European Commission | Horizon 2020 European Union funding for Research & Innovation

Cyber Security PPP: Addressing Advanced Cyber Security Threats and Threat Actors

Cyber Security Threats and Threat Actors Training - Assurance Driven Multi- Layer, end-to-end Simulation and Training

D3.2: CTTP Models and Programmes Specification Tool †

Abstract: This deliverable provides the guidelines for the CTTP Models and Programmes Specification Tool. It acts as the final outcome of the task "T3.1 – CTTP Language definition and Tool Support". As such, the deliverable presents the implementation approach, the CTTP tool and a deployment guide.

Contractual Date of Delivery	30/11/2019
Actual Date of Delivery	30/11/2019
Deliverable Security Class	Public
Editor	*Konstantinos Fysarakis, Michail Smyrlis (STS)*
Contributors	Fulvio Fratti (UMIL), Martin Kunc (CZNIC)
Quality Assurance	*George Hatzivasilis (FORTH),* *George Tsakirakis (ITML)*

† *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 786890.*

† *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 786890.*

The *THREAT-ARREST* Consortium

Foundation for Research and Technology – Hellas (FORTH)	Greece
SIMPLAN AG (SIMPLAN)	Germany
Sphynx Technology Solutions (STS)	Switzerland
Universita Degli Studi di Milano (UMIL)	Italy
ATOS Spain S.A. (ATOS)	Spain
IBM Israel – Science and Technology LTD (IBM)	Israel
Social Engineering Academy GMBH (SEA)	Germany
Information Technology for Market Leadership (ITML)	Greece
Bird & Bird LLP (B&B)	United Kingdom
Technische Universitaet Braunschweig (TUBS)	Germany
CZ.NIC, ZSPO (CZNIC)	Czech Republic
DANAOS Shipping Company LTD (DANAOS)	Cyprus
TUV HELLAS TUV NORD (TUV)	Greece
LIGHTSOURCE LAB LTD (LSE)	Ireland
Agenzia Regionale Strategica per la Salute ed il Sociale (ARESS)	Italy

Document Revisions & Quality Assurance

Internal Reviewers

1. *George Hatzivasilis (FORTH)*
2. *George Tsakirakis (ITML)*

Revisions

Version	Date	By	Overview
0.4	22/11/2019	Michail Smyrlis	Ready for internal review
0.3	14/11/2019	Michal Smyrlis	Final version of the CTTP tool guideline.
0.2	05/11/2019	Michail Smyrlis	Draft version of the CTTP tool
0.1	17/09/2019	Michail Smyrlis	ToC

Executive Summary

This deliverable presents the first version of the reference CTTP models and CTTP programmes for the three pilots of THREAT-ARREST and is developed under the task “T3.2 - CTTP models and programmes development”. It provides: (a) an overview of the existing landscape and the threats and controls identified for the pilots, (b) the definition of the training characteristics, and (c) the specification of the models and the CTTP model.

Table of Contents

List of Abbreviations

ANTLR ANother Tool for Language Recognition

API Application Programming Interface

CTTP Cyber Threat and Training Preparation

DAO Data Access Object

EBNF Extended Backus–Naur Form

HTTP Hypertext Transfer Protocol

POJO Plain Old Java Object

REST Representational state transfer

SQL Structured Query Language

VM Virtual Machine

XML eXtensible Markup Language

List of Figures

1 Introduction

This deliverable is the final output of task "T3.1 – CTTP Language definition and Tool Support". It covers the cyber threat and training preparation (CTTP) Models and Programmes specification tool and acts as a guideline for the implementation approach and the deployment of the tool. It also provides the updates occurred to the CTTP language described since the initial deliverable "D3.1 – CTTP Models and Programmes Specification Language". The output of this deliverable (D3.2) will be used in the tasks "T3.2 – CTTP models and programmes development" and "T3.3 – Adaptation of CTTP models for new cyber threats" and more specifically in the deliverables "D3.3 – Reference CTTP Models and Programmes Specifications v1", "D3.4 – CTTP Models and Programmes Adaptation Procedures", "D3.5 – Reference CTTP Models and Programmes Specifications v2", and "D3.6 – CTTP Models and Programmes Adaptation Tool".

The need of this tool is of high importance as it provides a way to parse the CTTP model defined using the specification language of D3.1 and store the different components in the THREAT-ARREST database to later provide meaningful input to each THREAT-ARREST tool. It also provides a way to edit the existing grammar by using the CTTP model editor described in Section 4.1.

Figure 1 presents the CTTP Models and Programmes Specification Tool with regards to the existing architecture. The component lays within the Training Tool and receives the initial input from the Assurance Tool.

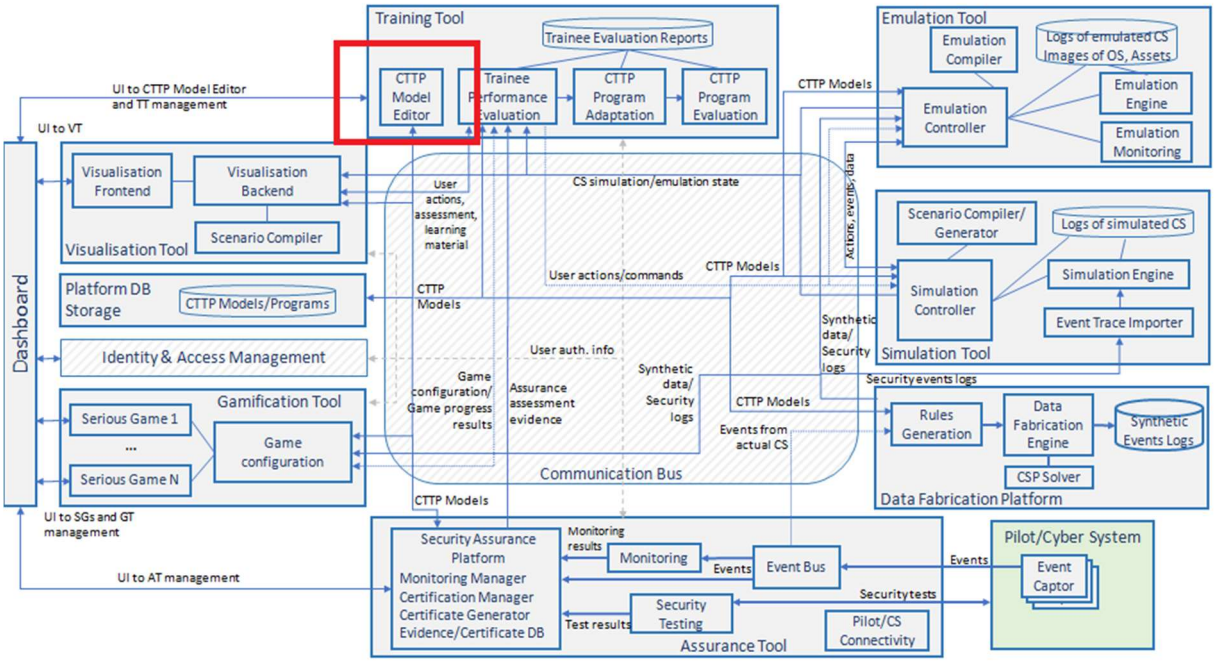

Figure 1: THREAT-ARREST Architecture and the CTTP model editor

Lastly, the document is organised as follows: Section 2 presents the implementation approach; Section 3 provides the structure of the CTTP language; Section 4 includes the deployment guide where an example will also be presented; Section 5 describes the THREAT-ARREST's lifecycle and how to collect the required information for the various CTTP parameters, while; Section 6 provides the conclusion.

