2.2 Modeling tutorial

This section will guide you through a modeling tutorial using the Enterprise Architecture Analysis Tool. We will introduce the classes and relationships of a metamodel capable of analyzing several different quality attributes, including service availability, modifiability, and cost. In this first subsection, the attributes are not explained, these will be presented later in the book (cf. chapter 3). The metamodel used here is partially based on ArchiMate and contains classes from the business layer, the application layer, and the infrastructure layer.

**Application layer** An *application service* is defined as a unit of functionality that a system exposes to its environment and that displays automated behavior. An application service could be Playing a game of chess with a user, or Printing a shopping list or Calculating a ballistic trajectory. The important thing is that the service provides something meaningful to the user.

An application service is exposed to the environment, but it is realized by an *application function*. For the chess game, an application function could be Calculating the next move. To calculate the next move is not in itself useful to the chess player, but it is a function that will contribute to something useful, that is to an application service.

An *application component* is a modular, deployable, and replaceable part of a software system. It is the entity that performs application functions. It could for instance be the Chess software that plays the chess game with the user.

An *application collaboration* is used in order to model what application components that are acting together to perform a collective behavior. It could for instance be the collaboration between the Chess software and a Web component, in order to find other players to compete with or to be able to post your game results online.

**Business layer** Just as the chess player uses the chess application, a designer in the automotive industry may use a design application to sketch on a car. The *business process* is the Designing of the vehicle, and processes may use application services, so here is an important link between the business and the IT. Other processes in a car company could be Producing the vehicle and Shipping the vehicle.

Business processes are performed by *roles*. In this case, the Designer is a role. Other roles could be Chief Production Officer and Machine Operator.

The customer, however, sees nothing of the design, production and shipping of a car. The customers interact with the company through *business services*. An example of a business service is Car selling.

**Infrastructure layer** Moving to the bottom part of the metamodel, the application layer is mirrored by an underlying infrastructure layer. In the same

---

3 Download at http://www.ics.kth.se/eaat
manner as business processes may use application services, application functions may use *infrastructure services*. Examples of infrastructure services are Messaging, Data management and Printing. Just as in the case of applications, infrastructure services are realized by *infrastructure functions*. These are in turn performed by *nodes*. A node could be an IBM Mainframe running z/OS, or a PC running Microsoft Windows.

In all, the metamodel is composed of ten classes, cf. Figure 16, and this metamodel will allow us to analyze system qualities such as service availability, modifiability, and cost.

![Figure 16: Tutorial metamodel example.](image)

The tutorial is composed of eight steps that will guide you through an enterprise architecture modeling and analysis example. The background is a case at a fictive energy company called ACME, which is about to start a project to improve the business process of analyzing automatic meter reading data. The architect at ACME has a proposal for the CIO that includes new applications and infrastructure supporting the process. The scenario also includes the hiring of a new person in charge of the meter data analysis. The CIO has expressed
that the most important aspects to consider, besides getting the right functionality, is to have high availability and that the solution is easy to change in the future in case new requirements occur. Also the company has a tight budget for these kinds of investments, thus the total cost is also a major factor in this case.

Outline

1. Add a first infrastructure layer with three classes and evaluate the availability
2. Add a second infrastructure layer with three additional classes and evaluate its availability
3. Add an application function, connect it to the two infrastructure services, and evaluate its availability
4. Add an application component, connect it to the application function, and see how it affects the availability
5. Add an application collaboration with coupling evidence
6. Add an application service and evaluate the modifiability
7. Add a business layer and evaluate the business process availability
8. Add cost evidence to the model and evaluate the total business process cost

The following paragraphs will detail the modeling steps.

Step one: Add a first infrastructure layer

- Add an Infrastructure Function
- Name it Local meter reading
- Press the calculate button
- Consider the Availability results (the availability of the Local meter reading function should be a normal distribution centered around 0.95)
- Add a Node
- Name it Local meter readers
- Connect the two classes
- Add 0.992 (99.2 %) as evidence to the attribute Availability of the Local meter readers node
- Press the calculate button
- Consider the Availability results (the availability of the Local meter reading function should now be 0.992 instead)

- Add an Infrastructure Service and name it Aggregated meter readings

- Connect the Aggregated meter reading service and the Local meter reading function. The Aggregated meter readings service is Realized by the Local meter reading function.

- Press calculate again and see how the availability of the node, that first had an effect on the function, now also has an effect on the service.

- Save the model regularly.

**Step two: Add a second infrastructure layer**

- Add a second Node and name it Database, add a second Infrastructure function and name it Data transmission, and add a second Infrastructure service and name it Data transfer.

- Connect the Node and the Function.

- Connect the function and the service with a RealizeAND association so the Data transfer service is Realized by the Data transmission function.

- Add 0.991 (99.1 %) as evidence to the attribute Availability of the Database node.

- Press the calculate button and consider the Availability results (the availability of the Data transfer service should be 0.991)

- Save the model regularly.

