Kick-Starting Your Enterprise Architecture Repository

The words **diagram** and **model** will be used interchangeably as they both have the same meaning. Architects prefer to use model while users prefer to use diagram, so keep that in mind. **Metamodels**, in a nutshell, are models that tell us how to build models, so it makes sense to start by building the metamodels first and then derive models from them. However, in some cases, you may find that starting from the model and reverse engineering it to develop the metamodel is more convenient, especially if you are experienced in reading and interpreting the standard metamodels of TOGAF® and ArchiMate®. It is the ever-lasting question of what comes first, *the chicken or the egg*.

We will learn the following while practicing building the Tracking App application context diagram:

- Building the application component context diagram
- Establishing your first diagram
- Adding elements to the diagram

Building the application component context diagram

A **context diagram**, as the name implies, is a diagram that shows the *surroundings* of a specific component. The main purpose of context diagrams is to provide a high-level introduction to an element such as an application component by answering the following questions:

- What is the element and why does it (or will it) exist?
- What does it provide to the enterprise?
- Who will be using it?
- Are there any other applications that interact or exchange data with it?

- What data is being (or will be) exchanged?

Context diagrams can be useful for describing existing elements (as is) and for describing targeted elements (to be). The same concept applies, and the same types of elements can be used in both cases.

Establishing your first diagram

Diagrams visually show how elements are related to convey an idea, such as what a specific component is composed of, how a specific service is provided, or how a group of data centers are connected and what they contain. An element can be represented by different diagrams, each showing a different aspect of it, so the same element can appear in many diagrams.

Packages, on the other hand, represent a physical containment of their content, such as elements, diagrams, and other packages. A child element can have one parent only, but parents can have many children. Packages look and act like folders in file systems as they *contain* other elements, while elements and diagrams are the equivalents of files in file systems. You can nest packages within packages as much as you may need, but you need to keep in mind that very deeply nested packages can make it difficult for you and other repository users to find the information, so keep the users in mind when you build the structure.

If users find it easy to locate and get information in the enterprise architecture repository, they will use it as a reliable source of information; otherwise, they will keep doing whatever they are comfortable with. Therefore, make sure that the structure of the repository, including the names of the packages, elements, and diagrams, makes sense.

Migrating content between your local repository and the centralized repository can take place at any package level, so you can export only the new content of the package that you need. From the Sparx application, navigate to the **Publish** ribbon. From the **Model Exchange** section of the ribbon, select **Export-XML Format.**

In this section, we will learn the following:

- How to create the repository file
- How to create the diagram

- How to describe the diagram
- How to change the diagram theme

Creating the repository file

After starting Sparx, you will be welcomed by the start page, which asks you whether you want to open a file, create a new one, or open one from the recently opened projects. This is our first diagram in a brand-new repository, so we need to create a new project:

1. Choose the **Create a New Project** option under the **Open** menu and select a folder to place the Sparx project file in.
2. Type any name of your choice; I will use the name `EA Repository` for this purpose.

You should now see a single item named **Model** in the Project Browser window. This is called the **root node** and we will use it as our top-level package for our architecture content for now.

3. We need to give the root node a more meaningful name than **Model**, so click on the root node then press *F2* to rename it to `Architecture Content`.

The structure of the repository is extremely flexible when there is a need to change it, and this need will come sooner or later. The enterprise will keep changing and the way you understand it will keep changing too, so the repository must be flexible enough to accommodate these changes with ease. Fortunately, this can be easily done in Sparx, and it is similar to organizing files and folders in a file system.

For example, if you put an element in package A, link this element to other elements, place the element on several diagrams, and then decide to move it to a different package, the links will not be broken no matter where the element is relocated.

We have an empty repository, so now we need to create a new package to contain our first diagram and all the related elements.

Creating the diagram

We need to create a package to contain the diagram that we will create and that will contain all the elements that will be placed on it. Diagrams cannot be created under

the root node directly; they need to be created within a package under the root node or under another package.

Creating packages and creating diagrams can be performed together in one step or separately as two steps. In this example, we will create both in one step.

To create a package, follow these steps in order:

1. Click on the **Architecture Content** root node and from the Toolbar, choose **Design** > **Package** > **Add Package** to create a new package. This will bring up the **New Package** pop-up window, as in the following figure:

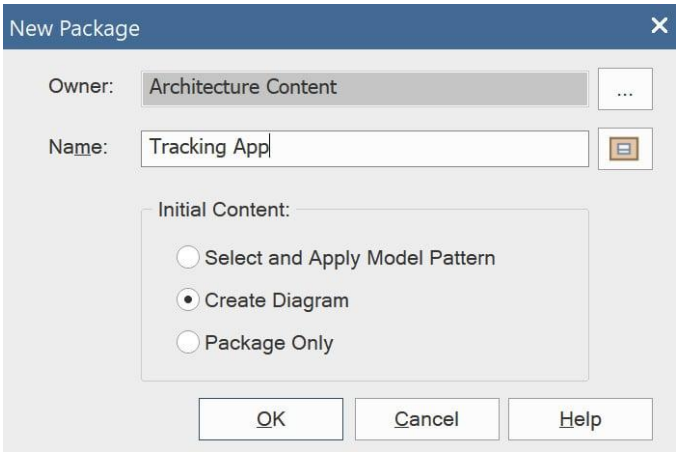

Figure – New Package pop-up window

2. Enter `Tracking App` in the **Name** field and select **Create Diagram**. Click **OK**, and the **New Diagram** window will open.

If you selected **Package Only** in the **New Package** window, the **New Diagram** window will not be launched automatically, and you will get an empty package. You can add a diagram at any time by right-clicking on the package and selecting **Add Diagram** from the context menu, which will bring up the same **New Diagram** window, as follows:

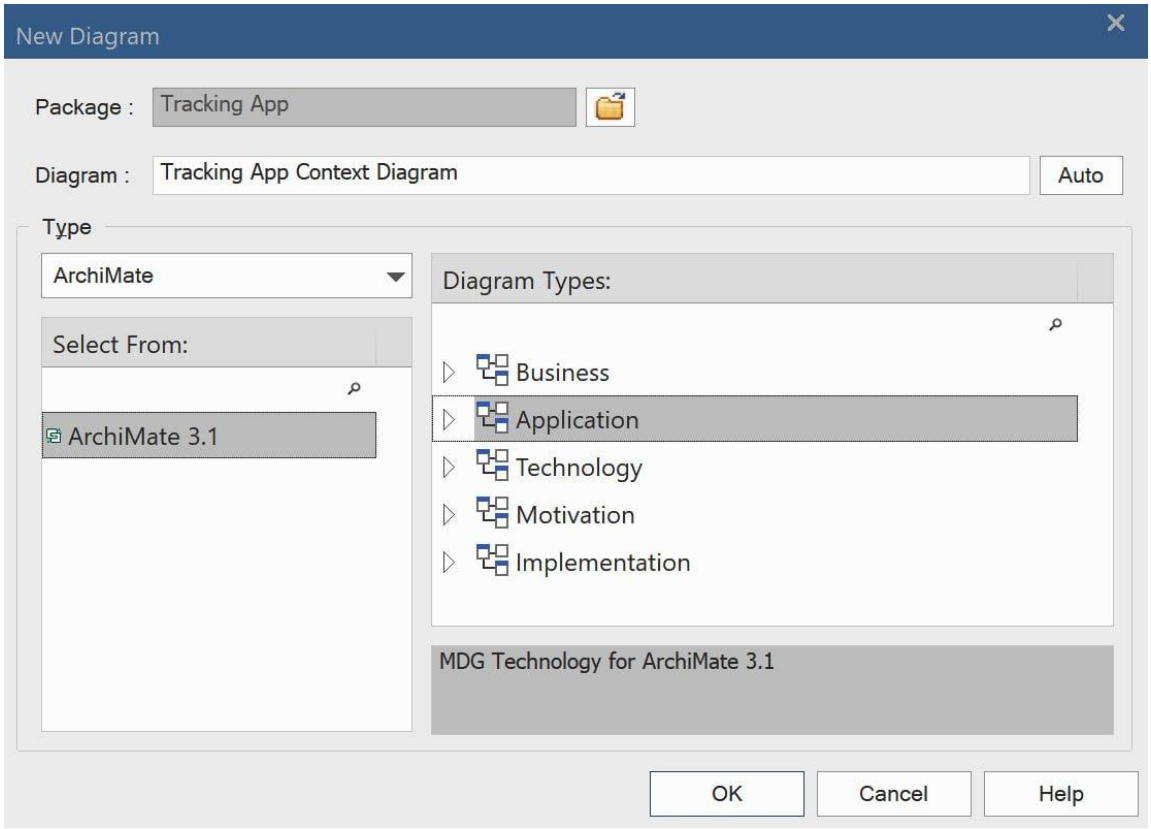

Figure – The New Diagram window

3. Type `Tracking App Context Diagram` into the **Diagram** field, select **Enterprise Architecture** > **ArchiMate** from the **Type** drop-down list, highlight **ArchiMate 3.1**, and select **Application** from the available diagram types.
4. Click **OK** to accept the selections, which will close the window and open a blank diagram for you to start drawing on.

We need to pause for a few seconds here and explore what we have on the screen after the previous actions. As you can see, an empty diagram is now shown in the diagram area as a **tab**. Clicking on the **x** icon on the tab will close it, and you can re-open it from the project browser window. In the project browser area to the left of the screen, you can see the newly created diagram under the **Tracking App** package. If you *double-click* the diagram icon, the diagram will be reopened and displayed in the diagram area again, as shown here:

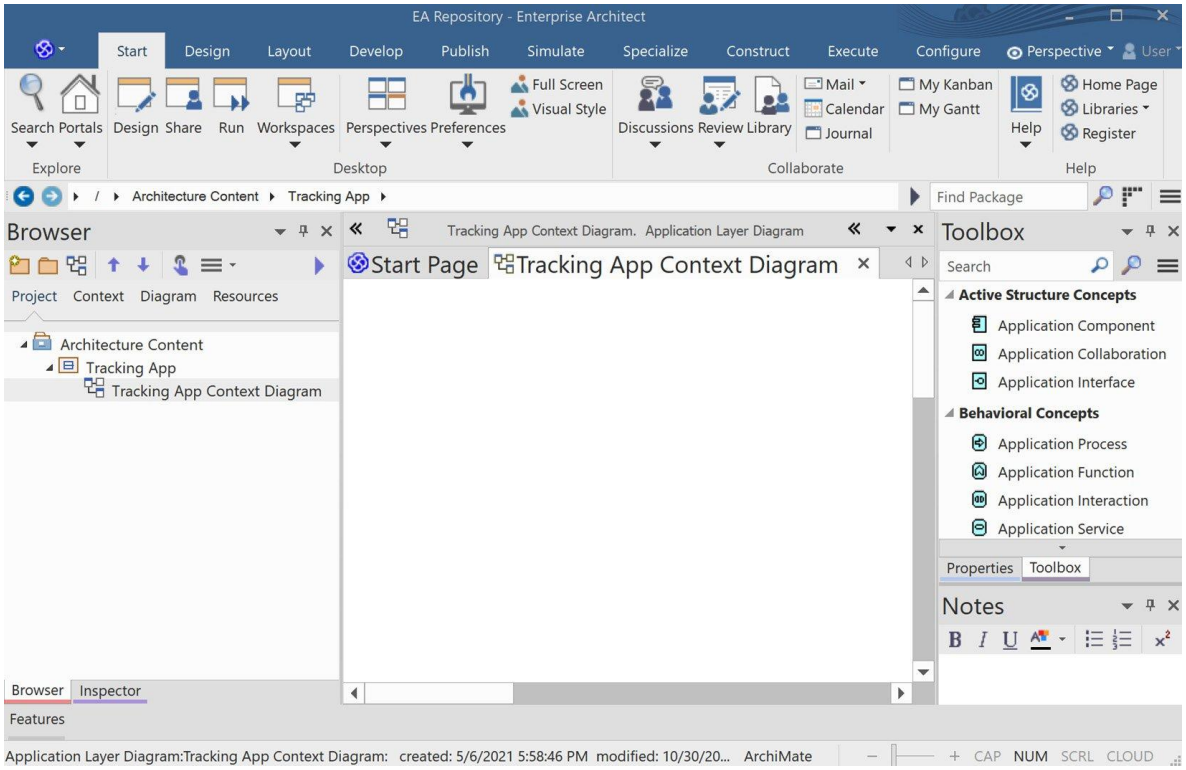

Figure – Empty application component diagram

To the right, there are two areas: the **Toolbox** and **Notes** areas. The **Toolbox** area changes according to the type of diagram that you have chosen in the **New Diagram** window. Because we chose **Application** as the diagram type, the **Toolbox** contains all ArchiMate® 3.1 Application-layer elements.

The **Notes** area is context sensitive to the currently selected item. If you select a different element from the Project Browser window, the **Notes** content will change accordingly. We will use the **Notes** area to describe all the elements that we create and make the content ready for publishing.

Describing the diagram

It is a good practice to add descriptive text for every diagram and every element you create. What you think of as obvious information may not be for other users, and sometimes may not be obvious to you if you revisit a diagram that you have created after a while. Adding comments in code while programming is just as important for helping yourself when you come back to it as helping other developers who will be working on it.

Using labels

Labels act like titles to diagrams and they immediately inform the readers what the diagram is about. Labels are elements, so we need to add them from the **Toolbox** to the diagram area and add descriptive text to them. Follow these steps to add a label to a diagram:

1. Scroll down the **Toolbox** all the way near the bottom until you find the **Common** section.
2. Expand the **Common** section and find **Text Element**. Click and hold **Text Element** and drag and drop it onto the diagram. Alternatively, you can click once on **Text Element** (or any element in general), move the mouse where you want to create it, and click on the diagram to create the element where you clicked.
3. Type `Tracking App Context Diagram` in the designated text area and click anywhere on the diagram to see your text element updated.
4. You can format the font and adjust the font size by locating the font configuration button in the **Layout** > **Style** menu, which is indicated by a capital letter **A**.
5. Select the font name, style, and size that you like from the pop-up window. The defaults are **Calibri**, **Regular**, and size **8**.
6. Since this is a label, change the size to **16** and press **OK** to accept and close.

You can move the label and place it at any desired location on the diagram, but labels are usually placed at the top-left corner of the diagram area as this is where readers will usually start reading.

Using the diagram notes field

Diagram notes are fields that can contain any text that you feel is useful to add. They are not visible in the diagram area, but Sparx users can see them in the **Notes** window when they click on the diagram area. Another useful benefit of entering a diagram description in the **Notes** field is making the documents that will be published from Sparx richer in content. You can customize your document templates in a way that prints the **Notes** field before or after the diagrams.

To add a description in the diagram **Notes** field, we need to do the following:

1. *Right-click* on the diagram background and select **Properties**, which will open the **Properties** pop-up window for the diagram. As you can see, there are multiple tabs in this popup, and we will explore some of these options as we go. On the first tab, labeled **General**, you can rename the diagram, change the author's name, change the notes, and adjust other settings.
2. In the **Notes** field at the bottom, type any text that describes what this diagram is about. You can use this sample text if you want: *This diagram shows the context of the Tracking App application component. It describes what the application component is, what it provided, who will use it, what other applications will integrate with it, and what it needs in order to operate*.
3. You can educate your Enterprise Architect users to look at the **Notes** section of diagrams if they want to know what the diagrams are about and what viewpoint they are addressing. Make your notes short and descriptive because most of the description should be conveyed in the diagram that you are creating, not in a text note.

