Appendix C

MDE Tools User Manual

In this appendix, a user manual for INNoVaServ is presented - the modeling environment for service design built in this doctoral thesis, whose details are described throughout Chapter 3.

C.0.1 Creating a New Project

To create a new service design project in INNoVaServ, select the "File" option in the top options bar of Eclipse. Next, click on "New" and then "Other...". In the panel that appears, choose the "Modeling Project" option from the Sirius folder, as shown in Fig. B 4.

After clicking "Next", assign a name to the project and click "Finish". The project should then appear in the Eclipse project explorer and be ready to host new models.

C.0.2 Working with Models

To create *SmaC*, e^{3}value and *SmaCQA* models, right-click on the project where they will be located and select "New" and then "Other...". A panel similar to Fig. B 4 will appear, where you can search for the desired model or expand the INNoVaServ Toolkit folder.

C.0.2.1 Creating SmaC Models

If you want to develop textual smart contract models, it's important to select the element whose name ends in "Diagram", as the wizard will create both the model and the corresponding diagram.

After clicking "Next", you can modify the default name of the textual smart contract model. It's important to select the "File" root element from the "Model Object" dropdown, as shown in Fig. C.1, once the creation wizard displays it.

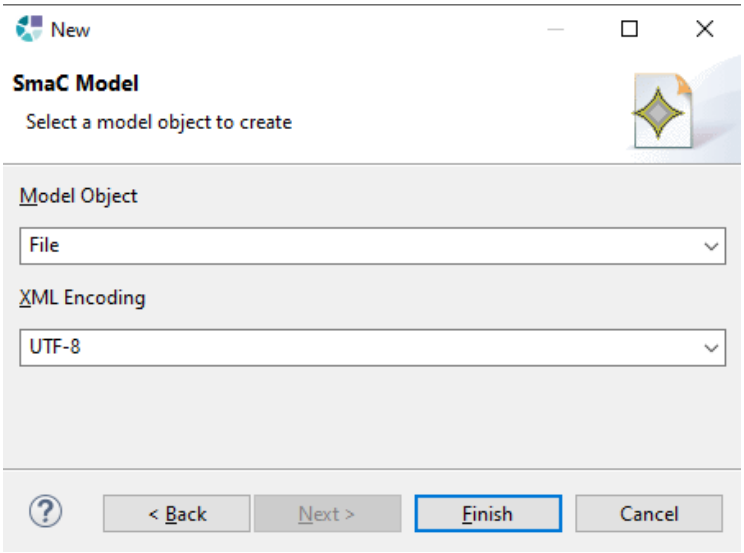

Figure C.1: Selection of the root element for the textual smart contract model

After clicking "Finish", a .sce file will be created in the corresponding project. Additionally, the diagram will open automatically in the Eclipse editor.

C.0.2.2 Definition of a SmaC Model

When defining a textual smart contract model with the *SmaC* environment, the tool establishes a pattern for defining the elements that compose it in order to facilitate coding and model readability. Therefore, the order for defining the elements of the model is as follows:

1. Definition of the version (mandatory).
2. Definition of imports (optional).
3. Definition of libraries (optional).
4. Definition of interfaces (optional).
5. Definition of abstract contracts (optional).
6. Definition of a contract (at least 1).

6.1 Definition of objects, attributes, enums, and structs (optional).

6.2 Definition of constructors (optional).

6.3 Definition of modifiers (optional).

6.4 Definition of events (optional).

6.5 Definition of functions (optional).

If the above order for defining the elements is not followed, the editor will display an error message indicating a definition error in the textual smart contract model.

C.0.2.3 Transformation of textual models generated by SmaC to visual models SmaCly

To generate a visual block model in XML format from a textual model of smart contracts coded with *SmaC*, you must click on the button located in the toolbar shown in Fig. C.2.

Figure C.2: Transformation option from SmaC space textual model to SmaCly space block model

Once the process, which does not require user intervention, is completed, a file with the name of the textual model in XML format will be generated. This file will be located in a folder created by the environment with the name of " ".