2 Implementation approach

To implement the CTTP Models and Programmes Specification Tool, we combined a number of existing technologies both for the creation of the parser and editor and for the testing of the Application Programming Interface (API). Below we present a brief description of each technology used.

2.1 The ANTLR framework

ANother Tool for Language Recognition (ANTLR) is a parser generator for reading, processing, executing, or translating structured text or binary files (Antlr.org, 2019). ANTLR takes as input a formal language description, called a grammar, and generates a parser for that language that can automatically build parse trees, which are data structures representing how a grammar matches the input. ANTLR also automatically generates tree walkers that one can use to visit the nodes of those trees to execute application-specific code. The language is specified by using an extended Backus-Naur form (EBNF) grammar (Scowen, 1993; ISO/IEC, 1996). ANTLR then generates lexers and parsers. The latter can automatically generate parse trees or abstract syntax trees.

2.2 Postman

Postman (Postman, 2019; Du et al., 2018) is an API development tool which helps to build, test and modify APIs. Almost any functionality that could be needed by any developer is included within this tool. It is used by over 5 million developers every month to make their API development easy and simple. It has the ability to make various types of Hypertext Transfer Protocol (HTTP) requests (GET, POST, PUT, PATCH), saving environments for later use, converting the API to code for various languages (like Java, JavaScript, Python etc.).

2.3 The CTTP Model tool source code

To implement the source code for the CTTP model tool, we used the Spring Boot framework (Spring Boot, 2019) as the Java platform. For every element identified within the CTTP grammar, we create its data access object (DAO), its entity (a Plain Old Java Object (POJO) that can be persisted to the database) and a service that contains the classes for inserting, deleting, updating or find the object to the database. The difference between the DAO and the service is that the former exists to provide a connection to the database while the latter provides the logic to operate on the data sent to and from the DAO and the client. We also used Hibernate (Hibernate, 2019) to map the entities to the database tables and the Java data types to the Structured Query Language (SQL) data types and also write the data queries and retrieve result sets.

3 The Cyber Threat Training and Preparation (CTTP) Model Tool

In this section, the Cyber Threat Training and Preparation (CTTP) Model Tool is presented. More specifically, we provide an overview of ANTLR, the tool used to generate the parser, and then provide multiple snapshots of the source code used to create the parser and store each attribute of the CTTP Language to the database. We also provide, the database schema of the different components identified within the language. Lastly, we present the updates occurred to the language described in "D3.1 – CTTP Models and Programmes Specification Language".

3.1 Overview

In the context of THREAT-ARREST, ANTLR was used to generate the parser of the CTTP model created in D3.1. More specifically, the language developed in D3.1 was used as the input of the tool while the output was the CTTP model source code in Java.

We then utilised the ANTLR 4 maven plugin[1] to generate the lexer as a base listener (i.e. the default way ANTLR provides to traverse the syntax tree) and a parser.

The first thing we did was to create a grammar file. The grammar file will include the CTTP model text we wanted to parse and attempted to match the string to a number of lexer tokens. This grammar file is a human-readable text file appended with '.g4'. Figure 2 **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**provides the initial rule of the grammar that is used to orchestrate the different components identified in it. More specifically, this rule provides the definition of the assets and its quantifiers as described in D3.1.

```
/*Parser Rules*/
cttp: (person)? (COMMA person)* (COMMA)?
      (softwareAsset)? (COMMA softwareAsset)* (COMMA)?
      (hardwareAsset)? (COMMA hardwareAsset)* (COMMA)?
      (data)? (COMMA data)* (COMMA)?
     (parameter)? (COMMA parameter)* (COMMA)?
     (operation)? (COMMA operation)* (COMMA)?
     (securityProperty)? (COMMA securityProperty)*
     (hardwareSecurityControl)? (COMMA hardwareSecurityControl)*
     (softwareSecurityControl)? (COMMA softwareSecurityControl)*
```

Figure 2: The CTTP Models and Programmes Specification grammar's rule.

The created lexer can be used as a parser and a listener to analyze the created CTTP grammar file (.g4). That means that we can get error messages if the example text does not comply with the grammar or the parse tree of the example text, created from ANTLR4, if there are no errors.

Based on the lexer, ANTLR4 generated the base listener which included an empty class for all the rules identified in the grammar. Figure 3 presents an example of a listener. More specifically, when a grammar is defined, ANTL4 provides a set of classes (enter and exit) for every different rule described in the grammar.

[1]ANTLR 4 maven plugin: https://www.antlr.org/api/maven-plugin/latest/

Figure 3: ANTLR4 Listener (Antlr4 - Visitor vs Listener Pattern, 2019)

Lastly, we created the *CTTPLanguageBaseListener* Implementation class, where we included the Java source code for all the different rules in order to store each element to its corresponding database table and expose a Representational state transfer (REST) API to *insert*, *get*, *delete* and *update* each different class.

3.2 The CTTP Model parser

3.2.1 The CTTP Model updates

The first version of the CTTP Model was described in "D3.1 – CTTP Models and Programmes Specification Language". While creating the model's parser, several updates occurred to the existing model. More specifically, we updated the name for a number of attributes throughout the grammar (i.e. *SecurityAssuranceModelElement.sdateFrom* to *SecurityAssuranceModelElement.activeFrom* etc.). We also inserted multiple elements or attributes that were essential for better describing the pilot's systems. Lastly, we identified elements that were of no use and removed them. These changes are shown in the following subsections.

3.2.2 The CTTP Model Controller

Figure 4 shows the source code of the CTTP Controller. More specifically, a POST method was created to submit the CTTP grammar to the parser and store its element to the database.

```
@PostMapping(value = "/threatarrest/rest/api/cttp/parser", consumes = "multipart/form-data")
public ResponseEntity<?> antlr(@RequestParam("Grammar") MultipartFile[] submissions) {
    if (submissions == null) {
        return new ResponseEntity<String>( body: "No grammar parsed. Select a file!", HttpStatus.BAD_REQUEST);
    } else {
        int i = submissions.length;
        if (i <= 0)
            return new ResponseEntity<String>( body: "no result", HttpStatus.BAD_GATEWAY);
        String line="";
        try {
            CharStream antlr4stream = CharStreams.fromReader(new FileReader(GrammarUtils.receiveFile(submissions[0])));
            ModelEditorLexer lexer = new ModelEditorLexer(antlr4stream);
            CommonTokenStream tokens = new CommonTokenStream(lexer);
            ModelEditorParser parser = new ModelEditorParser(tokens);
            ParseTree tree = parser.model();
            ParseTreeWalker.DEFAULT.walk( listener: this, tree);
            return new ResponseEntity<String>( body: "Objects stored!", HttpStatus.CREATED);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return new ResponseEntity<String>( body: "Failed to parse grammar", HttpStatus.BAD_GATEWAY);
    }

}
```

Figure 4: CTTP tool controller

3.2.3 The Core Assurance Model parser

The subsections below present the core model parser, the updates occurred to the components of it as described in D3.1, its grammar, and its Java classes that implements the data access objects and the services and its database schema.

