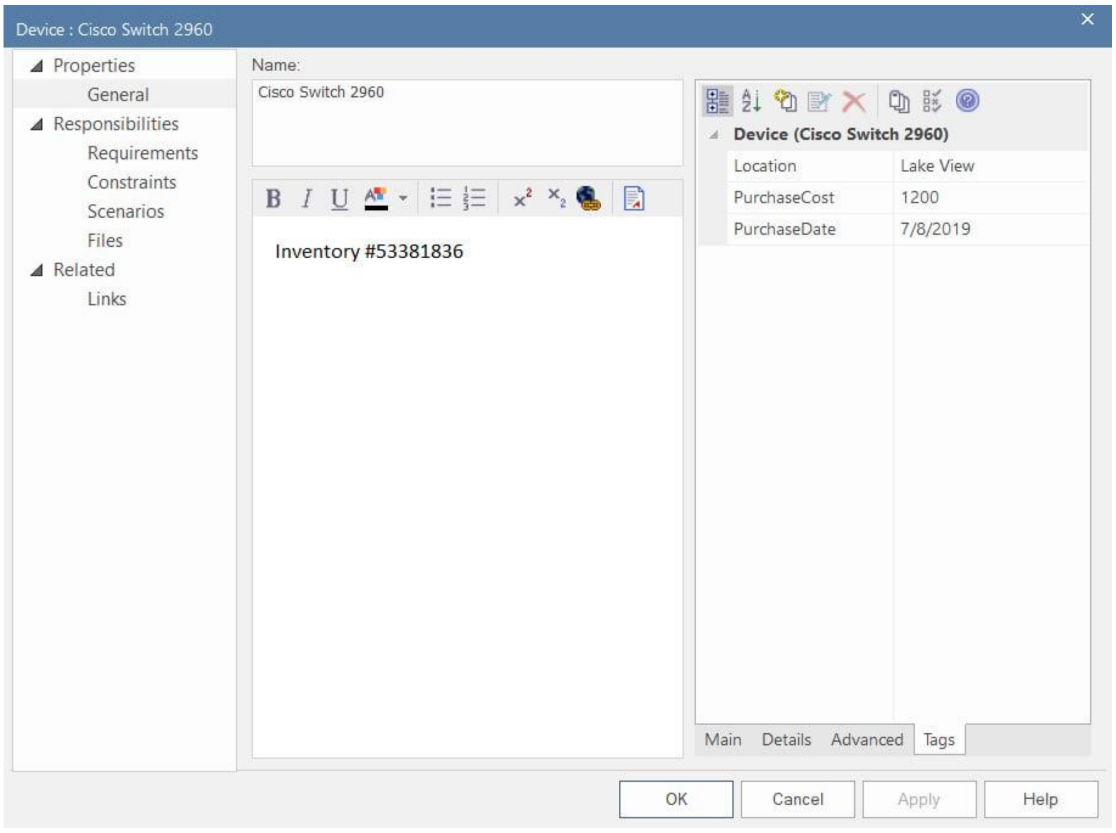


Figure – The tree Enterprise View catalogs

If you open the **Properties** dialog for any element in the **Equipment** or **System Software** catalogs, the Properties dialog will show information like this:



Device : Cisco Switch 2960

Name: Cisco Switch 2960

Inventory #53381836

Device (Cisco Switch 2960)

Location	Lake View
PurchaseCost	1200
PurchaseDate	7/8/2019

Figure – Equipment properties

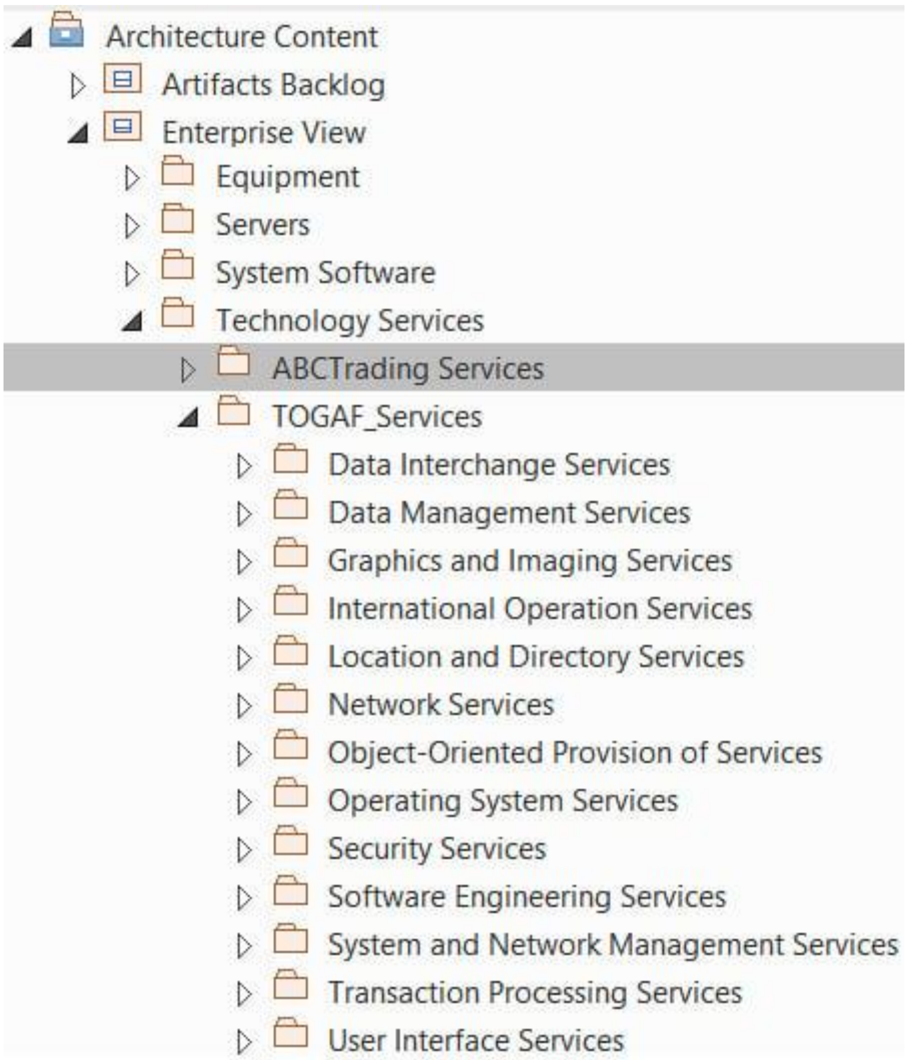
Now, let's see how we can use this information.

