Language-Driven Engineering Tutorial 1

2 November 2018

1 Email Classification with Binary Decision Diagrams

In this first session you will model your own email classification service that aims to help organise the stream of emails in your inbox. The service will select particularly *urgent* emails to help you read the most important ones first.

1.1 Setup Your Mindset-Supporting Integrated Development Environment (mIDE) for Binary Decision Diagrams

We begin with Binary Decision Diagrams (BDDs) as the chosen mind-set to model our email classification service. With CINCO we have a powerful framework to generate the desired mIDE that allows application experts to think in their chosen mindset.

1. Start `cinco` and choose a **new workspace** for this tutorial and `Launch`.
2. Import the prepared CINCO project `info.scce.addlib.mide.booleanlogic.zip`:
   (a) In the main toolbar choose `File` > `Import...` > `General` > `Existing Projects into Workspace` > `Next`.
   (b) Choose the option `Select archive file` and `Browse...` your file system to import the prepared project archive `info.scce.addlib.mide.booleanlogic.zip`. Click `Finish` to finalise the import.
3. Expand the new project `info.scce.addlib.mide.booleanlogic` in the project explorer on the left side and make yourself familiar with the overall project structure. In particular, you should find the CINCO model files in the `as-model` folder.
4. Generate the CINCO product: right-click `DDTool.cpd` and choose `Generate Cinco Product`. The process to generate the new mIDE may take some seconds to minutes.

5. You can now start the new mIDE: select and right-click the main project `info.scce.addlib.mide.booleanlogic` and choose `Run As` > `Eclipse Application`. This will start a new instance of Eclipse: the mIDE for Binary Decision Diagrams.

6. In the new mIDE close the `Welcome` page and open the `DDTool Perspective`: in the main toolbar choose `Window` > `Perspective` > `Open Perspective` > `Others...` > `DDTool Perspective` > `Open`.

7. To help you focus on modelling a good email classification service, we provide and exemplary user application that helps you analyse your results. Import the prepared inbox simulation project `info.scce.addlib.mide.example.email.zip`:

   (a) In the main toolbar choose `File` > `Import...` > `General` > `Existing Projects into Workspace` > `Next`.

   (b) Choose the option `Select archive file` and `Browse...` your file system to import the prepared project archive `info.scce.addlib.mide.example.email.zip`. Click `Finish` to finalise the import.

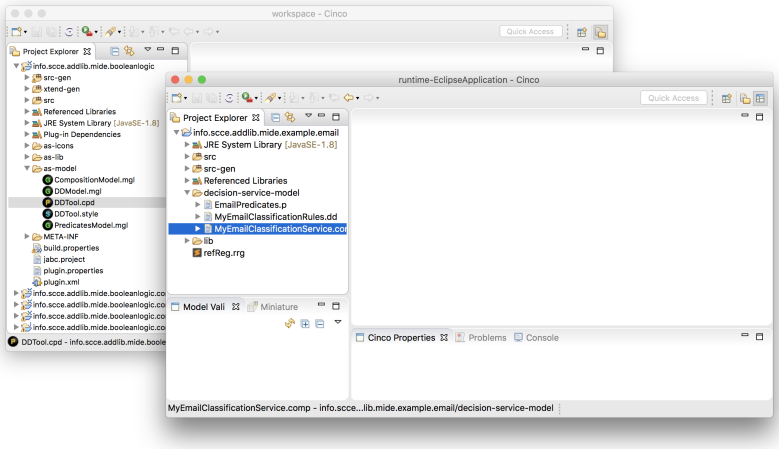

Figure 1: After successful project import you should see two instances of Eclipse, Cinco and the mIDE for Binary Decision Diagrams.

1.2 Model Your First Binary Decision Diagram

Make yourself familiar with the new mIDE for Binary Decision Diagrams and start modelling basic email classification services.

1. Find and open the file `MyEmailClassificationRules.dd`. Like all decision diagram models you will find it in the `decision-service-model` folder. This model contains two simple examples.
2. Find and open the file `EmailPredicates.p`. This file specifies a set of predicates that are used to characterise every email. Expand the predicates model file in the `Project Explorer` by clicking the little arrow to the left of its name. This will display a list of available predicates directly in the project explorer.
3. You can drag and drop these predicates onto the canvas of any decision diagram model. Find and open the file `MyEmailClassificationRules.dd` to make use of them in your first decision diagram.
4. Similarly, you can drag and drop `Function` and `Terminal` nodes from the `Palette` on the right side onto the canvas.
5. Hover over a predicate node and create an edge from it to another predicate node or to a terminal node. For every predicate node you can define a *then* and an *else* successor.
6. Finally, you can define the roots of decision diagrams with function nodes: give them a name in the `Cinco Properties` and create an edge to the leading predicate or terminal node. Function nodes represent Boolean functions of interest and are later used in composition models or directly generated to Java code.
7. Take your time and create some decision diagrams that help you classify emails based on the provided set of predicates. Do **not** touch the predicates model.
8. You can generate Java code from this model by clicking on the `G`-button in the Eclipse toolbar. The result will be saved in the `src-gen` folder of your project.