3.2.3.1 Core Assurance Model Updates

For the core assurance model, the following changes were made:

Deleted:

- *Process*
- *Vulnerabilities*

The core assurance model was updated to describe all the different assets identified from the pilots' systems. As for the vulnerabilities, we removed the class from the CTTP grammar as they will be part of the Assurance Tool (CUMULUS project, 2012; Krotsiani et al., 2017; Palacios et al., 2015; Hatzivasilis et al., 2019a; Hatzivasilis et al., 2019b), not the grammar. Upon successful identification of the different assets (via utilising the CTTP creator), the Assurance Tool will automatically update the Training Tool for the existing vulnerabilities of each identified asset.

3.2.3.2 Core Assurance Grammar

The subsections below present a subset of the core parser rules, the Java classes implementing the data access objects and the services and its database schema. The special characters show (i.e. *, *?*) are the quantifiers that provide the amount of times each rule can occur within the grammar. For instance, ***(person)? (COMMA person)* (COMMA)?*** means that we can have *0* or *1* person assets, following by *0* or more person assets and a comma if a different asset will be described.

3.2.3.2.1 Core Parser Rules

```
grammar CTTPLanguage;
/*Parser Rules*/
cttp:   (person)? (COMMA person)* (COMMA)?
        (softwareAsset)? (COMMA softwareAsset)* (COMMA)?
        (hardwareAsset)? (COMMA hardwareAsset)* (COMMA)?
        (data)? (COMMA data)* (COMMA)?
        (parameter)? (COMMA parameter)* (COMMA)?
        (operation)? (COMMA operation)* (COMMA)?
        (securityProperty)? (COMMA securityProperty)*
        (hardwareSecurityControl)? (COMMA hardwareSecurityControl)*
        (softwareSecurityControl)? (COMMA softwareSecurityControl)*
      ;
//Universal Identifiers for Assets
owner: 'owner' OPEN_PAREN  (STRING (COMMA)*)*  CLOSE_PAREN  ;
assetID: Identifier;
value: 'value' OPEN_PAREN DOUBLE CLOSE_PAREN;
name: 'name' OPEN_PAREN STRING CLOSE_PAREN;
activeFrom: 'activeFrom' OPEN_PAREN Date CLOSE_PAREN;
activeTo: 'activeTo' OPEN_PAREN Date CLOSE_PAREN;
description: 'description' OPEN_PAREN STRING CLOSE_PAREN;
creator: 'creator' OPEN_PAREN (OPEN_PAREN STRING CLOSE_PAREN (COMMA)*)*
CLOSE_PAREN;


propertyRule: 'property' OPEN_PAREN (OPEN_PAREN assetID COMMA secPropertyID
CLOSE_PAREN (COMMA)*)* CLOSE_PAREN;

controledByRule: 'controlledBy' OPEN_PAREN (  STRING  (COMMA)*)*
CLOSE_PAREN;

containsTypeRule: 'contains' (OPEN_PAREN assetID COMMA assetID COMMA
(containmentRule)+ CLOSE_PAREN
                   (COMMA)*)* CLOSE_PAREN;


containmentRule : 'containment' OPEN_PAREN (managedBy)+ COMMA operatesIn
CLOSE_PAREN;
containmentID: Identifier;
operatesIn : bool;
managedBy : managedByType;

managedByType: managedByTitle OPEN_PAREN (OPEN_PAREN assetID CLOSE_PAREN
(COMMA)*)* CLOSE_PAREN;
managedByTitle: 'managedBy';
```

3.2.3.2.2 Software Asset

```
/*Software Asset Element*/
softwareAsset : softTitle OPEN_PAREN assetVendor COMMA assetVersion COMMA name
COMMA softwareKind
                COMMA value COMMA currency COMMA softwareType (COMMA creator)?
                (COMMA owner)? (COMMA activeTo)? (COMMA NEWLINE description)?
(COMMA dataUse)?
                (COMMA requiresRule)? (COMMA providesImplRule)? (COMMA
containsTypeRule)?
                (COMMA controledByRule)? (COMMA hasSecurityProperty)* (COMMA
protectsRule)*
                CLOSE_PAREN
               ;
softTitle: 'SoftwareAsset';
```

```
softwareAssetID : Identifier;
softwareType : 'type' OPEN_PAREN SoftwareType CLOSE_PAREN;
assetVendor : 'vendor' OPEN_PAREN STRING CLOSE_PAREN;
assetVersion: 'version' OPEN_PAREN STRING CLOSE_PAREN;
currency: 'currency' OPEN_PAREN CurrencyType CLOSE_PAREN;
softwareKind: 'kind' OPEN_PAREN SoftwareKind CLOSE_PAREN;
hasSecurityProperty: 'property' OPEN_PAREN (OPEN_PAREN PropertyCategoryType
CLOSE_PAREN (COMMA)*)* CLOSE_PAREN;

requiresRule: 'requires' OPEN_PAREN (OPEN_PAREN softwareAssetID COMMA interfaceID
CLOSE_PAREN (COMMA)*)* CLOSE_PAREN;
providesImplRule: 'providesImpl' OPEN_PAREN (OPEN_PAREN softwareAssetID COMMA
interfaceImpID CLOSE_PAREN (COMMA)*)* CLOSE_PAREN;
dataUse: 'uses' OPEN_PAREN (dataID (COMMA)*)* CLOSE_PAREN;
```