Modeling technology services

We will be exploring ways to identify potential duplicate resources within *ABC Trading*'s technology infrastructure. To do this, we will be referencing the devices and systems software elements that we imported in the last section. The process that we will use involves identifying and modeling the discrete services supplied by each of the technology elements in place.

Identifying existing services

Technology Service elements are used to describe functionality performed by the node that is made available to other elements in the environment. If you're new to the concept of technology-providing services, this process can seem a bit daunting. What are the services? What should they be called? What do they mean? Fortunately for us, the TOGAF® **Technical Reference Model** (**TRM**), https://pubs.opengroup.org/architecture/togaf8-doc/arch/chap19.html, provides an excellent starting point in the form of a taxonomy of technical services. To help you use that taxonomy, we have included over 100 service categories as Technology Service elements in our Sparx repository, located at https://github.com/PacktPublishing/Practical-Model-Driven-Enterprise-Architecture/blob/main/Chapter07/EA%20Repository.eapx. The location of these services in the repository is depicted in the following screenshot:



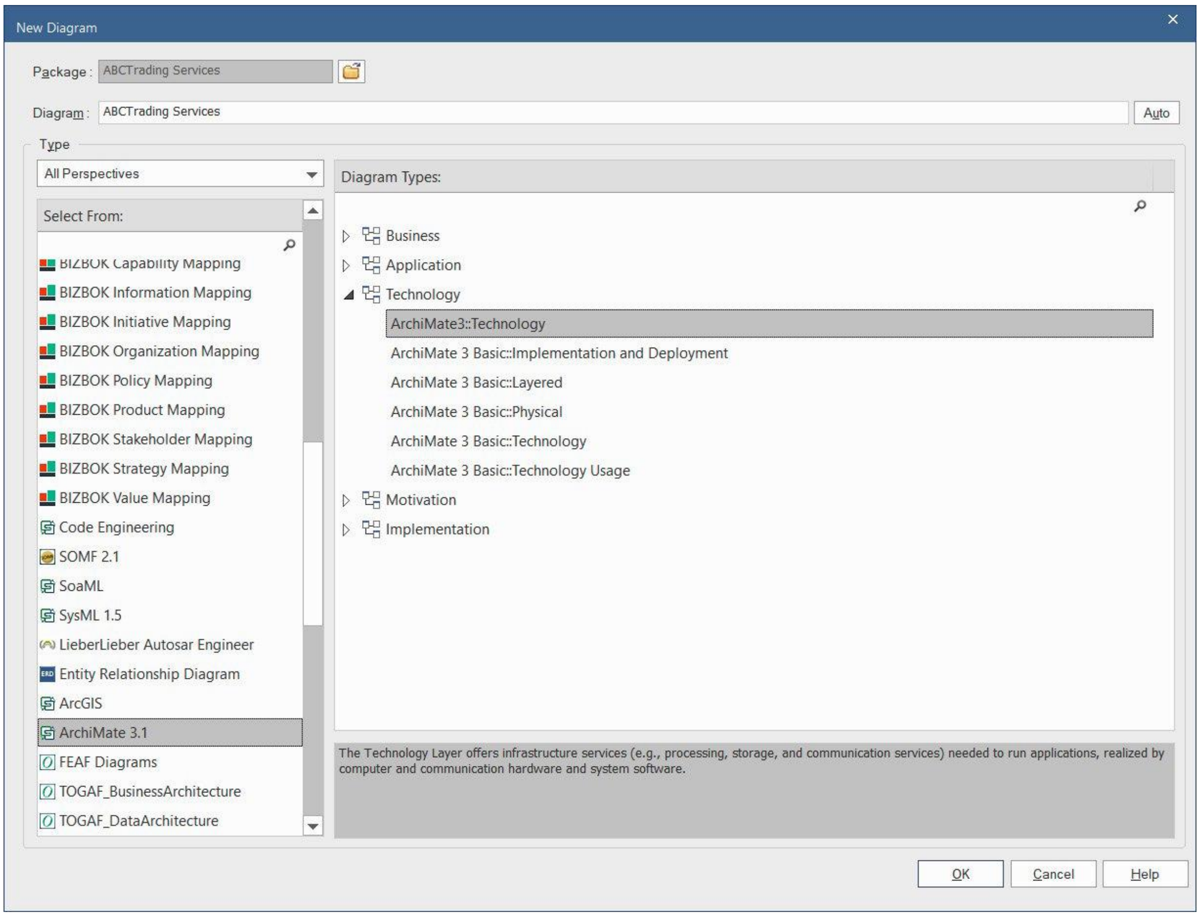
- Architecture Content
  - Artifacts Backlog
  - Enterprise View
    - Equipment
    - Servers
    - System Software
    - Technology Services
      - ABCTrading Services
      - TOGAF_Services
        - Data Interchange Services
        - Data Management Services
        - Graphics and Imaging Services
        - International Operation Services
        - Location and Directory Services
        - Network Services
        - Object-Oriented Provision of Services
        - Operating System Services
        - Security Services
        - Software Engineering Services
        - System and Network Management Services
        - Transaction Processing Services
        - User Interface Services