1.3 Model Your First Composition of Binary Decision Diagrams

Decision diagrams are great to model small decision services but as soon as the decision services becomes more complex and concern a variety of different aspects, scalability becomes an issue. Modularity can help: as individually modelled decision services simply represent Boolean functions, they can be combined using logical operators: conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). Make yourself familiar with the composition model and compose your previously modelled decision diagrams.

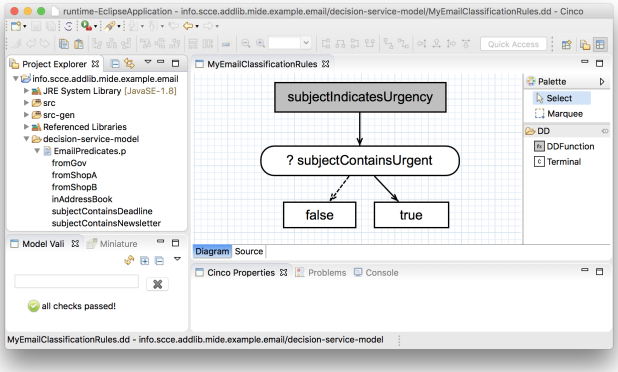

Figure 2: Exemplary decision diagram model as provided in the example project.

1. Open the composition model `MyEmailClassificationService.com p`. You will see a simple conjunction of two decision rules. The composition aims to identify *urgent* emails.

2. Expand the decision diagram model `MyEmailClassificationRules .dd` in the `Project Explorer` by clicking the little arrow to the left of its name. This will display a list of available functions directly in the project explorer.

3. Drag and drop previously created decision diagrams from the `Project Explorer` onto the canvas. They will appear as nodes so you can use them as operands of logical operations.

4. From the `Palette` on the right you can drag and drop operations onto the canvas.

5. To define the operands of any operation you can create edges from the operands to the operation. Operands can be referenced decision diagrams or other operations. In general you can distinguish between the left and the right operand but for Boolean algebra this would not make a difference thanks to commutativity.

6. Finally, you can define the roots of composition models with function nodes: give them a name in the `Cinco Properties` and create an edge from the leading operation. Function nodes represent exposed Boolean functions. They can be used in other composition models or directly generated to Java code.

7. Take your time and compose your decision diagrams to more accurately classify *urgent* emails.

8. You can generate Java code from this model by clicking on the G-button in the Eclipse toolbar. The result will be saved in the `src-gen` folder of your project.

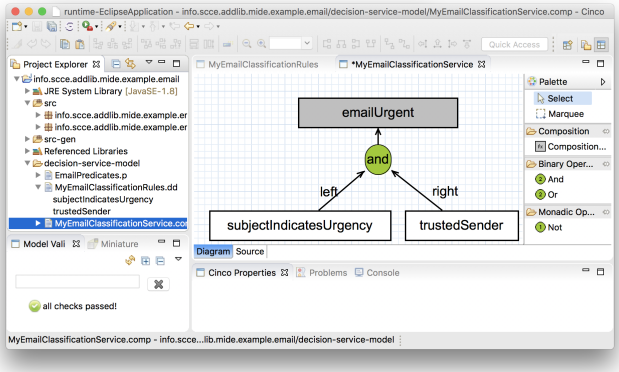

Figure 3: Exemplary composition model building the conjunction of subjectIndicatesUrgency and trustedSender.

1.4 Analyse Your Email Classification Service

When you finished modelling your decision service `MyEmailClassificationService.comp`, ensure to generate its Java implementation. The result will be saved in the `src-gen` folder of your project and can be used naturally in the Java project. This project implements an email simulation that lets you analyse your decision services.

1. Open the main Java file `Application.java` in `src/info.scce.addlib.mide.example.email`. This is where you may configure the application when needed. The application assumes a decision service with the name `MyEmailClassificationService`. Unless you have changed the name everything should work as it was set up. If the project contains any errors at this point, please ask for help.

2. We will now start the email simulation and use your email classification service. The simulation will generate random email predicate profiles and display some statistics about the results of your service (and others if you wish).

   (a) Find the file `Application.java` and right-click. Choose `Run As > Java Application`.

   (b) Select your own service and `Start` the simulation. See how your service classifies the emails and feel free to compare it to other services from the given options.

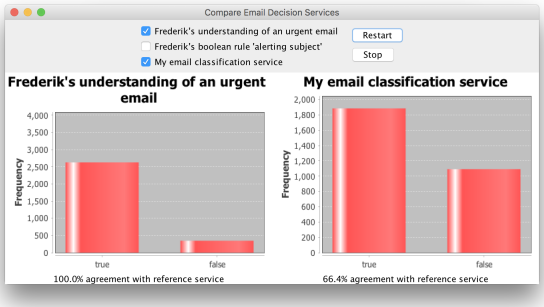

Figure 4: The email simulation using two services to classify emails

1.5 Improve your Decisions Service

Take your time to improve your email classification service, analyse it, and compare it to the provided example services.