3.2.3.2.3 Hardware Asset

```
/*Hardware Asset Element*/
hardwareAsset :
            hardwareAssetTitle OPEN_PAREN assetVendor COMMA assetVersion COMMA
name
            COMMA value COMMA currency COMMA hwType (COMMA creator)?
            (COMMA owner)? (COMMA activeTo)? (COMMA description)? (COMMA
hasSubmoduleRule)? (COMMA containmentRule)?
            (COMMA controledByRule)? (COMMA cpuModule)? (COMMA memoryModule)*
(COMMA driveModule)* (COMMA portModule)?
            (COMMA hasSecurityProperty)* (COMMA protectsRule)*
            CLOSE_PAREN;

hardwareAssetTitle : 'HardwareAsset';
hardwareID : Identifier;
hwType: 'hwType' OPEN_PAREN HardwareType CLOSE_PAREN;
hasSubmoduleRule: 'hasSubmodule' OPEN_PAREN (OPEN_PAREN hardwareID COMMA
hardwareID CLOSE_PAREN
                    (COMMA)*)* CLOSE_PAREN;
/*Hardware Modules*/
memoryModule:
        memoryModuleTitle OPEN_PAREN memorySize
        COMMA memoryType COMMA memorySpeed COMMA memoryManufacturer CLOSE_PAREN
        ;
memoryModuleTitle : 'MemoryModule';
memorySize: 'size' OPEN_PAREN INT CLOSE_PAREN;
memoryType : 'type' OPEN_PAREN STRING CLOSE_PAREN;
memorySpeed : 'speed' OPEN_PAREN DOUBLE CLOSE_PAREN;
memoryManufacturer: 'manufacturer' OPEN_PAREN STRING CLOSE_PAREN;

driveModule:
    driveModuleTitle OPEN_PAREN  hardwareID COMMA
    driveController COMMA driveModel COMMA driveFirmware COMMA driveCapacity
CLOSE_PAREN
    ;
driveModuleTitle: 'DriveModule';
driveController : 'controller' OPEN_PAREN STRING CLOSE_PAREN;
driveFirmware : 'firmware' OPEN_PAREN STRING CLOSE_PAREN;
driveModel : 'model' OPEN_PAREN STRING CLOSE_PAREN;
driveCapacity : 'capacity' OPEN_PAREN STRING CLOSE_PAREN;

portModule:
    portModuleTitle OPEN_PAREN  hardwareID COMMA
    inputOutputPort COMMA isAlwaysConnected CLOSE_PAREN
```

```
    ;
portModuleTitle: 'PortModule';
inputOutputPort : 'ioType' OPEN_PAREN InputOutputType CLOSE_PAREN;
isAlwaysConnected : 'isAlwaysConnected' OPEN_PAREN bool CLOSE_PAREN;

cpuModule : cpuTitle OPEN_PAREN hardwareID COMMA
            processorName COMMA numberOfCores COMMA numberOfThreads COMMA
processorBaseFrequence
            COMMA socket CLOSE_PAREN;
cpuTitle : 'CpuModule';
processorName : 'processorName' OPEN_PAREN STRING CLOSE_PAREN;
numberOfCores  : 'cores' OPEN_PAREN INT CLOSE_PAREN;
numberOfThreads : 'threads' OPEN_PAREN INT CLOSE_PAREN;
processorBaseFrequence : 'baseFrequency' OPEN_PAREN DOUBLE CLOSE_PAREN;
socket : 'socket' OPEN_PAREN STRING CLOSE_PAREN;
```

3.2.3.3 Java Classes

Software Asset Controller

```
@Override
public void enterSoftwareAsset(ModelEditorParser.SoftwareAssetContext ctx) {
    int countForStatus = 0;
    Softwareasset softwareasset = new Softwareasset();
    Timestamp timestamp = new Timestamp(System.currentTimeMillis());

softwareasset.setAssetVendor(GrammarUtils.removequotation(ctx.assetVendor().STRING
().getText()));

softwareasset.setAssetVersion(GrammarUtils.removequotation(ctx.assetVersion().STRI
NG().getText()));

softwareasset.setAssetName(GrammarUtils.removequotation(ctx.name().STRING().getTex
t()));
    softwareasset.setAssetValue(Double.parseDouble(ctx.value().getText()));
    Asset asset = new Asset();
    int assetType =
assetTypeService.findByName("Software".toLowerCase()).getAtID();
    asset.setTypeID(assetType);
    asset.setActiveFrom(softwareasset.getActiveFrom());
    asset.setActiveTo(softwareasset.getActiveTo());
    asset.setTableName("Softwareasset");
    asset.setOrganisationID(softwareasset.getOrganisationID());
    Project project = projectService.findById(softwareasset.getProjectID());
    Set<Project> projects = new HashSet<>();
    projects.add(project);
    asset.setProjects(projects);
    Project project1 = projectService.findById(softwareasset.getProjectID());
    assetService.insert(asset);
    softwareasset.setAsset(asset);
    Integer currencyTypID =
currencyTypeService.findByName(ctx.currency().CurrencyType().getText()).getCurrenc
yId();
    Integer softTypeID =
softwaretypeService.findByName(ctx.softwareType().SoftwareType().getText()).getSty
peId();
    Integer kindID =
softwarekindtypeService.findByName(ctx.softwareKind().SoftwareKind().getText()).ge
tKindID();
    softwareasset.setTypeID(softTypeID);
```

```
    softwareasset.setCurrencyID(currencyTypID);
    softwareasset.setKindID(kindID);
    softwareasset.setActiveFrom(timestamp);
    if(ctx.owner()!=null) {
        countForStatus++;
        Set<Person> personSet = new HashSet<Person>(0);
        ctx.owner().STRING().forEach((c) ->
                personSet.add(

personService.findByName(GrammarUtils.removequotation((c).getText().toLowerCase())
)
                )

        );
        softwareasset.setPersonSet(personSet);
    }
    Set<User> userSet = new HashSet<>();
    if(ctx.creator()!=null) {
        countForStatus++;
        ctx.creator().STRING().forEach((creator) ->
                userSet.add(

userService.findByUsername(GrammarUtils.removequotation((creator).getText().toLowe
rCase())
                        ))
        );
        softwareasset.setUserSet(userSet);
    }
    if(ctx.activeTo()!=null){
        countForStatus++;


softwareasset.setActiveTo(GrammarUtils.convertToTimestamp(ctx.activeTo().Date().ge
tText()));
    }
    if(GrammarUtils.removequotation(ctx.description().STRING().getText())!=null){
        countForStatus++;

softwareasset.setDescription(GrammarUtils.removequotation(ctx.description().STRING
().getText()));
    }
    if(countForStatus==4){
        softwareasset.setStatusID(2);
    }

    if (softwareassetService.insert(softwareasset)!=null){
        if (project1.getActiveTo()!=null) {
            if (project1.getActiveTo().before(timestamp)) {

project1.setStatusID(projectStatusService.findByName("closed".toLowerCase()).getPr
sID());
            }
            else{

project1.setStatusID(projectStatusService.findByName("defined".toLowerCase()).getP
rsID());
            }
        }
        else{
```

```
project1.setStatusID(projectStatusService.findByName("defined".toLowerCase()).getP
rsID());
        }
        projectService.update(project1);
    }
}
```

Software Asset DAO

```
public interface SoftwareassetDAO {
    SoftwareAsset insert(SoftwareAsset aModelElem);

    void update(SoftwareAsset aModelElem);

    void remove(int theId);

    List<SoftwareAsset> findAll();

    SoftwareAsset findById(int theId);

    SoftwareAsset findByName(String theName);
}
```