Figure – The location of the TOGAF® service categories

While these TOGAF® technology service categories are an excellent starting point for forming your technology services catalog, they aren't ideal for demonstrating our next step; there are just too many of them. For that reason, we will use a much smaller set of services that we've created from scratch.

The following is the process we will use to create our much smaller service catalog:

1. Select the **Enterprise View** element in the Project browser.
2. Click the new folder icon.
3. Name the folder `Technology Services`.
4. Create a sub-folder called `ABC Trading Services`.
5. Click on the **New Diagram** button.
6. Select **ArchiMate 3.1** and the **ArchiMate 3::Technology** diagram type from the **New Diagram** dialog:


New Diagram

Package : ABCTrading Services

Diagram : ABCTrading Services

Type: All Perspectives

Select From:
- BIZBOK Capability Mapping
- BIZBOK Information Mapping
- BIZBOK Initiative Mapping
- BIZBOK Organization Mapping
- BIZBOK Policy Mapping
- BIZBOK Product Mapping
- BIZBOK Stakeholder Mapping
- BIZBOK Strategy Mapping
- BIZBOK Value Mapping
- Code Engineering
- SOMF 2.1
- SoaML
- SysML 1.5
- LieberLieber Autosar Engineer
- Entity Relationship Diagram
- ArcGIS
- ArchiMate 3.1
- FEAF Diagrams
- TOGAF_BusinessArchitecture
- TOGAF_DataArchitecture

Diagram Types:
- Business
- Application
- Technology
  - ArchiMate3::Technology
  - ArchiMate 3 Basic::Implementation and Deployment
  - ArchiMate 3 Basic::Layered
  - ArchiMate 3 Basic::Physical
  - ArchiMate 3 Basic::Technology
  - ArchiMate 3 Basic::Technology Usage
- Motivation
- Implementation

The Technology Layer offers infrastructure services (e.g., processing, storage, and communication services) needed to run applications, realized by computer and communication hardware and system software.

Figure – The New Diagram dialog

7. If you're creating your organization's technology services catalog, select the **Technology Service** element type from the toolbox and drag it onto the diagram. Name the new technology service appropriately. Perform this step for each of the services in your enterprise. If you're following along with our example, skip this step and go to the next one.
8. If you're following along with our example, all of our technology services have already been defined in the `ABCTrading Services` folder. From the browser, simply select all of the technology services and drag them onto the diagram in one step.

The following diagram identifies a list of services supplied by *ABC Trading* technologies. As mentioned previously, this is an abbreviated set for demonstration:

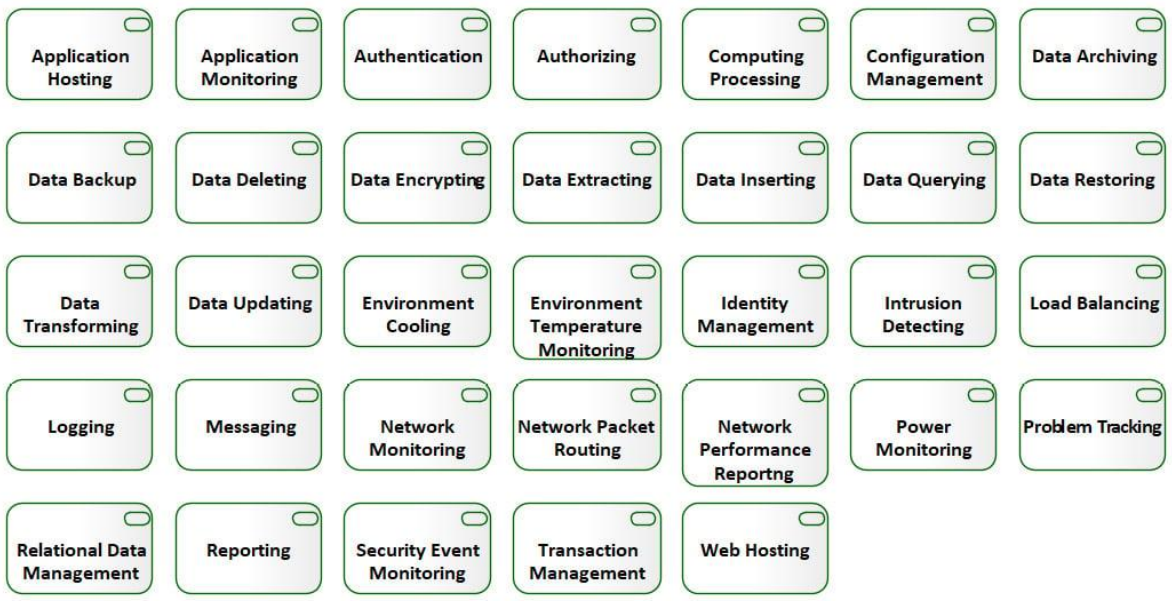


Figure – A technology services catalog example

Important Note

Creating a diagram such as the preceding one can be fast and easy. Simply select all the elements you want and drag them over. Then, select **Layout** > **Diagram Layout** > **Apply Default Layout**. You can easily try different layout types until you find the one you like.

As with every other enterprise element, technology services do not exist by themselves and must be related to other elements. We will see how technology services are related to technology components and how we can generate useful models using this relationship.

Mapping services to technology components

We need to identify which services are provided by each node in our repository. We say that a node is *assigned* to a service, so we do this by establishing an assigned link between the node and the function. The following figure shows us how a technology node mapped to a service might look:

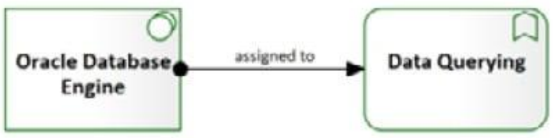


Figure – Using the assignment relationship

The most obvious way for us to establish a link like this is to create a diagram, drag the elements onto the diagram, and then drag a link from one element to the next. The following figure shows an example of how we do this for device nodes:

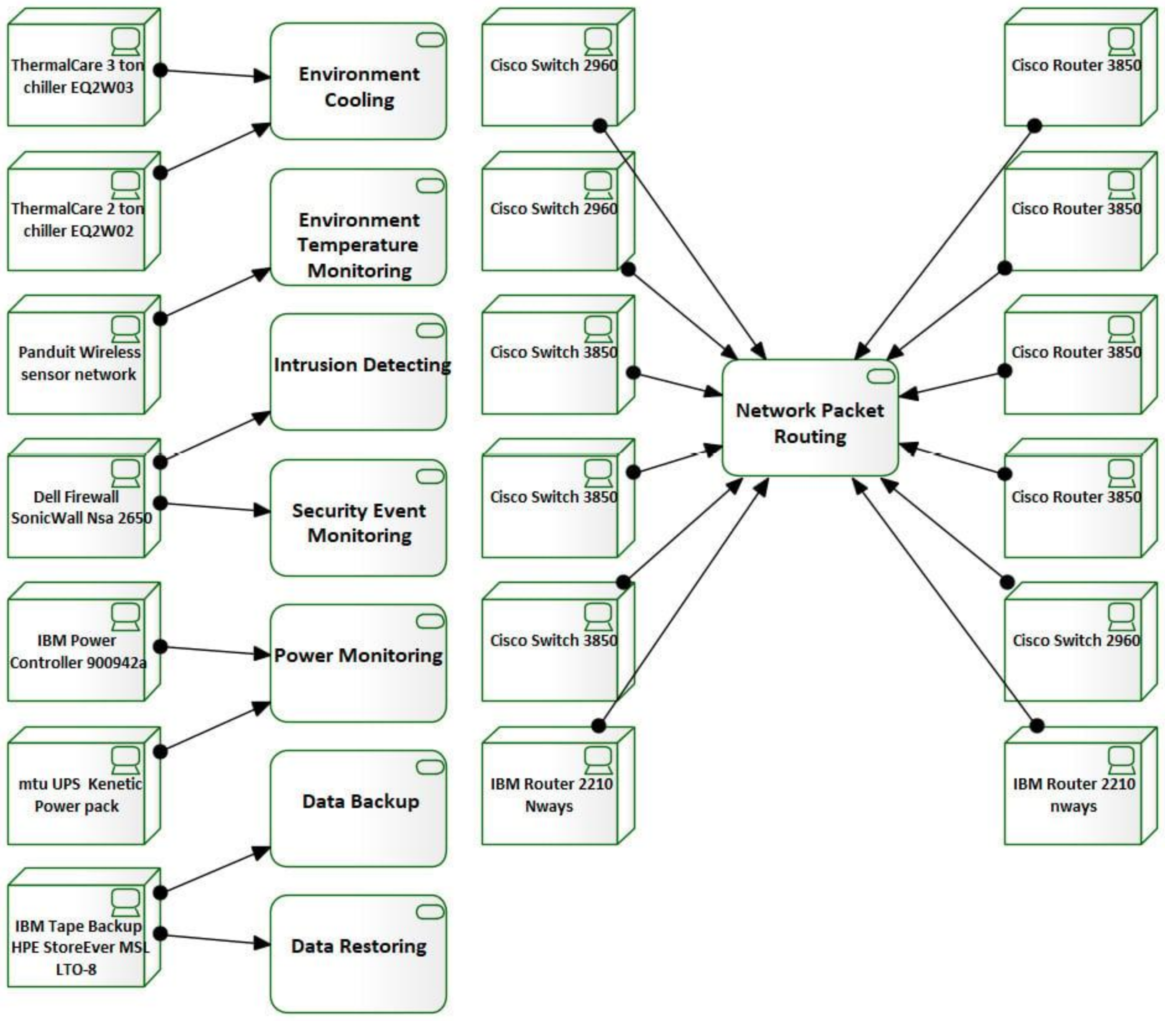


Figure – Assigning services to devices

As you can see, the problem with this approach is that there are a lot of connections to be made. Although this is a temporary work diagram, it still becomes cluttered rather quickly. With hundreds or even thousands of technology nodes and potentially hundreds of services, creating such a diagram becomes impractical. Nor would we want to create a diagram for each element. We'll take a look at another feature of Sparx that gives us an alternative to the diagram method – the matrix feature.

Using the matrix feature

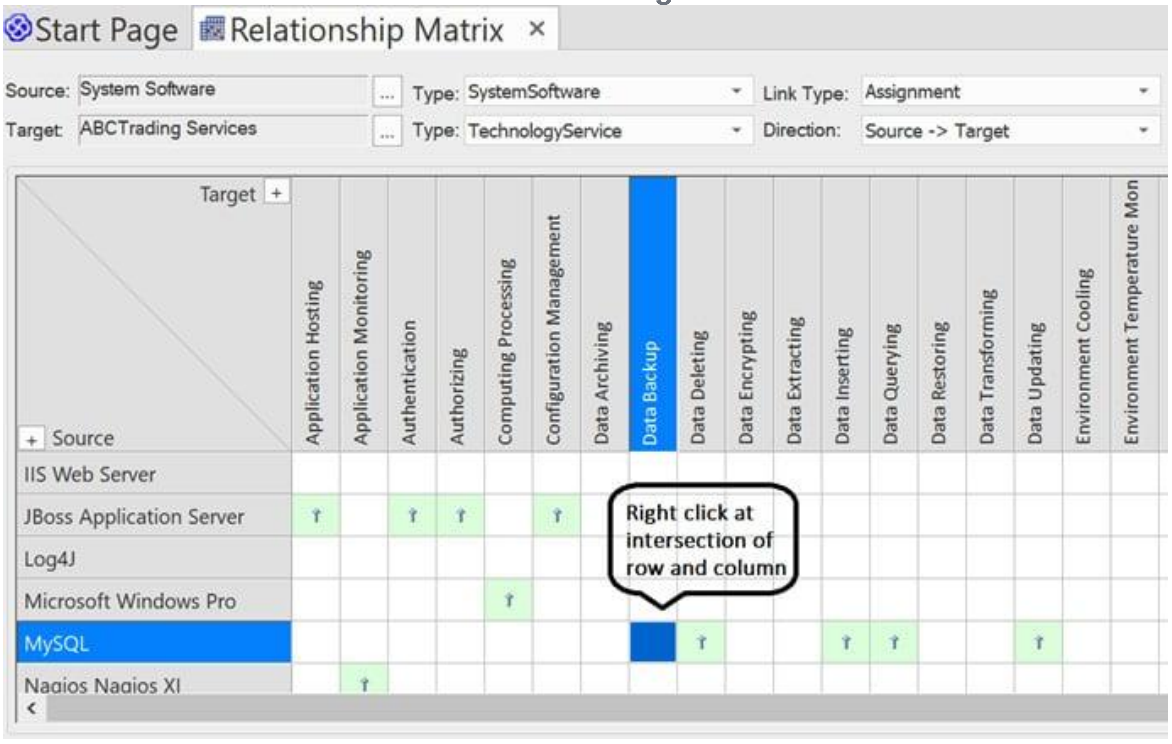
This simply explains an alternative to dragging all elements onto a diagram. This feature lets you establish links between elements using a two-dimensional matrix, with one package of elements as the rows and another package as the columns. The following are the steps to opening the **Relationship Matrix** tab:

1. In the Project browser, select the `System Software` folder.
2. Go to **Design** > **Matrix** > **Open as source**. The **Matrix** tab is displayed.

   Important Note

   The following screenshot of the Relationship Matrix, has been cropped in order to show the details. You may want to toggle full-screen mode by selecting **Start** > **Full Screen**.