Keep this popup open as we need to change the theme of the diagram in the next step.

Changing the diagram theme

Themes are predefined settings that affect the way your diagrams look, mainly by applying different background and foreground colors. Themes are applicable at the diagram level, so you can have different themes for different diagrams.

However, because the selection of themes affects the way your diagrams look, I advise you to stick to one theme for the entire repository. Every enterprise architect who contributes to the content will be required to use the same theme to maintain consistency. If you think that this will be too much to enforce, then you're better off using the default theme and asking everyone not to change it.

To change the theme of the current diagram, you need to have the diagram properties popup open:

1. Go to the **Theme** tab and scroll through the dropdown until you find **High Contrast White**.
2. Select **High Contrast White** from the list and click **OK** to apply the changes, and then close the popup.

The choice of themes and colors is purely up to your preference, so you are free to pick any one of them you feel comfortable with.

Note

If you are working with a team of architects, you all need to agree on the same theme and use it to maintain consistency among your deliverables. If team members are using different themes but not using the default colors of that theme, the way diagrams will look will vary from one computer to another, which affects the consistency among enterprise architecture team members.

We have established the initial structure for the content that will be created and established an empty diagram.

Adding elements to the diagram

Element is a generic word for anything you place on a diagram. New elements are provided in toolboxes, and for each type of diagram, Sparx makes one toolbox the default toolbox for that diagram. When creating our diagram, we chose that we wanted to create an ArchiMate® 3.1 Application diagram; therefore, Sparx has set ArchiMate® 3.1 Application as the default toolbox for us. This can be changed, but first, we need to add the main element to the diagram, which is the application component.

Starting with the application component

It is always a great idea to have an imaginary picture of the diagram in your head or a sketch on a piece of paper before starting to model it. In this diagram, we need to convey what the new application is about, what it provides, who will be using it, and what other applications need to integrate with it. This should be a conceptual diagram that describes the application at a high level, and it should be easy to read and be understood by any audience.

Defining an application component

An application component is defined as "*an encapsulation of application functionality aligned to implementation structure, which is modular and replaceable*"

An application component is meant to provide one or more *functions* to users and other applications (formally known as **Actors**). It *encapsulates* these functionalities, so it is unknown to the Actors how it works internally, but services are provided to them through interfaces. Interfaces can vary based on the Actors, such as the **User Interfaces** (**UIs**) for communicating with human Actors (or *users*) and the **Application Programming Interfaces** (**APIs**) for communicating programmatically with other applications. In modern programming, UIs are nothing more than loosely coupled application components that communicate data captured in forms to other backend application components through APIs.

Application components can comprise other application components, so the size of the component does not matter if it is modular and replaceable. An application component can be as large as a monolith **Enterprise Resource Planning** (**ERP**) application that has hundreds of modules, each having hundreds of submodules, or it can be as small as a microservice that takes in a text value and returns it with double quotes around it. These are both considered application components according to the definitions.

ArchiMate® 3.1 provides two notations for modeling application components, rectangular and borderless, as shown in the following figure:

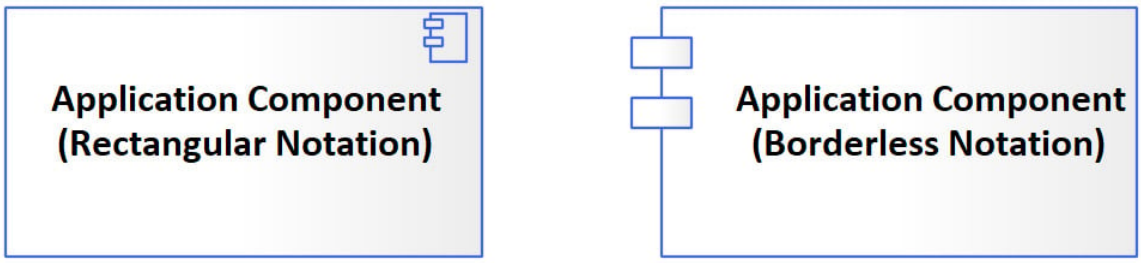

Figure – Application component notations in ArchiMate®

Creating an Application Component

The first element that we need to add to the diagram is the Application Component element, so follow these steps:

1. From the **Toolbox**, drag an **Application Component** element and drop it in the center of the diagram.
2. Sparx will expect you to rename the element to a more meaningful name, so you can type `Tracking App` and then press *Enter* or click anywhere outside the element.

You have performed two steps in one: you have created a new element in the repository that represents Tracking App, and you have placed that element on the diagram. There are other ways to create elements directly in the Project Browser without having them placed on a diagram, but this is the most common and easiest way to do it.

Keep in mind that if you want to create another diagram that contains Tracking App, you must reuse the same application element from the Project Browser. If you drag another Application Component element from the Toolbox and name it `Tracking App` too, Sparx will accept it and will never check for or warn you about possible duplicates because the name is not a unique identifier for elements.

You need to know whether you want to create a new element or reuse an already existing one. Always remember to drag and drop from the *Toolbox* to *create new* elements and drag and drop from the *Project Browser* to *reuse* existing elements.

If you are not sure whether an element exists or not, you can press *Ctrl* + *F* to search for it and reuse it. Searching for elements is essential to building a well-organized and trusted enterprise architecture repository, and luckily, Sparx has a good search engine that you will find handy.

Adding a description to a component

Make it a habit to describe every element you create in the repository, because the enterprise architecture repository is supposed to be the source of the most accurate information about your enterprise elements:

1. Open the **Properties** window of the **Tracking App** element by right-clicking on the element on the diagram and selecting **Properties** > **Properties**. Alternatively, you can select the element and then press *Alt* + *Enter* as a keyboard shortcut to open the same **Properties** window.
2. Type in a short description of the application component in the **Notes** section, such as *This mobile app will be installed on truck drivers' smartphones to allow the shipping managers and partners to track the location of the truck at any given moment*.

Displaying a note in a Note element

After you close the **Properties** window, the notes will be hidden, and if someone is looking at a published or printed version of the diagram, they will not be able to see the notes. Therefore, we need to place a **Note** element to show this information:

1. Scroll down the **Toolbox**, expand the **Common** area, and you will see the **Note** element at the top of the list.
2. Drag the **Note** element and drop it onto the diagram area close to the Application Component element.

   Important

   Do not get confused between the **Notes** *field* and the **Note** *element*. Every element and every diagram has a designated **Notes** field that you can use to write descriptive text about that element or diagram. These notes are not visible on the diagrams by default but can be visible in some Sparx windows. On the other hand, there is a **Note** element, which is part of the **Common** group in every toolbox. You can place it on a diagram to display a descriptive message about anything you want, and it is always visible to the reader, just like a *Post-it* note.

Notice that adding the **Note** element to the diagram did not appear in the project browser; that is because some elements, such as **Note**, **Text**, and **Boundary**, are not meant to be reused and they belong only to the diagram that they have been placed on. In other words, they are not considered **architectural elements**, so they have no physical presence in the enterprise architecture repository, but just a visual presence on a specific diagram.

The **Note** element is empty, and you can type anything in it or link it to another element and display that element's **Notes** field instead. In our case, we need to display the text that we have typed in the application component's **Notes** field to make it visible always because our goal is to tell the readers what the application is.