Software Asset DAO Implementation

```
package ch.sphynx.Implementation.dao;

import ch.sphynx.Implementation.entities.SoftwareAsset;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class SoftwareassetDAOImpl implements SoftwareassetDAO{
    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<SoftwareAsset> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<SoftwareAsset> softwareAssetList =  session.createQuery("from
SoftwareAsset").list();

            for(SoftwareAsset a: softwareAssetList){
                logger.info("SoftwareAsset List::" + a);
            }
            return softwareAssetList;
```

```
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public SoftwareAsset findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SoftwareAsset a = (SoftwareAsset) session.load(SoftwareAsset.class,
new Integer(theId));
            logger.info("SoftwareAsset loaded successfully, SoftwareAsset
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public SoftwareAsset findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from SoftwareAsset where name=:name");
            query.setParameter("name", theName);

            SoftwareAsset a = (SoftwareAsset) query.uniqueResult();
            logger.info("SoftwareAsset loaded successfully, SoftwareAsset
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public SoftwareAsset insert(SoftwareAsset softwareAssetObjectInsert) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(softwareAssetObjectInsert);
            return softwareAssetObjectInsert;
        }
        catch(Exception e){
           throw  e;
        }
    }

    @Override
    public void update(SoftwareAsset softwareAssetObjectUpdate) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(softwareAssetObjectUpdate);
```

```
            logger.info("SoftwareAsset updated successfully, SoftwareAsset
Details="+ softwareAssetObjectUpdate);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SoftwareAsset a = (SoftwareAsset) session.load(SoftwareAsset.class,
new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("SoftwareAsset deleted successfully, SoftwareAsset
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Software Asset Entity

```
package ch.sphynx.Implementation.entities;


import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;
import java.sql.Timestamp;
import java.util.Date;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Entity
@Table(name="SoftwareAsset")
public class SoftwareAsset implements java.io.Serializable{
    @Id
    @Column(name = "softwareassetID", nullable = false)
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Long saID;
    private String assetName;
    private String assetVendor;
    private String assetVersion;
    private Double assetValue;
    @Transient
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
```

```
    private Timestamp activeFrom;
    @Transient
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private Timestamp activeTo;
    @Transient
    @JsonIgnore
    private String description;
    private Integer currencyID;
    private Integer typeID;

    @OneToOne
    @JoinColumn(name="assetID")
    private Asset asset;
    private Integer kindID;
    private Integer statusID;
    @Transient
    private Long projectID;
    @Transient
    private Long organisationID;
    @JsonIgnore
    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name = "OwnerSoftwareAssetJoin",
            joinColumns = @JoinColumn(name = "softwareassetID",unique = true),
            inverseJoinColumns = @JoinColumn(name = "personID",unique = true))
    private Set<Person> personSet = new HashSet<>();

    @JsonIgnore
    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name = "CreatorSoftwareAssetJoin",
            joinColumns = @JoinColumn(name = "saID",unique = true),
            inverseJoinColumns = @JoinColumn(name = "userID",unique = true))
    private Set<User> userSet= new HashSet<>();



    public SoftwareAsset() {
    }

    public SoftwareAsset(String assetName, String assetVendor, String
assetVersion, Double assetValue, Timestamp activeFrom, Timestamp activeTo, String
description, Integer currencyID, Integer typeID, Asset asset, Integer kindID,
Integer statusID, Long projectID, Long organisationID, Set<Person> personSet,
Set<User> userSet) {
        this.assetName = assetName;
        this.assetVendor = assetVendor;
        this.assetVersion = assetVersion;
        this.assetValue = assetValue;
        this.activeFrom = activeFrom;
        this.activeTo = activeTo;
        this.description = description;
        this.currencyID = currencyID;
        this.typeID = typeID;
        this.asset = asset;
        this.kindID = kindID;
        this.statusID = statusID;
        this.projectID = projectID;
        this.organisationID = organisationID;
        this.personSet = personSet;
        this.userSet = userSet;
    }
```

```
    public Long getSaID() {
        return saID;
    }

    public void setSaID(Long saID) {
        this.saID = saID;
    }

    public String getAssetName() {
        return assetName;
    }

    public void setAssetName(String assetName) {
        this.assetName = assetName;
    }

    public String getAssetVendor() {
        return assetVendor;
    }

    public void setAssetVendor(String assetVendor) {
        this.assetVendor = assetVendor;
    }

    public String getAssetVersion() {
        return assetVersion;
    }

    public void setAssetVersion(String assetVersion) {
        this.assetVersion = assetVersion;
    }

    public Double getAssetValue() {
        return assetValue;
    }

    public void setAssetValue(Double assetValue) {
        this.assetValue = assetValue;
    }

    public Timestamp getActiveFrom() {
        return activeFrom;
    }

    public void setActiveFrom(Timestamp activeFrom) {
        this.activeFrom = activeFrom;
    }

    public Timestamp getActiveTo() {
        return activeTo;
    }

    public void setActiveTo(Timestamp activeTo) {
        this.activeTo = activeTo;
    }

    public String getDescription() {
        return description;
    }
```

```
    public void setDescription(String description) {
        this.description = description;
    }

    public Integer getCurrencyID() {
        return currencyID;
    }

    public void setCurrencyID(Integer currencyID) {
        this.currencyID = currencyID;
    }

    public Integer getTypeID() {
        return typeID;
    }

    public void setTypeID(Integer typeID) {
        this.typeID = typeID;
    }

    public Asset getAsset() {
        return asset;
    }

    public void setAsset(Asset asset) {
        this.asset = asset;
    }

    public Integer getKindID() {
        return kindID;
    }

    public void setKindID(Integer kindID) {
        this.kindID = kindID;
    }

    public Integer getStatusID() {
        return statusID;
    }

    public void setStatusID(Integer statusID) {
        this.statusID = statusID;
    }

    public Long getProjectID() {
        return projectID;
    }

    public void setProjectID(Long projectID) {
        this.projectID = projectID;
    }

    public Long getOrganisationID() {
        return organisationID;
    }

    public void setOrganisationID(Long organisationID) {
        this.organisationID = organisationID;
    }
```

```
    public Set<Person> getPersonSet() {
        return personSet;
    }

    public void setPersonSet(Set<Person> personSet) {
        this.personSet = personSet;
    }

    public Set<User> getUserSet() {
        return userSet;
    }

    public void setUserSet(Set<User> userSet) {
        this.userSet = userSet;
    }
}
```

Software Asset Service

```
public interface SoftwareassetService {

    SoftwareAsset insert(SoftwareAsset aModelElem);

    void update(SoftwareAsset aModelElem);

    void remove(int theId);

    List<SoftwareAsset> findAll();

    SoftwareAsset findById(int theId);

    SoftwareAsset findByName(String theName);

}
```

Software Asset Service Implementation

```
package ch.sphynx.Implementation.services;

import ch.sphynx.Implementation.dao.SoftwareassetDAO;
import ch.sphynx.Implementation.entities.SoftwareAsset;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
public class SoftwareassetServiceImpl implements SoftwareassetService {

    @Autowired
    private SoftwareassetDAO objectDAO ;

    public SoftwareassetServiceImpl(SoftwareassetDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public SoftwareAsset insert(SoftwareAsset aModelElem) {
```

```
        return objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(SoftwareAsset aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<SoftwareAsset> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public SoftwareAsset findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public SoftwareAsset findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.3.4 Database

Based on the above, a database structure was created to illustrate the core assurance model database tables, as depicted in Figure 5.