3. The **Source** field should be prepopulated with the **System Software** folder name. In the **Target** field, browse and select the **ABC Trading Services** folder.
4. In the **Source Type** field, select **SystemSoftware** from the drop-down list.
5. In the **Target Type** field, select **TechnologyService**.
6. In the **Link Type** field, select **Assignment**.
7. The **Direction** field is set to **Source -> Target**:



Source: System Software ... Type: SystemSoftware Link Type: Assignment
Target: ABCTrading Services ... Type: TechnologyService Direction: Source -> Target

Target / Source	Application Hosting	Application Monitoring	Authentication	Authorizing	Computing Processing	Configuration Management	Data Archiving	Data Backup	Data Deleting	Data Encrypting	Data Extracting	Data Inserting	Data Querying	Data Restoring	Data Transforming	Data Updating	Environment Cooling	Environment Temperature Mon
IIS Web Server																		
JBoss Application Server	↑		↑	↑		↑												
Log4J																		
Microsoft Windows Pro					↑													
MySQL									↑			↑	↑			↑		
Nagios Nagios XI		↑																

Figure – The Relationship Matrix tab

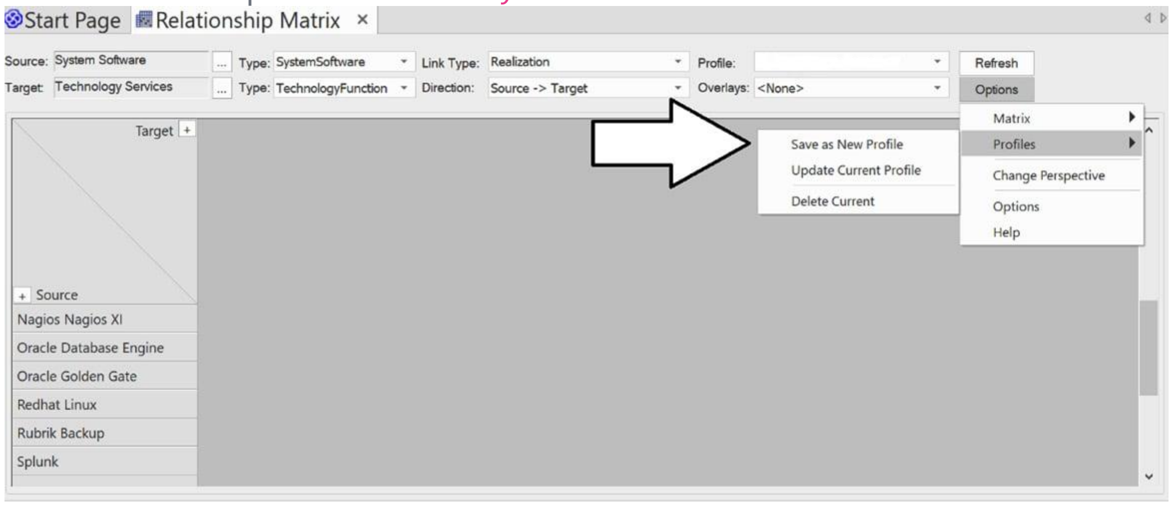
8. Select a cell from within the matrix.
9. Right-click on the cell and click **Create new relationship** > **ArchiMate3.1:Assignment**.

   Important Note

   You may need to reposition the boundaries between the rows and columns to see the full element name. Simply drag the boundary to reposition it.

Once we have the matrix the way we like it, we'll save the settings in a new matrix profile. This will allow us to close the tab and return later without reentering the configuration information. We will be able to simply select the saved profile from the **Profiles** drop-down list.

10. Press the **Options** button at the upper-right corner of the **Relationship Matrix** tab.
11. Select **Profiles** > **Save as New Profile**.
12. Give the new profile the name `SystemSoftwareServices`:



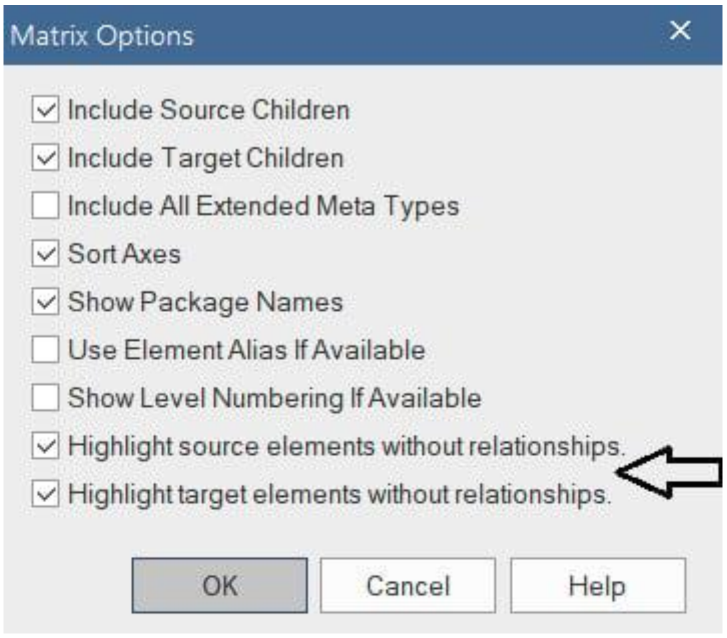
Source: System Software Type: SystemSoftware Link Type: Realization Profile:
Target: Technology Services Type: TechnologyFunction Direction: Source -> Target Overlays: <None>