3. Click on the **Note** element and you will see three small action buttons on the right side of the element: an arrow, a menu, and a brush, as you can see in the following diagram:

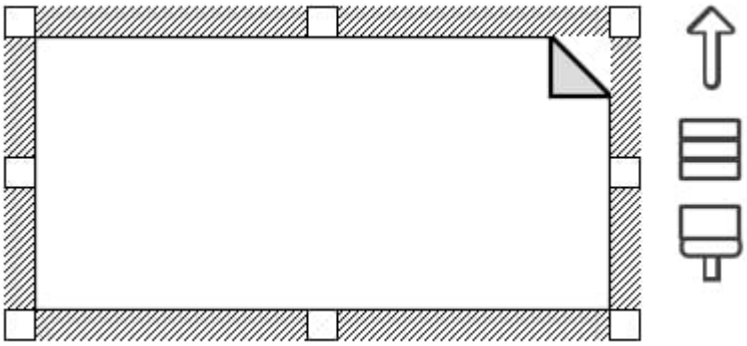

Figure – Action buttons appear when you highlight an element

These three buttons provide contextual actions, and they are available for each element on the diagram. Click on the Application Component element now and you will see the same action buttons next to it:

- The *arrow*-shaped button is used to create *relationships* between elements.
- The *menu*-shaped button displays a *context menu*.
- The *brush*-shaped button provides appearance and *formatting* options.

You can still access all these options using different menus, so it is just a matter of convenience.

4. Click and hold the arrow button and drag it to the Application Component element. Once the mouse pointer is above the application component, release the mouse and you will get a list of possible relationships that you can create between the source (in this case, the **Note** element) and the target (in this case, the Application Component element).
5. For these two types of elements, there is only one type of possible relationship: **Link**. When selected, you will see a dotted line now connecting the two elements, which represents a **link relationship**.

The **Note** element is still empty, and we can type the same text description that we typed earlier in the application component's **Notes** field. However, this means that we must maintain that text in two different places now. This can be a time-consuming task when you have a large repository full of elements and notes, and most of the time, you will end up making changes in one place and forgetting the other, resulting in inconsistency.

What we need to do is to tell Sparx that we want to display the exact same application component notes in the **Note** element.

6. To do that, *right-click* on the link relationship and click on the **Link this note to an element feature** option, which will bring up a pop-up dialog box.
7. Open the **Feature Type** drop-down list and select **Element Note**.
8. Click **OK** to accept the changes and close the popup.

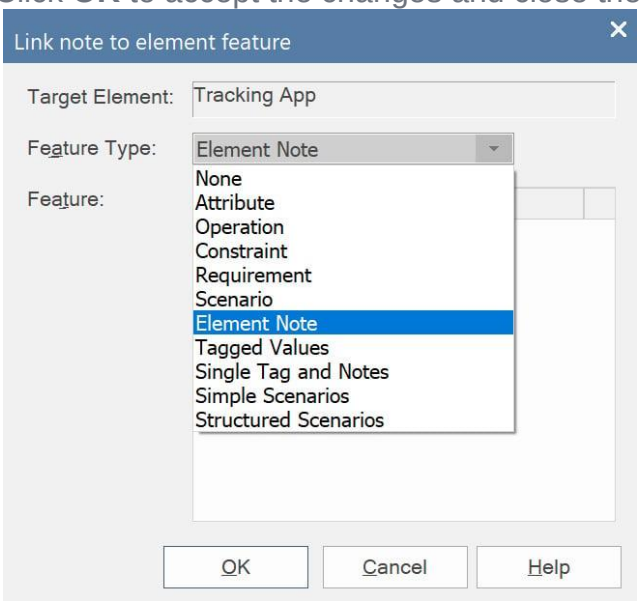

Figure – Link note to element feature pop-up window

You can see now that the text that you typed in the **Note** section of the application component is now displayed in the **Note** element on the diagram.

Styling elements

You can optionally change the color of the Application Component and **Note** elements to give them a nicer look. Take the following steps to do that:

1. From the **Layout** > **Style** menu, set the fill color to *white* (RGB 255, 255, 255) and the border color to *royal blue* (RGB 65, 105, 225).
2. From the same **Layout** > **Style** menu, change the font size from **8** to **10**.

Since Application Component will be the main element on the diagram, let's give it a thicker border and make it bigger in size so the reader will be able to tell from the first look which element in the diagram is the main subject.

3. Using the same **Layout** > **Style** menu items, adjust the width of the border from the default value of **1** pt to **2** pt.
4. Also, change the fill color of the **Note** element to *light yellow* (RGB 255, 255, 224) to give it the style of a real *Post-it* note.

5. Finally, change the notation of the application component to the borderless notation by right-clicking on it and unchecking **Advanced** > **Use Rectangle Notation**. Alternatively, you can click on the Application Component element, then click on the brush action button and uncheck **Use Rectangle Notation** from the menu.
6. Save your changes by pressing *Ctrl* + *S* on your keyboard or click on **Layout** > **Diagram** > **Save** from the Toolbar.

By now, your diagram should look like the following diagram, or you should at least know why it does not. You can see the label in the top-left corner, the application component, and the **Note** element displaying what is in the **Notes** field of the component:

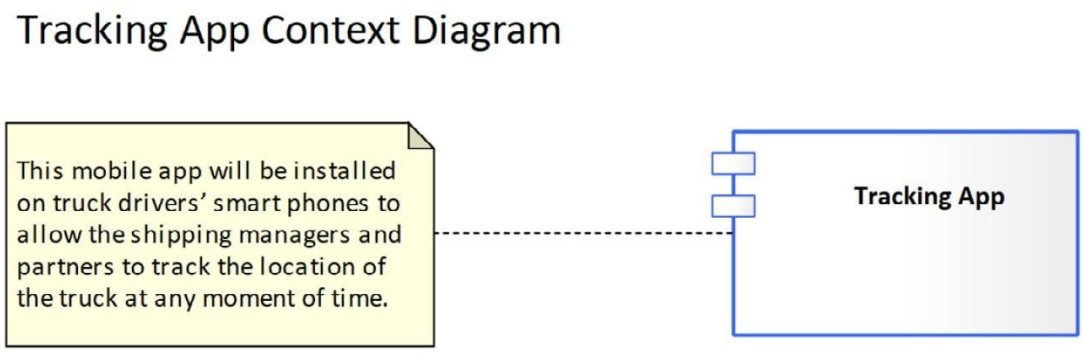

Figure – The Tracking App context diagram so far

The next thing that we need to show on the diagram is what Tracking App provides to its users.

Introducing application services

Application components are meant to provide services to users, so we need to add more information to our diagram telling them what services Tracking App will provide. Tracking App is targeted to provide a single service within the current scope, which is *Vehicle Tracking*.

The best way to identify the services that an application component provides is to refer to the **Use Case** documents if available. Use cases describe how Actors will interact with a system; they tell you what services a component exposes to the external Actors, which can either be users or other applications. Remember that we are still conceptualizing the application so we may discover more information as we share the diagrams with the stakeholders. A simple diagram will help stakeholders to

reorganize their thoughts and visualize what they are expecting to see in the target solution, so high-level use cases are what you need to consider at this stage.

Adding an Application Service element

We need to describe the services that are provided by Tracking App. ArchiMate® has provided the Application Service element for this purpose, and each service will be represented by an Application Service element.

Like what we have previously done when we added an application component and a note to the diagram, the Application Service element can be found in the same toolbox. Follow these steps to add the Vehicle Tracking service to the diagram:

1. From the **Toolbox**, drag an **Application Service** element, drop it on the diagram, and rename it to `Vehicle Tracking`.
2. Open the **Element Properties** window and in the **Notes** section, type a short description about this service, such as *This service provides information about the current location of the smartphone that will be carried by the truck drivers during shipment delivery*.
3. Click **OK** to close the **Properties** window and return to the diagram.

To remain consistent with other elements in the diagram, we need to style the Application Service element.

Styling the element

Optionally, style the Application Service element to give it the look and feel of other application architecture elements:

1. Follow the same steps that we defined for the application component to style the application service. Make the font size **10**, the fill color *white* (RGB 255, 255, 255), and the border color *royal blue* (RGB 65, 105, 225).
2. Since the application service is not the center of focus in this diagram, we need to keep the border width as **1**.

To avoid repeating the same styling steps for every element we add, it will be better if we save the style and reuse it whenever needed.