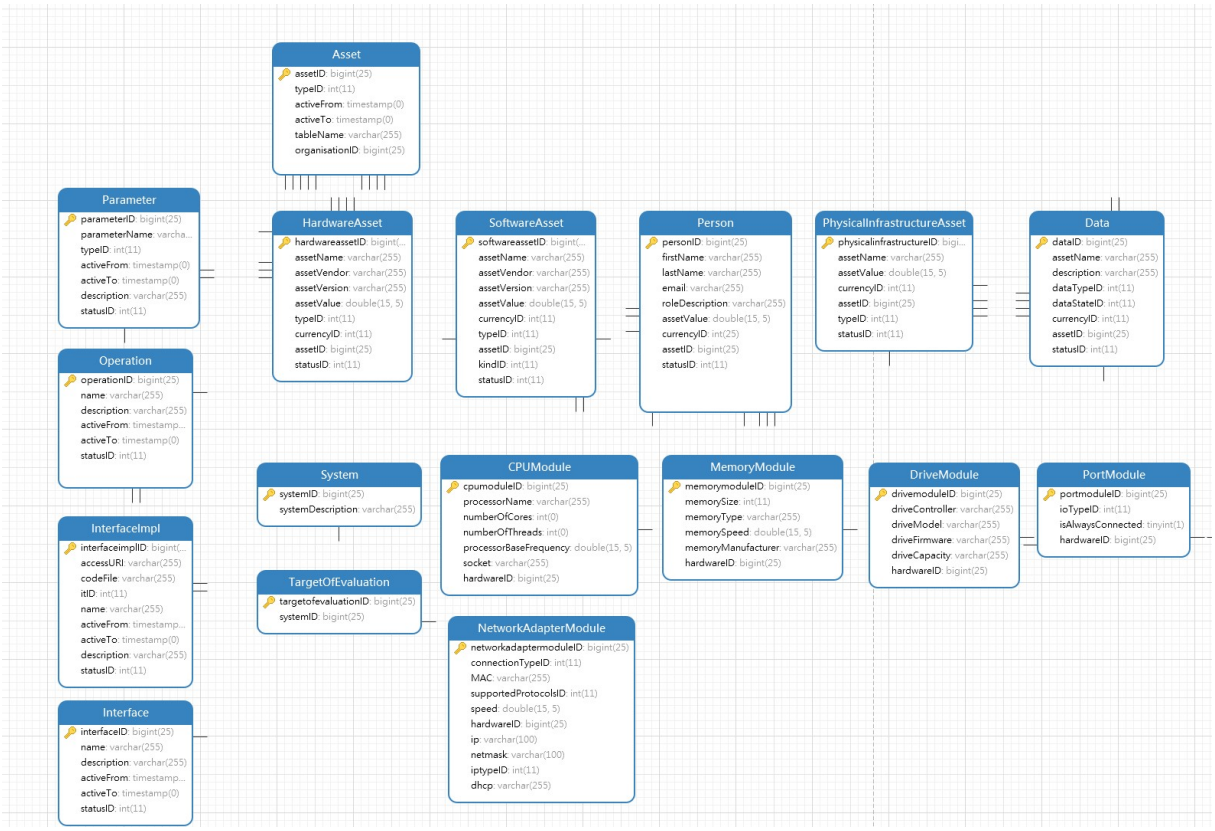

Figure 5: Asset database schema

3.2.4 The Training parser

The subsections below present the training programme parser, the updates occurred to the components of it as described in D3.1, its grammar, and its Java classes that implement the data access objects and the services, and its database schema.

3.2.4.1 The Training Programme updates

For the Training Programme, the following changes were made:

Deleted:

- *actualTrace*;
- *expectedTrace*;

The grammar required for the parser was updated to fit the new model and the respective Java classes and database tables where created for the implementation of the parser and can be found in the Sections below.

3.2.4.2 Grammar

```
******Training Programme (TP)*******

trainingProgramme: tpTitle OPEN_PAREN tpDescription COMMA tpGoal COMMA (tpRole)+
COMMA (tpType)? COMMA (legalFramework)* COMMA tpDifficulty (COMMA
concernsAssetRule)* (COMMA consistsOfPhaseRule)+
(COMMA employsSecurityControlRule)*  (COMMA coversThreatRule)+ (COMMA
involvesPropertyRule)+
(COMMA hasEmulationRule)+ (COMMA hasSimulationRule)+ (COMMA hasGamificationRule)+
(COMMA hasDataFabricationRule)+
(COMMA supportsTrainingProgrammeExecutionRule)+ CLOSE_PAREN;
tpID: Identifier;
```

```
tpTitle: 'Training Programme';
tpDescription: 'description' OPEN_PAREN STRING CLOSE_PAREN;
tpGoal: 'trainingProgrammeGoal' OPEN_PAREN goal CLOSE_PAREN;
tpRole: 'role' OPEN_PAREN PersonRoleType CLOSE_PAREN;
tpType: 'type' OPEN_PAREN TypesOfActionType  CLOSE_PAREN;
legalFramework: 'legalFramework' OPEN_PAREN legalFramework CLOSE_PAREN;
tpDifficulty: 'difficulty' OPEN_PAREN INT CLOSE_PAREN;
consistsOfPhaseRule:'consistsOf' OPEN_PAREN phaseID CLOSE_PAREN;
concernsAssetRule: 'concerns' OPEN_PAREN  assetID CLOSE_PAREN;
coversThreatRule:'covers' OPEN_PAREN threatID CLOSE_PAREN;
involvesPropertyRule: 'involves' OPEN_PAREN secPropertyID CLOSE_PAREN ;
employsSecurityControlRule: 'employs'OPEN_PAREN secControlID CLOSE_PAREN;
hasEmulationRule: 'includesEmModel' OPEN_PAREN cttpEmulationModelID CLOSE_PAREN;
hasSimulationRule: 'comprisesOfSimModel' OPEN_PAREN cttpSimulationModelID
CLOSE_PAREN;
hasGamificationRule: 'comprisesOfSimModel' OPEN_PAREN cttpGamificationModelID
CLOSE_PAREN;
hasDataFabricationRule: 'comprisesOfSimModel' OPEN_PAREN
cttpDataFabricationModelID CLOSE_PAREN;
supportsTrainingProgrammeExecutionRule: 'supports' OPEN_PAREN tpExecutionID
CLOSE_PAREN;


/*Account*/
account:accountTitle OPEN_PAREN  accountRole COMMA (followsRule)* COMMA
(isUtilizedByRule)+  CLOSE_PAREN;
accountTitle : 'Account';
accountID : Identifier;
accountRole : PersonRoleType;
followsRule : 'follows' OPEN_PAREN tpExecutionID CLOSE_PAREN;
isUtilizedByRule: 'isUtilizedBy' OPEN_PAREN personID CLOSE_PAREN;

/*TrainingProgramExecution*/
tpExecution:tpExecutionTitle OPEN_PAREN  (isFollowedByRule)+ COMMA
(isSupportedByRule)? CLOSE_PAREN;
tpExecutionTitle : 'Account';
tpExecutionID : Identifier;
isFollowedByRule: 'isFollowedBy' OPEN_PAREN accountID CLOSE_PAREN;
isSupportedByRule: 'isSupportedBy' OPEN_PAREN tpID CLOSE_PAREN;


/*Phase*/
phase:phaseTitle OPEN_PAREN orderOfExecution COMMA isPartOfTpRule COMMA
(isSupportedByEmRule)+
(isFacilitatedBySimRule)+ (isDrivenByRule)* CLOSE_PAREN;
phaseTitle : 'ExpectedTrace';
phaseID : Identifier;
orderOfExecution : 'orderOfExecution' OPEN_PAREN INT CLOSE_PAREN;
isPartOfTpRule: 'isPartOf' OPEN_PAREN tpID CLOSE_PAREN;
isSupportedByEmRule: 'isSupportedBy' OPEN_PAREN cttpEmulationModelID CLOSE_PAREN;
isFacilitatedBySimRule: 'isFacilitatedBy' OPEN_PAREN cttpSimulationModelID
CLOSE_PAREN;
isDrivenByRule: 'isDrivenBy' OPEN_PAREN eventSequenceID CLOSE_PAREN;

/*ThreatActor*/
threatActor: threatActorTitle OPEN_PAREN threatActorType COMMA (causesRule)*
CLOSE_PAREN;
threatActorTitle : 'threatActor';
threatActorID : Identifier;
```

```
threatActorType : OPEN_PAREN ThreatActorType CLOSE_PAREN;
causesRule: 'causes' OPEN_PAREN eventSequenceID CLOSE_PAREN;
```