Source:
Nagios Nagios XI
Oracle Database Engine
Oracle Golden Gate
Redhat Linux
Rubrik Backup
Splunk

Figure – Save the matrix configuration as a new profile

As you go through the process of creating links between nodes and services, you may realize that a particular service element doesn't quite fulfill your expectations. You have complete control over the model. You can rename a service or create a new one that is slightly different to fit the need. If you do this, you'll want to define these differences in the notes section of the model element. In fact, it's not a bad idea to document all of the technology services this way. It may help when trying to decide whether a particular node actually provides the service in question. Always keep the goal of the exercise in mind – in this case, to catch redundant or unused technology.

Once we have mapped each system software node to at least one service, we will need to repeat this process for the **Equipment** and **Servers** packages. As you progress through the process of creating these links, the matrix can become quite complicated. There are options available within the following **Matrix Options** dialog that allow you to highlight rows and columns without relationships. You can access this dialog from the **Options** button in **Relationship Matrix**.


Matrix Options

- [✓] Include Source Children
- [✓] Include Target Children
- [ ] Include All Extended Meta Types
- [✓] Sort Axes
- [✓] Show Package Names
- [ ] Use Element Alias If Available
- [ ] Show Level Numbering If Available
- [✓] Highlight source elements without relationships.
- [✓] Highlight target elements without relationships.

Figure – The Matrix Options dialog

These options result in a convenient means of keeping track of your progress. If you look at the Sparx Relationship Matrix on your **User Interface** (**UI**), you will find that system software rows with no relationships to services are highlighted in blue. Services with no related system software elements are highlighted in pink.

Once we're satisfied that our matrix is complete, we will turn our attention to how to present the information.

Reporting our findings

Once again, how we present our findings depends on the audience. There are many options available. We need to try to anticipate whether our audience would be more receptive to diagrams or lists. How much information is necessary? We also need to take into consideration what will be done with this information. What are the next steps?

In our case, the CTO is rather new to the company. They are quite experienced, but they came from a somewhat different industry. Perhaps we need to add more descriptions of the technologies in question. We need to ask ourselves the following questions:

- How formal do we need to be in presenting the information?
- Do we need to introduce this subject?
- Do we need to include a cover sheet?
- How much access do we have to this person?
- Are they so busy that we wouldn't be able to meet again for weeks?

These questions may have been answered by having worked with the CTO in the past, but we haven't.

The diagram option

The most obvious and easiest solution to presenting our findings is a diagram, but as we've seen while establishing these links, a single diagram can become unwieldy. It might be visually pleasing if we create multiple diagrams, each showing an example of problem areas. In this exercise, in cases where more than one technology component is assigned to a technology service, we have a potential duplication of technologies. In those cases, we create a new diagram and drag the offending elements onto the diagram for illustration. The following figure shows an example of one potential problem area at *ABC Trading*:

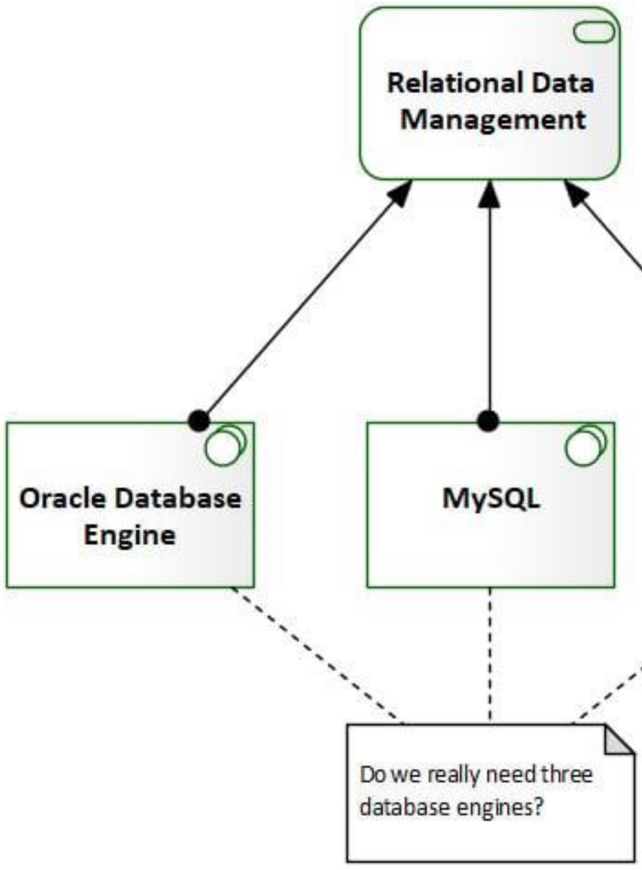
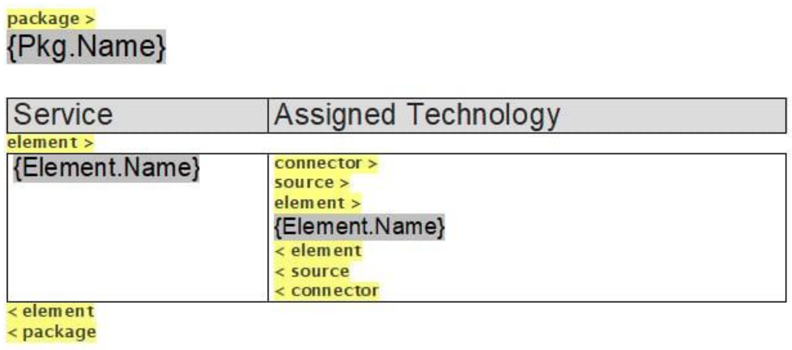

Relational Data Management
Oracle Database Engine
MySQL
SQL Server
Do we really need three database engines?