Saving the style for reusability

We will be using the same element style over and over for every application architecture element in the repository, so it is best to save it for later reuse. Saving styles also increases the consistency between different team members as they will all use the same style. Follow these steps to save the style for reuse:

1. Highlight the Application Service element on the diagram and locate the **Layout** > **Style** > **Manage appearance styles** toolbar button (which has a disk and pencil icon).
2. Click on the small arrow to open the drop-down list and select **Save as New Style**. Self-descriptive names are always recommended but we are limited to 12 characters only for the style name. Type `App Arch` as the value and click **OK** to accept.

The name of your newly added style now appears in the styles drop-down list in the same menu items group. You can now apply this style to any application architecture element that you create simply by highlighting the element and selecting this style from the menu.

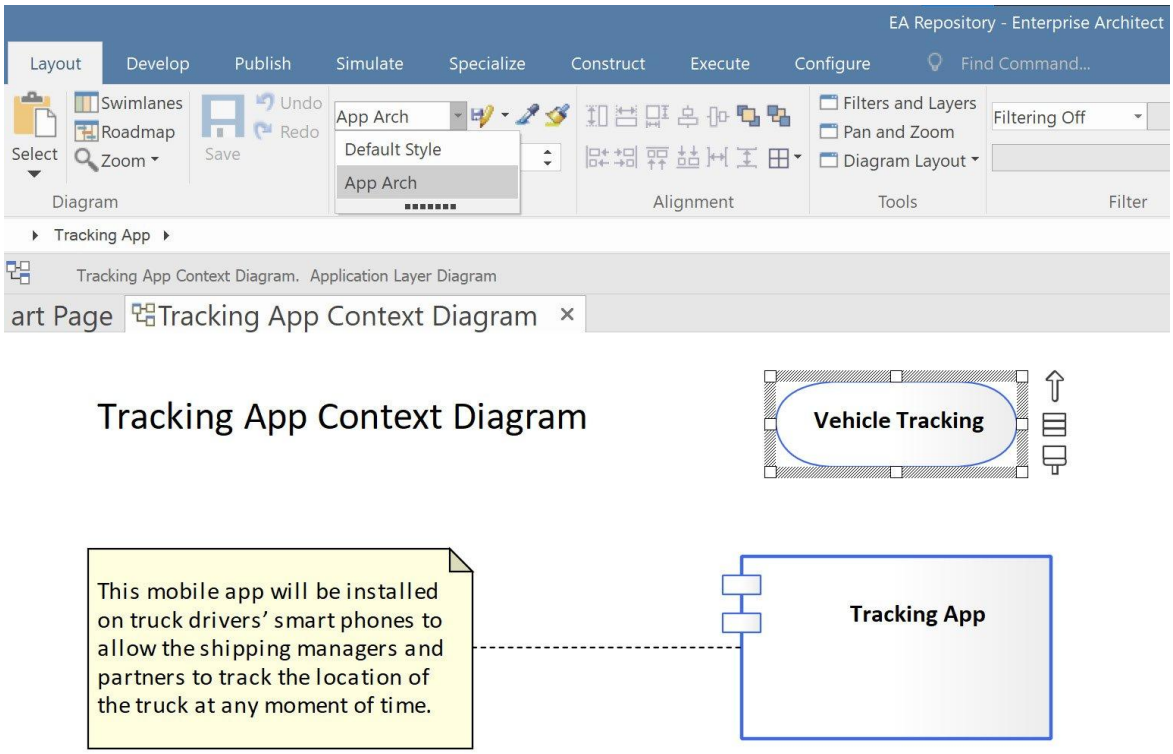

Figure – The style of the highlighted element is saved for future reuse

Now we have the application service created and styled, we need to relate it to the Tracking App application component.

Relating the component to the service

According to the ArchiMate® 3.1 specification, the relationship between an application component and an application service is **assignment**. Assignment simply means that we are describing part of the Tracking App behavior in the Vehicle Tracking service. This relationship is represented as an arrow with a solid line and a dot at its base, as you can see in the following diagram:

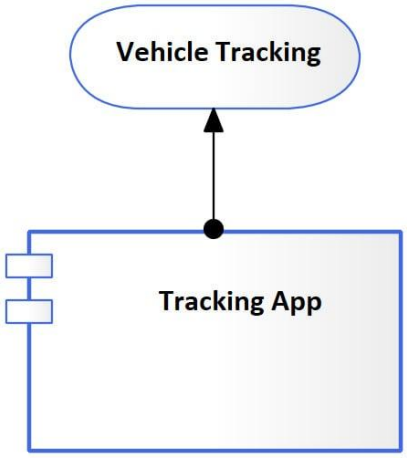

Figure – The assignment relationship

We need to create an assignment relationship between the Application Component and Application Service elements, so we need to do the following:

1. Locate the assignment relationship in the **Toolbox**, click on it once, and you will see the shape of the mouse pointer changing to a hand symbol, indicating that you are now in relationship creation mode.
2. Click on the application component and keep holding the mouse button, move your mouse over the application service, and release it.

Alternatively, you can click on the application component, click and hold the arrow-shaped action button, release it on the application service, and select **Assigned to** from the menu.

3. Press *Ctrl* + *S* to save.

This will create the desired relationship between the two elements. This relationship is not only visible on this diagram, but it also connects them internally and will appear automatically whenever you place both elements on any other diagram. This relationship is part of the enterprise architecture repository and is not limited to the current diagram.

To know the relationships between any element and other elements, highlight it and open the **Properties** pop-up window by pressing *Alt* + *Enter* or right-clicking on the element and choosing **Properties** > **Properties** > **Related** > **Links**, as shown in the following figure:

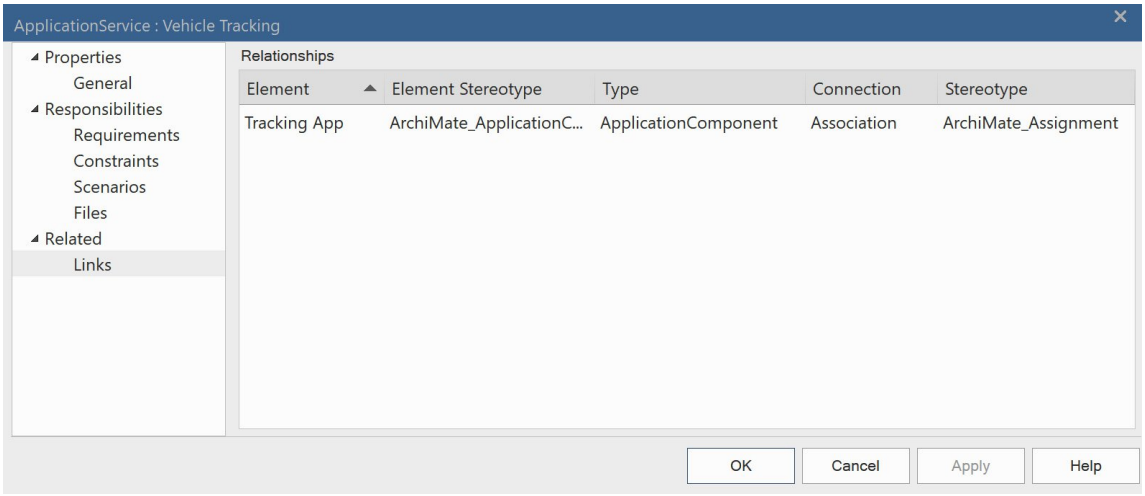

Figure – Relationships between the current element and other elements

If you discover a need for adding additional application services, you can follow the same steps to add them. For now, your diagram must look as in the following figure:

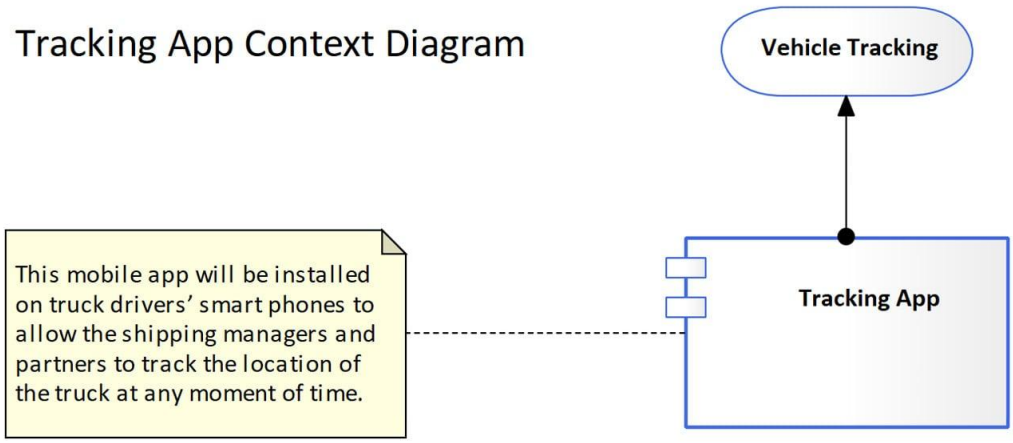

Figure – Tracking App and the service it provides

Adding Actors

Now we need to place the users of the application on the diagram, and we will use the **Business Actor** element for this purpose. Business Actors are not part of the Application toolbox, so we need to bring in the ArchiMate® 3.1 Business toolbox instead. We will perform the following steps to add Business Actors to the application context diagram:

1. Activate the ArchiMate® 3.1 Business toolbox.
2. Place the Business Actors on the diagram.
3. Define the proper relationships between the Business Actors and other elements.
4. Style the Business Actors.

Activating the ArchiMate® 3.1 Business toolbox

Sparx made ArchiMate® 3.1 Application the default toolbox for our diagram because we earlier chose that we wanted to create an ArchiMate® 3.1 Application diagram. However, in many cases, we may need to add elements from different toolboxes, so Sparx provides a way to do this. If you look at the top part of the **Toolbox**, you will see a *search box*, a *search button*, and a toolbox menu (also known as a *hamburger menu*), as shown in the following figure:

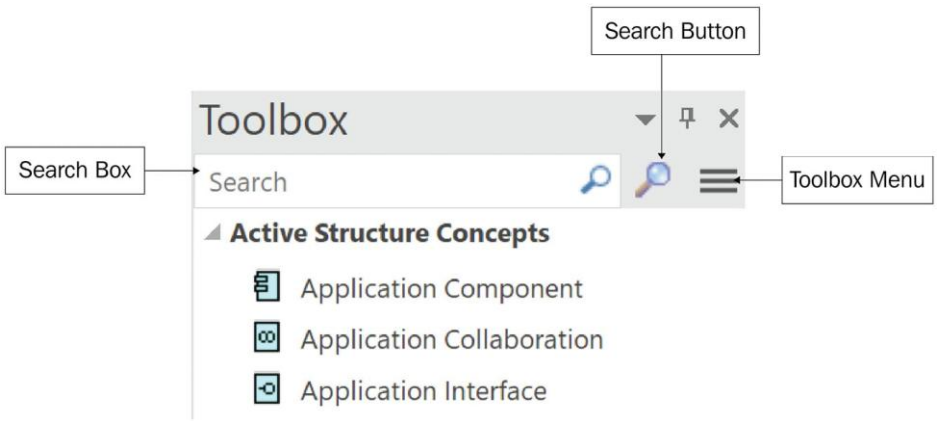

Figure – The Toolbox menu items

The search box allows you to search within the active toolbox, the search button allows you to search within all Sparx toolboxes, and the toolbox menu allows you to change the toolbox and the **Perspective**.

Sparx groups toolboxes that serve a similar purpose in what it calls a Perspective, so there is a Perspective for ArchiMate® 3.1 that contains all the toolboxes under the ArchiMate® 3.1 specification, there is a Perspective for UML 2.5, a Perspective for Strategy, and so on. Use the toolbox menu whenever you need to use an element from a different toolbox within the same Perspective, or to change to another Perspective altogether.

To activate the Business toolbox, click the toolbox menu > **ArchiMate 3.1** > **Business**, where you can see all the business architecture elements.

Placing the Business Actors on the diagram

While reading the Use Case documentation, you will have realized that three main Business Actors will be using Tracking App: truck drivers, shipping managers, and business partners.

To add the Business Actors to the diagram, make sure you do the following:

1. With the Business toolbox open, drag three **Business Actor** elements from the **Toolbox**, and drop them onto the diagram above the application service.
2. Rename the three Actors as `Truck Driver`, `Shipping Manager`, and `Business Partner`.
3. Add a short description for each Actor in the designated **Notes** field.

With more elements added to the diagram, you may realize that you need to move elements to a different location to make space for more elements to be added. Let's look at how to move the elements on a diagram.

If you need some space between the diagram label and the elements that you have placed, you can follow the same selection techniques that you are already familiar with in other tools, such as the following:

- You can *click and hold* your left mouse button to draw a selection rectangle. Every element that falls fully or partially within the selection rectangle will be selected.
- You can hold the *Ctrl* key on the keyboard and *click* on elements to select or unselect them.
- You can press *Ctrl* + *A* to select all elements on the diagram and then use the *Ctrl* + *click* technique to unselect the unwanted ones.

All these techniques are pretty standard and available in every modeling tool (and even in Windows File Explorer), possibly with slight variations, so there is nothing new to say here except to remind you that the same old techniques that you may know already work in Sparx too.

After selecting the elements that you want to move, click and hold any of them, drag them to where you want to move them, and release. Alternatively, you can move a group of selected elements by one point at a time using *Shift* + arrow keys to move the selection to the left, right, up, or down accordingly. This is a very helpful technique to slowly move objects for more accurate positioning.

Defining the relationships to Actors

It is important to keep in mind that everything is connected in the enterprise. There should be no single element in the entire enterprise repository that exists by itself and has no connections to at least one other element. Defining the proper relationships will help us to identify and analyze the impact of making changes.

The relationship between an application service and a Business Actor is of the **serving** type. The serving relationship looks like a straight arrow with a solid line. The arrow base is connected to the element that is providing the service, and the arrowhead points to the element that receives the service.

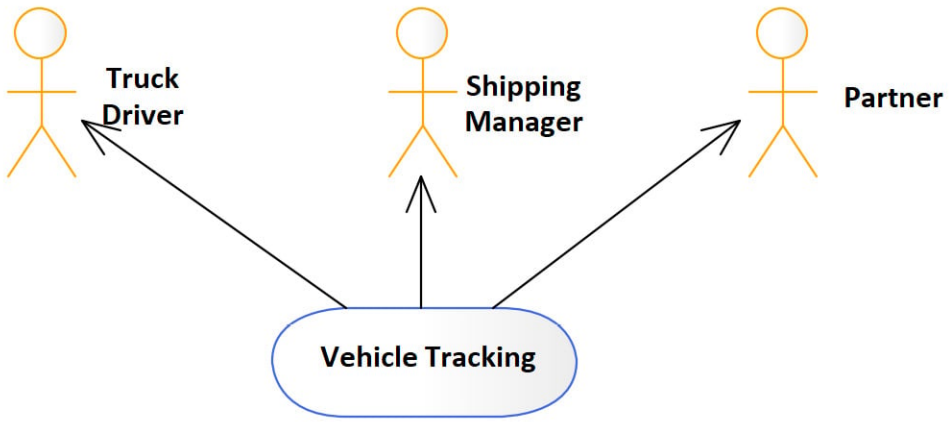

Figure – An application service serving Business Actors

Since you know how the relationship looks, it will be easy to find it in the **Toolbox** and create it between the elements:

1. Locate the serving relationship in the **Toolbox**, click and hold the application service, move to the first Actor, and release.