3.2.4.3 Java Classes

3.2.4.3.1 Training Programme

Training Programme DAO

```
public interface TrainingProgrammeDAO {

    public void insert(TrainingProgramme aModelElem);

    public void update(TrainingProgramme aModelElem);

    public void remove(int theId);

    public List<TrainingProgramme> findAll();

    public TrainingProgramme findById(int theId);

    public TrainingProgramme findByName(String theName);
}
```

Training Programme DAO Implementation

```
@Repository
public class TrainingProgrammeDAOImpl implements TrainingProgrammeDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<TrainingProgramme> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<TrainingProgramme> TrainingProgrammeList =
session.createQuery("from TrainingProgramme").list();

            for(TrainingProgramme a: TrainingProgrammeList){
                logger.info("TrainingProgramme List::" + a);
            }
            return TrainingProgrammeList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public TrainingProgramme findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            TrainingProgramme a = (TrainingProgramme)
```

```
session.load(TrainingProgramme.class, new Integer(theId));
            logger.info("TrainingProgramme loaded successfully, TrainingProgramme
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public TrainingProgramme findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from TrainingProgramme where name=:name");
            query.setParameter("name", theName);

            TrainingProgramme a = (TrainingProgramme) query.uniqueResult();
            logger.info("TrainingProgramme loaded successfully, TrainingProgramme
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(TrainingProgramme TrainingProgrammeElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(TrainingProgrammeElement);
            logger.info("TrainingProgramme saved successfully, TrainingProgramme
Details="+ TrainingProgrammeElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(TrainingProgramme TrainingProgramme) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(TrainingProgramme);
            logger.info("TrainingProgramme updated successfully, TrainingProgramme
Details="+ TrainingProgramme);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
```

```
        try{
            Session session = this.sessionFactory.getCurrentSession();
            TrainingProgramme a = (TrainingProgramme)
session.load(TrainingProgramme.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("TrainingProgramme deleted successfully, TrainingProgramme
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Training Programme Entity

```
@Entity
public class TrainingProgramme implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long TrainingProgrammeID;
    private long goalID;
    private long typesOfActionID;
    private long dataFabricationModelID;
    private int difficulty;

    /*
    Many to Many Relationships
     */
    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammecttpsimmodeljoin",
                joinColumns = @JoinColumn(name="trainingProgrammeID", unique =
true),
                inverseJoinColumns = @JoinColumn(name="cttpSimModelID", unique =
true))
    private Set<CTTPSimulationModel> cttpSimulationModelSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammecttpgamificationmodeljoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="cttpGamificationModelID",
unique = true))
    private Set<CTTPGamificationModel> cttpGamificationModelSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammecttpemmodeljoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="cttpEmModelID", unique = true))
    private Set<CTTPEmulationModel>cttpEmulationModelSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammeassetjoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="assetID", unique = true))
```

```
    private Set<Asset>  assetSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammethreatjoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="threatID", unique = true))
    private Set<Threat>  threatSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammecttpsimmodeljoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="cttpSimModelID", unique =
true))
    private Set<SecurityControl>  securityControlSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammecttpsimmodeljoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="cttpSimModelID", unique =
true))
    private Set<SecurityProperty>  securityPropertySet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammerolejoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="roleID", unique = true))
    private Set<Role>  rolesSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammelegaLframeworktypejoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="legaFrameworkTypeID", unique =
true))
    private Set<LegalFrameworkType> legalFrameworkTypeSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogrammephasejoin",
            joinColumns = @JoinColumn(name="trainingProgrammeID", unique = true),
            inverseJoinColumns = @JoinColumn(name="phaseID", unique = true))
    private Set<Phase> phaseSet;



    public long getTrainingProgrammeID() {
        return TrainingProgrammeID;
    }
    public void setTrainingProgrammeID(long trainingProgrammeID) {
        TrainingProgrammeID = trainingProgrammeID;
    }
    public long getGoalID() {
        return goalID;
    }
    public void setGoalID(long goalID) {
        this.goalID = goalID;
    }
    public long getTypesOfActionID() {
        return typesOfActionID;
    }
    public void setTypesOfActionID(long typesOfActionID) {
        this.typesOfActionID = typesOfActionID;
```

```
    }
    public long getDataFabricationModelID() {
        return dataFabricationModelID;
    }
    public void setDataFabricationModelID(long dataFabricationModelID) {
        this.dataFabricationModelID = dataFabricationModelID;
    }
    public int getDifficulty() {
        return difficulty;
    }
    public void setDifficulty(int difficulty) {
        this.difficulty = difficulty;
    }
```

Training Programme Service

```
public interface TrainingProgrammeService {
    void insert(TrainingProgramme aModelElem);

    void update(TrainingProgramme aModelElem);

    void remove(int theId);

    List<TrainingProgramme> findAll();

    TrainingProgramme findById(int theId);

    TrainingProgramme findByName(String theName);
}
```

Training Programme Service Implementation

```
@Repository
public class TrainingProgrammeServiceImpl implements TrainingProgrammeService {
    @Autowired
    private TrainingProgrammeDAO objectDAO ;

    @Autowired
    public TrainingProgrammeServiceImpl(TrainingProgrammeDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(TrainingProgramme aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(TrainingProgramme aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
```