C.0.2.4 Transformation of visual models generated by SmaCly to textual models SmaC

To generate a textual model from a visual model of smart contracts coded with *SmaCly*, you must select the visual model within the project where it is hosted and left-click on the button located in the toolbar shown in Fig. C.3.

Figure C.3: Transformation option from SmaCly block model to SmaC textual model

C.0.2.5 Creating SmaCQA Models

To develop textual models of questions and answers about e^{3}value models, the New and Other... option must be selected by right-clicking on the project where the model will be located. Subsequently, the *SmaCQA* Model option must be selected through the visual assistant. After clicking on Next, it is possible to modify the default name of the model. In the next screen, the Model element must be selected as the Model Object, as shown in Fig. C.4:

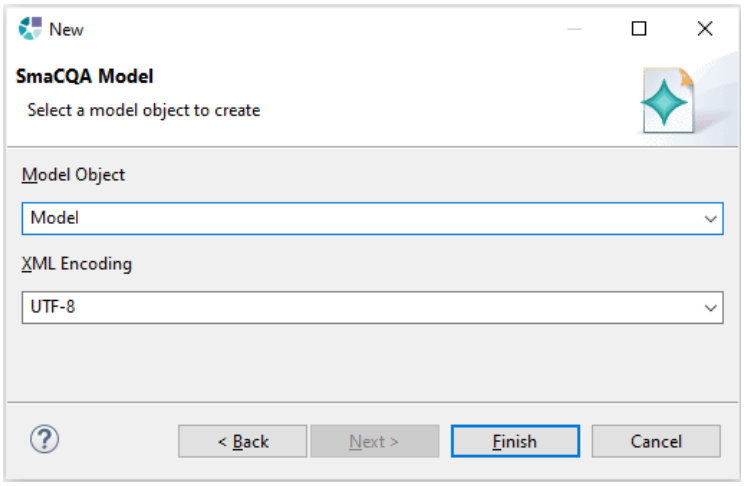

Figure C.4: Root selection of the textual model of questions and answers

After clicking on Next, the default name of the textual model of questions and answers for the e^{3}value model can be modified. Clicking on Finish will create a file with the extension .qa in the corresponding project. Additionally, the diagram will automatically open in the Eclipse editor.

C.0.2.6 Exporting the Model in HTML Format

To export the information collected in the textual question-and-answer model about an e^{3}value model to HTML format, click on the button in the IDE toolbar shown in Fig. C.5:

Figure C.5: Option for exporting the model to HTML format

After this, the environment will generate a file with an HTML extension, which can be viewed by any browser. The file contains a visual representation in the form

of a table, as shown in Fig. C.6, with the information collected from the textual question-and-answer model.

VALUE EXCHANGE: MaterialsProducer –> Developer

QUESTION	ASPECT	ANSWER
1.5 If the object of value negotiated in the value exchange is a tangible entity that can be represented as a digital entity (not a token). What are the properties of that object?	"Data"	YES
1.1 If the exchange of value is subject to a duration of time. What would this be?(indicated in minutes,days,weeks or years)	"Data"	150 minutes
1.4 Are the same conditions always maintained when exchanging value?	"Data"	Yes
2.2 What is the name of the tax?	"Legal"	VAT
2.2.1 Who collects the tax?	"Legal"	Goverment

VALUE EXCHANGE: Investor –> Developer

QUESTION	ASPECT	ANSWER
1.4 Are the same conditions always maintained when exchanging value?	"Data"	Yes
2.2 What is the name of the tax?	"Legal"	VAT
2.2.1 Who collects the tax?	"Legal"	Goverment
3.1 Which would be the minimum amount if necessary in this exchange?	"Economy"	3 ether

VALUE EXCHANGE: Developer –> Investor

QUESTION	ASPECT	ANSWER
1.5 If the object of value traded on the value exchange is a digital token. What are the properties of said token?	"Data"	Token Name: Energy_Pledge; Token Symbol: ENY; Token Decimals: 18; Token Supply: 100000000; Mintable?: Yes; Burnable?: Yes
1.2 If the exchange of value could only take place after a certain time. What would this be?(indicated in minutes,days,weeks or years)	"Data"	3 days
1.4 Are the same conditions always maintained when exchanging value?	"Data"	Yes

Figure C.6: View of the model representation in HTML format

C.0.2.7 Creation of e^{3}value models

To create a visual model of the e^{3}value notation, you must select the option New and Other... by right-clicking on the project where the model will be located. Next, you must select the option, within the INNoVaServ Toolkit folder, using the visual assistant. When Next is clicked, it is possible to modify the default name of the model. Then, Next should be clicked again, and in the following screen, select the "Diagram" element as the Model Object, as shown in Fig. C.7.