2. Repeat the same for the other two Actors.
3. Another way to connect is to click on the source element (the application service) to highlight it, use the small action arrow that appears to the side, hold it, and release it above the target element (the Actor), and then select **Serving** from the context menu.

Just like we did with other elements that we have placed on the diagram, we need to style the Business Actors.

Styling the Business Actors

For styling business architecture elements such as Business Actors, we will use the same font size that we have used for the application elements. However, we will use a different color for the border to visually differentiate elements that belong to different architecture layers. After styling one Business Actor element, we will save the style and apply it to the remaining Business Actors, as you will see in the following steps:

1. Select one of the Business Actors and adjust its font size to **10**, set the fill color to *white* (RGB 255, 255, 255), and set the border color to *orange* (RGB 255, 165, 0).
2. Save this style for future reuse as `Biz Arch` so you can apply it to other business architecture-layer elements without repeating the styling steps every time.

Now we have the style saved, we can apply it to any other element. Reusing styles is a very important factor in maintaining the consistency of diagrams within a single repository, especially when there are multiple users contributing to the content.

3. Highlight the other Business Actors on the diagram either together as a group or one by one and apply the saved **Biz Arch** style to them.
4. Optionally, you can change the notation of the Business Actors to the borderless notation.
5. Save your work by pressing *Ctrl* + *S*.

If you have followed the steps so far, your diagram will look like the following:

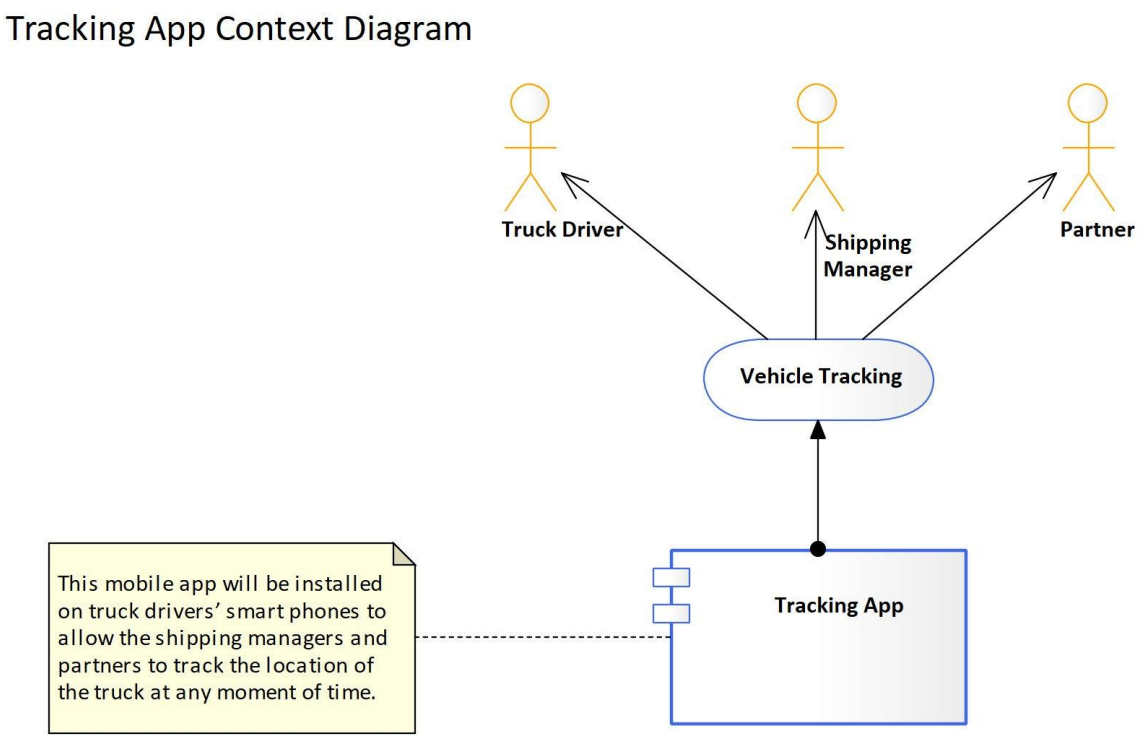

Figure – Tracking App context diagram so far

The diagram is shaping up and is telling us more about Tracking App now. The project stakeholders may agree or disagree with what you are proposing but this is the main objective of conceptual diagrams. Our goal is to have all stakeholders on the same page of understanding, and opening doors for questions and answers always helps with this goal.

Identifying other dependent elements

With the identified services of Tracking App, there is not much information to exchange with other applications. Tracking App needs to send the phone's current location at specified time periods to the Trading Web application.

The Trading Web application is the main business application that internal users and external partners use to make orders, get item information, make payments, and access many other core online services. Identifying all these services will add great value to your enterprise repository, because the Trading Web application represents the information technology backbone of the ABC Trading business. The best thing you can do in this case is to add a new user story to the artifacts backlog and discuss it with the product owner. For now, all we know is that our enterprise has an

application component called Trading Web and it will exchange data with Tracking App.

Adding the Trading Web application component

We know that we have not created the Trading Web application component before, so we will create a new application component from the **Toolbox**:

1. Drag an application component from the **Toolbox** and drop it on the diagram to the side of **Tracking App**.
2. Rename it to `Trading Web`, style it by using the **App Arch** style, and add a description in the **Notes** area as desired.
3. Add descriptive text to the **Notes** field of the Trading Web component.

Tracking App will send the location of the truck (or vehicle in general) to the Trading Web app through the Vehicle Tracking service at every specified period. In other words, *Tracking App* will be *serving Trading Web* with *vehicle location data*.

4. Use the Toolbox or the arrow action button to create a serving relationship going from the Vehicle Tracking service to the Trading Web application component.

Styling the Trading Web application component

Trading Web is an application component, so you can reuse the previously saved **App Arch** style as we have done earlier, or use this quick copy/paste method to apply a style from one element to another element on the spot:

1. Highlight the **Vehicle Tracking** application service.
2. Locate the **Layout** > **Style** > **Pickup** toolbar button, which has an *eye drop* icon. Click it, and the style will be copied to the clipboard and will remain there until you copy another style or close Sparx.
3. Now, highlight **Trading Web** and click the **Layout** > **Style** > **Apply** button – which has a *brush* icon – to apply the style that is in memory on the selected element.
4. Either keep the default rectangular notation or use the borderless one.

So far, we have learned two ways for reusing styles, and each has its own benefits:

- **Saving the style**: This is useful if you will use the same style over and over in different places within the repository for a long period of time, so it is more for *permanent* reusability.
- **Copy/paste the style**: This is useful for quickly applying a style of one element to another, and if you want it to stay in memory for a short period of time, so it is more for *temporary* reusability.

Save a style only if you are sure that you will reuse it later because you do not want to *pollute* the styles list with unused elements. If you are not sure that you will reuse the style again, it is more efficient and cleaner for the repository to copy/paste the style from another styled element.