Figure – Duplicate technologies

The report option

An alternative to a diagram is generating a report based on the contents of the repository. We can build a simple report template that produces a list in table form. The template is shown in the following figure:



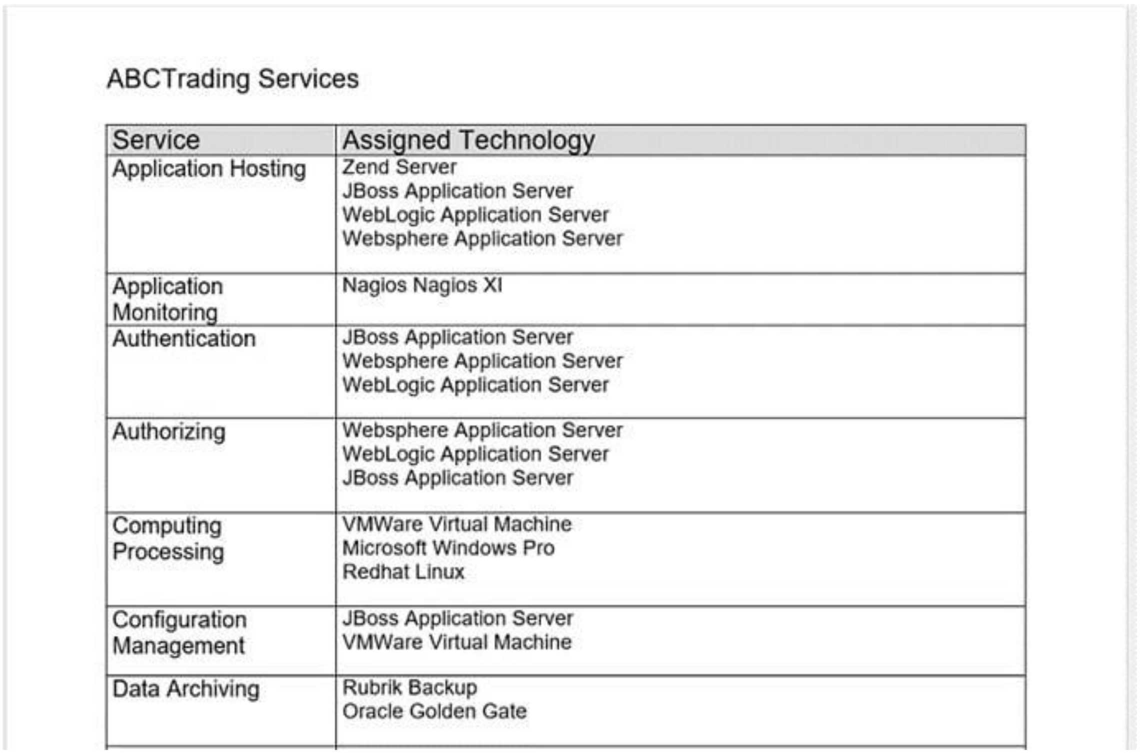
package >
{Pkg.Name}

Service	Assigned Technology
element >	
{Element.Name}	connector > source > element > {Element.Name} < element < source < connector

< element
< package

Figure – A simple report template

An example of report output is shown in the following figure:



ABCTrading Services

Service	Assigned Technology
Application Hosting	Zend Server JBoss Application Server WebLogic Application Server Websphere Application Server
Application Monitoring	Nagios Nagios XI
Authentication	JBoss Application Server Websphere Application Server WebLogic Application Server
Authorizing	Websphere Application Server WebLogic Application Server JBoss Application Server
Computing Processing	VMWare Virtual Machine Microsoft Windows Pro Redhat Linux
Configuration Management	JBoss Application Server VMWare Virtual Machine
Data Archiving	Rubrik Backup Oracle Golden Gate

Figure – A partial duplicate technologies report

As you can see, this report accomplishes the goal of identifying *potentially* duplicate technologies. Each service in the first column that contains more than one technology node in the second column can be a duplicate. For our scenario, we have decided to take an iterative approach. We will produce this simple report, then meet with the CTO, review the information, and determine at that time whether they need more information. The fact is that this information will need further scrutiny. In our scenario, however, this is sufficient to present to the new CTO. This may be all they need. More likely, they will solicit your help in narrowing this information down. That's fine, as all of this information is in reusable form; it's all in our repository. We can add, augment, or refine this information further as we see fit.