**Step three: Calculate the availability of an application function**

- Add an Application Function and name it Data collection

- Connect the Data collection function with the Aggregated meter readings and the Data transfer services, since the Data collection function uses both services the relationship should be of type AND. The relationships should be DataCollection.useAND.LocalMeterReading and DataCollection.useAND.DataTransmission.

- Press the calculate button and consider the Availability of the Data collection function, which should be 0.983 (98.3 %)

- See the application function availability viewpoint of the model in Figure 17.

- Save the model regularly.
Figure 17: The application function availability viewpoint of the tutorial model example.

**Step four: Add an application component**

- Add an Application Component and name it AMR master
- Connect the AMR master with the Data collection function
- Press calculate and consider the Availability result of the Data collection, the result should now read 93.4%
- Add 0.995 as evidence to the attribute Availability of the AMR master component
- Press calculate and consider the Availability result of the Data collection function again, the result should now read 97.8%
- Save the model regularly.

**Step five: Add application collaboration**

- Add a second Application component and name it Business intelligence.
- Add an Application collaboration class and name it Data exchange.
- Connect the two Application components through the Application collaboration class.
- Add a second Application function and name it Data compiler.
- Connect the AMR master and the Business intelligence components with the Data compiler function.
- Add Size 430000 and Gearing factor 53 to the AMR master, as well as Size 75000 and Gearing factor 55 to the Business intelligence component.

- Press calculate and consider the External couplings attributes of the two components, all four should now be 1 (since we have one collaboration of unknown kind).

- Add 1 Content coupling, 2 Common couplings, and 11 Data couplings to the Data exchange collaboration between the two components.

- Press calculate again and consider the External coupling attributes of the two components, all four should now be 5.9 instead.

- Save the model regularly.

**Step six: Calculate application modifiability**

- Add an Application service and name it Compiled meter data.

- Connect the Compiled meter data service with the two Application functions, since the service is Realized by both functions the relationship should be of type AND.

- Add a baseline Gearing factor of 53 to the Compiled meter data service.

- Press calculate and consider the Modifiability of the service, which should be 3. This is a rather low value indicating that the service will be difficult/costly to change in the future, mainly due to large source code and the tight coupling between the components realizing the service.

- See the application service modifiability viewpoint of the model in Figure 18.

- Save the model regularly.

**Step seven: Add a business layer**

- Add a Business process and name it Analyze meter data.

- The Analyze meter data process Uses the Compiled meter data service. Add this relationship (since there is only one user, either userAND or UserOR) can be selected.

- Add a Role and name it Meter data analyzer.

- Connect the Meter data analyzer role with the Analyze meter data process.

- Add 99 % Availability as evidence for the Role and 99.7 % Availability for the Business intelligence component.
Figure 18: The application service modifiability viewpoint of the tutorial model example.

- Press calculate and consider the Availability for the Business process Analyze meter data, which should be 96%. This is a rather low value for a business process and it is mainly due to the high dependence of the manual work of the Meter data analyzer role.

- Save the model regularly.

**Step eight: Calculate the business process cost**

- Add the Initial and Yearly costs of the Nodes, Components, and Role as described in Table 1.

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>Initial Cost</th>
<th>Yearly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local meter readers</td>
<td>Node</td>
<td>1 000 000</td>
<td>100 000</td>
</tr>
<tr>
<td>Database</td>
<td>Node</td>
<td>50 000</td>
<td>10 000</td>
</tr>
<tr>
<td>AMR master</td>
<td>App.Comp.</td>
<td>300 000</td>
<td>45 000</td>
</tr>
<tr>
<td>Business intelligence</td>
<td>App.Comp.</td>
<td>450 000</td>
<td>25 000</td>
</tr>
<tr>
<td>Meter data analyzer</td>
<td>Role</td>
<td>500 000</td>
<td>1 000 000</td>
</tr>
</tbody>
</table>
Press calculate and consider the cost of the business process Analyze meter data, which should be 3,480,000 SEK. This cost is based on the assumption that the role, components, and nodes are new. Once these have been in place for some time the initial costs will be written-off and the total cost of the service will only be based on the yearly costs (this has not been implemented in the metamodel yet). Also, in this small example architecture the business process does not share any components, nodes, or the role with others. If this would have been the case the costs would also have been shared, thus a smaller economic burden would have fallen on this particular service.

- Save the model regularly.
- The complete model can now be seen in Figure 19.

The CIO is satisfied with the functionality of the solution and the cost, but there is a wish to find a scenario with higher modifiability and at least better
availability in the infrastructure and application layers. Therefore, the architect needs to find a second scenario that the CIO can compare this solution with before a decision can be made.

This chapter of the book has focused on explaining what a metamodel is and how to model architectures. The next chapter will present a metamodel for enterprise architecture analysis of multiple attributes including modifiability, availability, cost, interoperability, application usage, and data accuracy.