This is what the diagram will look like at this point:

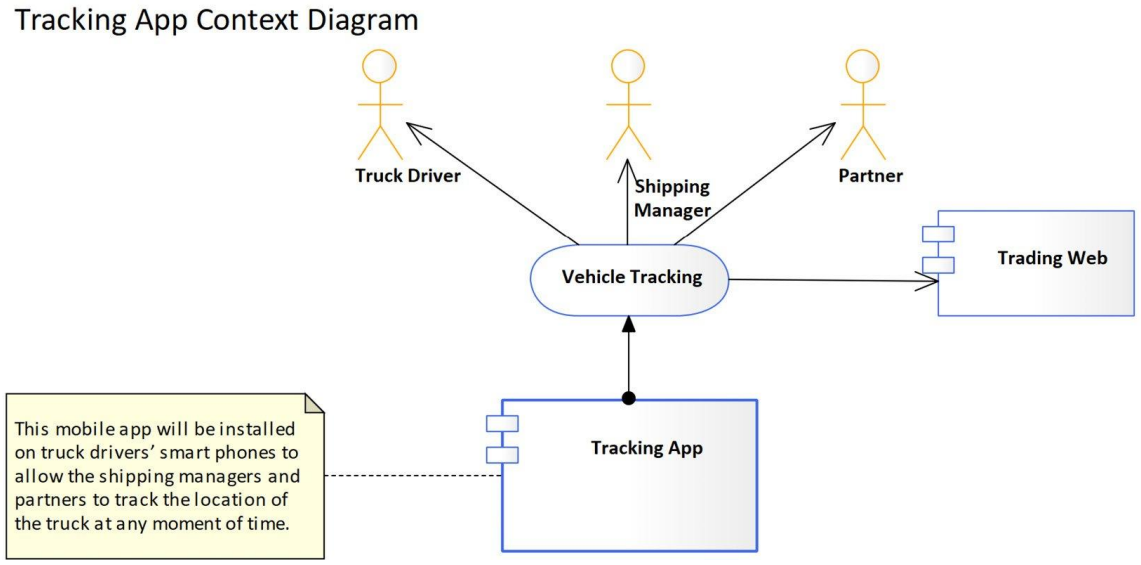

Figure – The Tracking App context diagram up to this point

Adding the supporting technology services

The location of the truck will be determined by the location of the phone that runs the application. This is a service that is already provided by the phone's operating system, so all we need to do is to use it. At this level of abstraction, we only need to mention that we will use the location service provided by the operating system. This type of service is known as **technology services**.

Technology services are like application services since they both describe the exposed behavior of a structure element. Technology services are *business*

neutral while application services have some *business logic* that is related to the business domain of the organization. The Vehicle Tracking service, for example, is a service that is of use only to a business that has requirements to track vehicles, so it is classified as an application service. The Device Location service, on the other hand, is a service provided by the phone regardless of what applications use it. It can be used by a navigation application, a marketing application, a social application, or even a game, so it is business neutral and therefore classified as a technology service.

We need to add a new Technology Service element to the diagram, but since the currently active toolbox is the Application architecture toolbox, we need to change it to the **Technology** architecture toolbox. Follow these steps to add a new Technology Service element to the diagram:

1. Click on the hamburger action menu, and then select the **ArchiMate 3.1** > **Technology** architecture. This will activate the ArchiMate® Technology architecture toolbox.
2. Locate the Technology Service element in the Toolbox, drag it, and drop it on the diagram below the Application Component element.
3. Rename the service to `Device Location`, change the font size to **10**, change the fill color to *white* (RGB 255, 255, 255), and change the border color to *dark green* (RGB 0, 100, 0).
4. Save the style as a new `Tech Arch` style for future reusability.
5. Add descriptive text to the **Notes** field describing the technology service.
6. Use the borderless notation if you prefer, or keep the default rectangular notation.
7. The Device Location service will provide *location data* to the application component. In other words, the *Device Location service* is *serving* the *Tracking App component* with the *phone's location data*. Therefore, create a relationship of the *serving* type going from the technology service to the application component.
8. Save the changes by pressing *Ctrl* + *S*.

Adding the sent data

The operating system will provide the device's location in the form of data. We do not know a lot about the structure of the data at this time, but we know that data will be provided from the Device Location service to the Tracking App component, so we

need to model that. There are multiple ways to model how data flows from one element to another.

1. Open the **ArchiMate 3.1** > **Application** toolbox and locate the **Data Object** element.
2. Drag the **Data Object** element and drop it onto the diagram near the **Device Location** service.
3. Change the data object's name to `Device Location Data` and style it as **App Arch**.
4. Add a proper description to the **Notes** field describing the data object.

Notice that there is only one notation for data objects so you will not see the option to change it in the context menu. Now we need to indicate that the data object will be provided or *served* by the Device Location technology service to the Tracking App component.

5. Click on the serving relationship and you will see a small action arrow that appears to the right of the relationship.
6. Hold this arrow, drag it to Device Location Data, and select **Associated with**. This creates an association relationship between the serving relationship and the data object and tells the user what data will be served.
7. Press *Ctrl* + *S* to save.

If you were following properly, then your diagram will look similar to the following one:

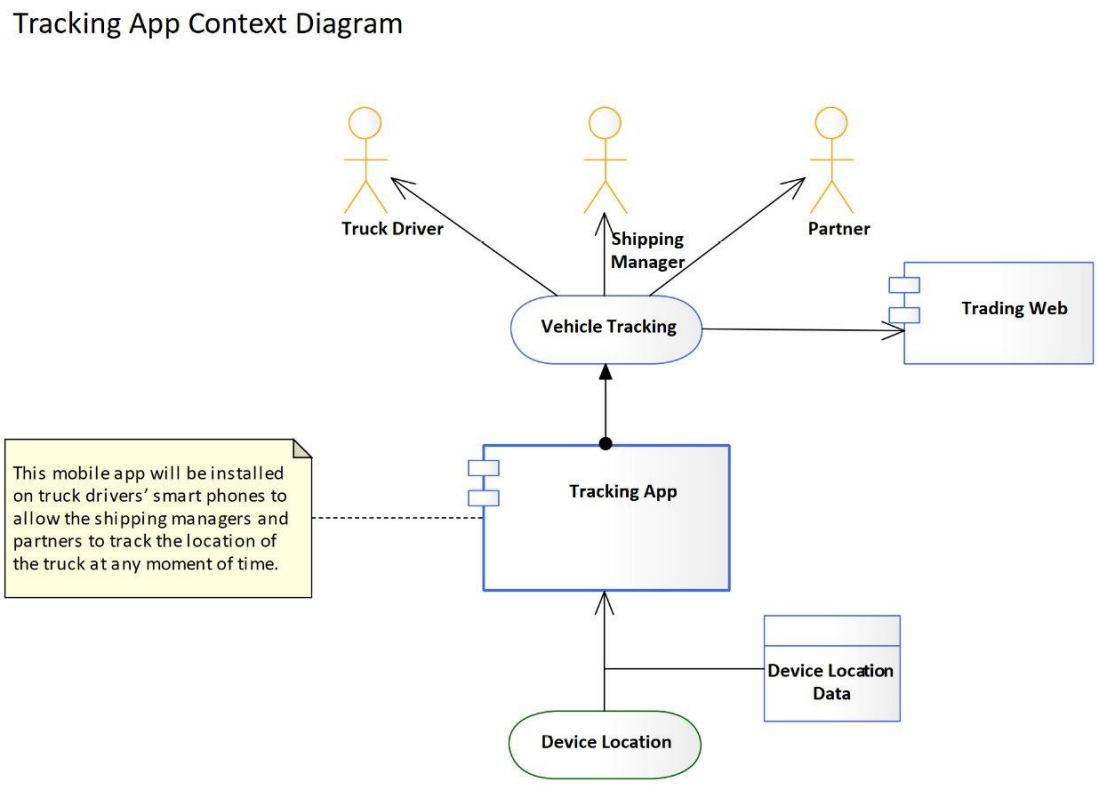

Figure – The Tracking App context diagram in its final state

The diagram is mature enough at this point to be shared with stakeholders. If a stakeholder wants to deliver a presentation on the application, having this diagram can answer a lot of the questions in a very simple yet well-documented way. The diagram gives enough information at a conceptual level; you must keep your context diagrams at this level of abstraction.

The application context diagram is meant to build a common understanding of a specific application component. By looking at it, any user at any level of experience will have the same understanding of what the application component is or is supposed to be. There is no technical jargon, no abbreviations, no third-party products, and no specific implementation enforcement. It is meant to have every stakeholder on the same page.