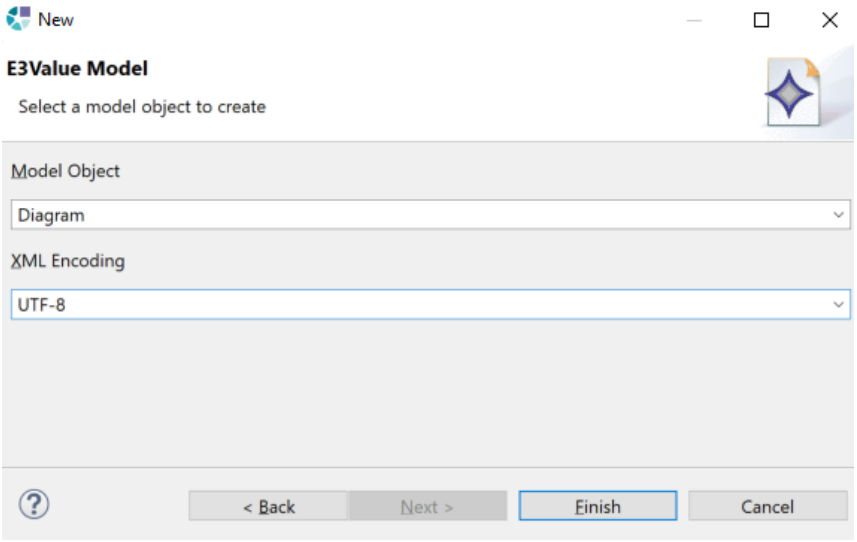

Figure C.7: Selection of the e^{3}value model root

Once the above process is completed, the IDE will open a tab to offer the user a tree-like editor for editing the newly created model instance. To use the graphical editor defined by the authors, we must right-click on the root of the model with the "e3value" extension and select the "New Representation" option and then the "new E3Value" option as shown in Fig. C.8:

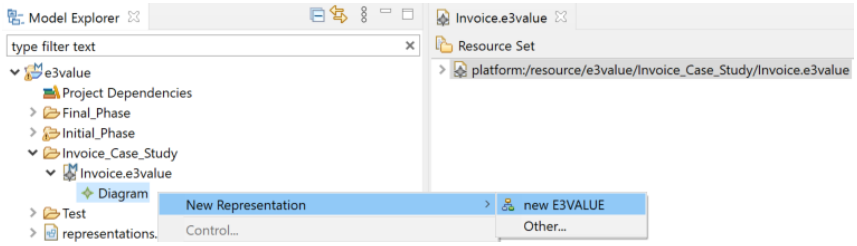

Figure C.8: Selection of the e^{3}value model representation

Subsequently, a tab will open as a canvas along with the toolbar for editing the e^{3}value model as shown in Fig. C.9:

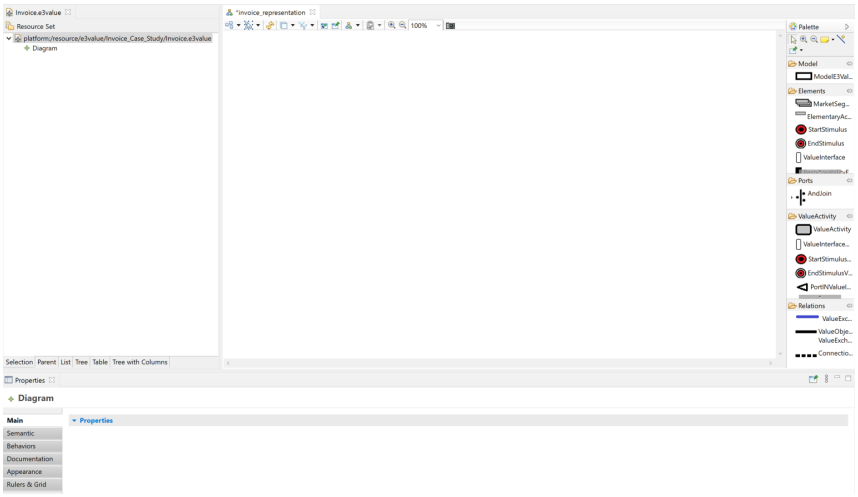

Figure C.9: e^{3}value model editor

C.0.2.8 Transformation from e^{3}value visual models to SmaC textual models

To generate a textual smart contract model from an e^{3}value model and its associated qa model, both models must be selected by pressing the CTRL button, and then left-clicking on the button located in the toolbar shown in Fig. **??**.

After performing this action, the editor will offer the user a semi-automatic process wizard as a guide. The wizard will initially show the models it receives as data sources to generate a textual smart contract model from them, as seen in Fig. C.10:

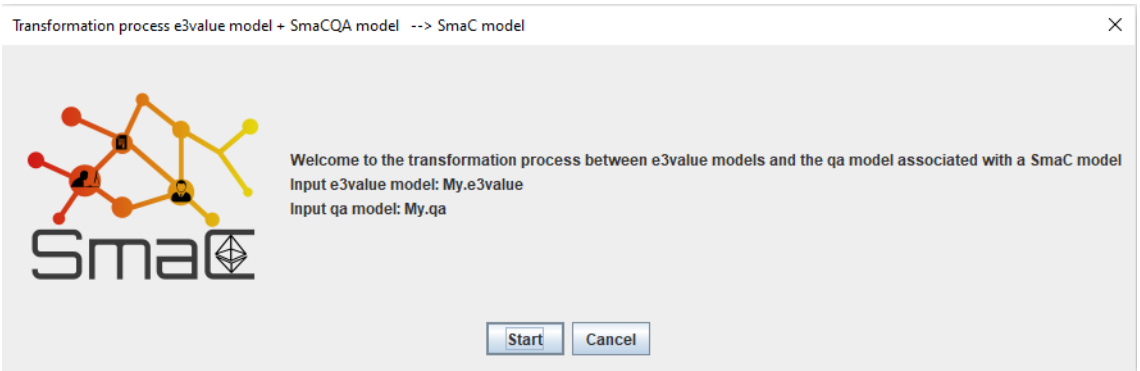

Figure C.10: Start panel of transformation process

To start the process, the user must left-click on the "Start" button. The information collected in both models will be automatically processed, generating elements

of the future smart contract. After finishing, the assistant will begin to display a series of panels to the user, requesting both mandatory information to complete the smart contract, such as the compiler version (see Fig. C.11), as well as optional information such as expanding the predefined data type User or Company.

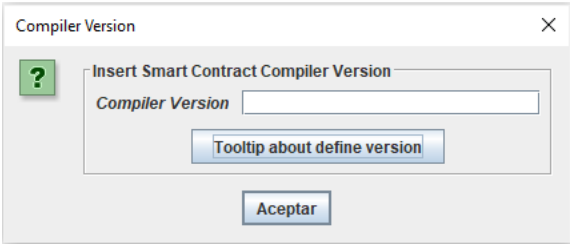

Figure C.11: Compiler version requirement

During the manual process of generating the new smart contract model, the assistant has tooltips about what type of information the user can include at a given moment in the process. A view of this functionality can be seen in Fig. C.12, where after clicking on the "Tooltip about define version" button shown in Fig. C.11, the assistant displays a help panel.

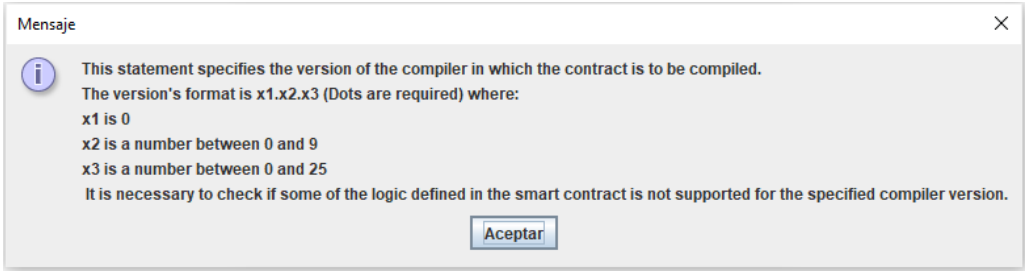

Figure C.12: Help on defining the compiler version

Next, for each pair of value interfaces in which value exchanges occur between two actors, the assistant will display a dialogue box for the user to specify the name of the smart contract to be defined, as shown in Fig. C.13:

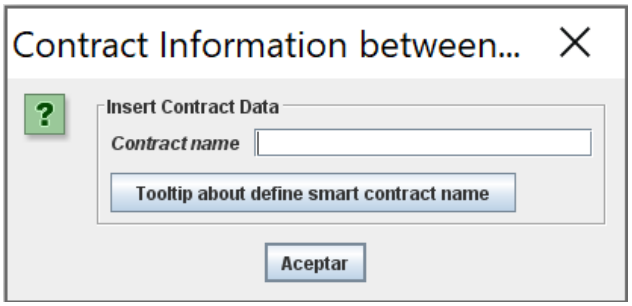

Figure C.13: Defining the name of the smart contract

The dialog box shown in Figure C.13 provides information to the user about the actors involved in value exchanges, in order to specify a name for the smart contract that reflects these actors. By clicking the "Tooltip about define smart contrac" button, the user can access a view of this information, as shown in Figure C.14:

Figure C.14: Information about the smart contract to be defined

Once a name has been assigned to the current smart contract being modeled, the transformation process assistant will prompt the user to specify the type of actor according to the custom data types (User & Company) in the textual grammar of *SmaC*, as shown in Figure C.15:

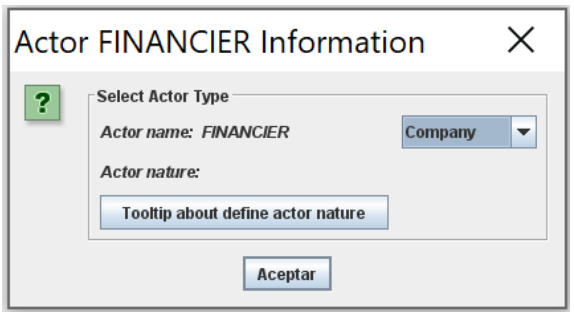

Figure C.15: Assignment of a struct data type to an identified actor

Furthermore, once the type assignment is accepted, the assistant will offer the user the possibility of adding more information to the data type that defines the actor, if necessary to complete the functionality of the smart contract.

Finally, for each identified value exchange between actors, the assistant provides the user with a series of fields to define each value exchange as a function in Solidity language. Therefore, it is necessary for the user to specify a name for this function, its visibility to determine if users can interact with the smart contract through this function, and whether a currency transfer will occur in the logic implemented by this function, as shown in Fig. C.16:

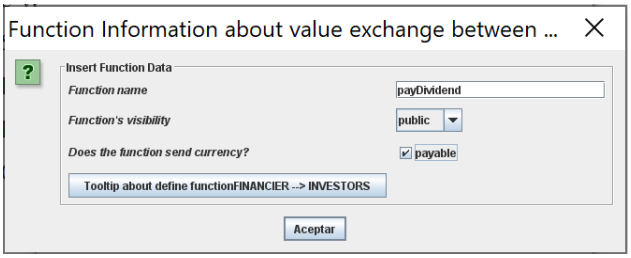

Figure C.16: Definition of a smart contract function

Once the manual process is completed, the assistant will indicate that the process has finished by generating the textual model of the smart contract in the SmaC-src-generated folder.

C.0.2.9 Generation of EMF models

To generate an EMF model from the information in a textual model, click on the button in the toolbar as shown in Fig. C.17.

Figure C.17: Generation of an EMF (xmi) model

Once the process is finished, which does not require user intervention, a model with an XMI extension will be generated with the information of the textual model that will be located in a folder created by the environment with the name "EMF-src-generated".