```
    @Transactional
    public List<TrainingProgramme> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public TrainingProgramme findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public TrainingProgramme findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.4.3.2 Training Program Execution

Training Programme Execution Entity

```
@Entity
public class TrainingProgramExecution implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long trainingProgramExecutionID;
    private long trainingProgrammeID;


    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="trainingprogramexecutionaccountjoin",
            joinColumns = @JoinColumn(name="trainingProgrammExecutionID", unique =
true),
            inverseJoinColumns = @JoinColumn(name="accountID", unique = true))
    private Set<Account> accountSet;
    @Id
    public long getTrainingProgramExecutionID() {
        return trainingProgramExecutionID;
    }
    public void setTrainingProgramExecutionID(long trainingProgramExecutionID) {
        this.trainingProgramExecutionID = trainingProgramExecutionID;
    }
    @Id
    public long getTrainingProgrammeID() {
        return trainingProgrammeID;
    }
    public void setTrainingProgrammeID(long trainingProgrammeID) {
        this.trainingProgrammeID = trainingProgrammeID;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof TrainingProgramExecution)) return false;
        TrainingProgramExecution that = (TrainingProgramExecution) o;
        return getTrainingProgramExecutionID() ==
that.getTrainingProgramExecutionID() &&
                getTrainingProgrammeID() == that.getTrainingProgrammeID();
    }
    @Override
```

```
    public int hashCode() {
        return Objects.hash(getTrainingProgramExecutionID(),
getTrainingProgrammeID());
    }

}
```

Training Programme Execution DAO Implementation

```
@Repository
public class TrainingProgrammeDAOImpl implements TrainingProgrammeDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<TrainingProgramme> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<TrainingProgramme> TrainingProgrammeList =
session.createQuery("from TrainingProgramme").list();

            for(TrainingProgramme a: TrainingProgrammeList){
                logger.info("TrainingProgramme List::" + a);
            }
            return TrainingProgrammeList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public TrainingProgramme findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            TrainingProgramme a = (TrainingProgramme)
session.load(TrainingProgramme.class, new Integer(theId));
            logger.info("TrainingProgramme loaded successfully, TrainingProgramme
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public TrainingProgramme findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
```

```
                    createQuery("from TrainingProgramme where name=:name");
            query.setParameter("name", theName);

            TrainingProgramme a = (TrainingProgramme) query.uniqueResult();
            logger.info("TrainingProgramme loaded successfully, TrainingProgramme
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(TrainingProgramme TrainingProgrammeElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(TrainingProgrammeElement);
            logger.info("TrainingProgramme saved successfully, TrainingProgramme
Details="+ TrainingProgrammeElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(TrainingProgramme TrainingProgramme) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(TrainingProgramme);
            logger.info("TrainingProgramme updated successfully, TrainingProgramme
Details="+ TrainingProgramme);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            TrainingProgramme a = (TrainingProgramme)
session.load(TrainingProgramme.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("TrainingProgramme deleted successfully, TrainingProgramme
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }
```

```
    }
}
```

Training Programme Service Implementation

```
@Repository
public class TrainingProgramExecutionServiceImpl implements
TrainingProgramExecutionService {
    @Autowired
    private TrainingProgramExecutionDAO objectDAO ;

    @Autowired
    public TrainingProgramExecutionServiceImpl(TrainingProgramExecutionDAO
constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(TrainingProgramExecution aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(TrainingProgramExecution aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<TrainingProgramExecution> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public TrainingProgramExecution findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public TrainingProgramExecution findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.4.3.3 Account

Account Entity

```
@Entity
public class Account implements java.io.Serializable{
    @Id
```

```
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long accountID;
    private String accountRoleID;


    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="accountpersonJoin",
            joinColumns = @JoinColumn(name="accountID", unique = true),
            inverseJoinColumns = @JoinColumn(name="personID", unique = true))
    private Set<Person> personSet;

    @Id
    public long getAccountID() {
        return accountID;
    }
    public void setAccountID(long accountID) {
        this.accountID = accountID;
    }
    @Id
    public String getAccountRoleID() {
        return accountRoleID;
    }
    public void setAccountRoleID(String accountRoleID) {
        this.accountRoleID = accountRoleID;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Account)) return false;
        Account account = (Account) o;
        return getAccountID() == account.getAccountID() &&
                Objects.equals(getAccountRoleID(), account.getAccountRoleID());
    }
    @Override
    public int hashCode() {
        return Objects.hash(getAccountID(), getAccountRoleID());
    }
}
```

Account DAO Implementation

```
@Repository
public class AccountDAOImpl implements AccountDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
    @Override
    public List<Account> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Account> AccountList =  session.createQuery("from
Account").list();

            for(Account a: AccountList){
                logger.info("Account List::" + a);
```

```
            }
            return AccountList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Account findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Account a = (Account) session.load(Account.class, new Integer(theId));
            logger.info("Account loaded successfully, Account details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Account findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Account where name=:name");
            query.setParameter("name", theName);

            Account a = (Account) query.uniqueResult();
            logger.info("Account loaded successfully, Account details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Account AccountElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(AccountElement);
            logger.info("Account saved successfully, Account Details="+
AccountElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Account Account) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Account);
```

```
            logger.info("Account updated successfully, Account Details="+
Account);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Account a = (Account) session.load(Account.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Account deleted successfully, Account details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Account Service

```
@Repository
public class AccountServiceImpl implements AccountService {
    @Autowired
    private AccountDAO objectDAO ;

    @Autowired
    public AccountServiceImpl(AccountDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Account aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Account aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
```

```
    public List<Account> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Account findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Account findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.4.3.4 Phase

Phase Entity

```
@Entity
public class Phase implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long phaseID;
    private int orderOfExecution;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="phaseeventsequencejoin",
            joinColumns = @JoinColumn(name="phaseID", unique = true),
            inverseJoinColumns = @JoinColumn(name="eventSequenceID", unique =
true))
    private Set<EventSequence> eventSequenceSet;

    @ManyToMany(mappedBy = "phaseSet", cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<CTTPEmulationModel>cttpEmulationModelSet;
    @ManyToMany(mappedBy = "phaseSet", cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<CTTPSimulationModel>cttpSimulationModelSet;
    @ManyToMany(mappedBy = "phaseSet", cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<TrainingProgramme>trainingProgrammeSet;

    public long getPhaseID() {
        return phaseID;
    }

    public void setPhaseID(long phaseID) {
        this.phaseID = phaseID;
    }

    public int getOrderOfExecution() {
        return orderOfExecution;
    }

    public void setOrderOfExecution(int orderOfExecution) {
        this.orderOfExecution = orderOfExecution;
```

```
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Phase)) return false;
        Phase phase = (Phase) o;
        return getPhaseID() == phase.getPhaseID() &&
                getOrderOfExecution() == phase.getOrderOfExecution();
    }

    @Override
    public int hashCode() {
        return Objects.hash(getPhaseID(), getOrderOfExecution());
    }
}
```

Phase DAO Implementation

```
@Repository
public class PhaseDAOImpl implements PhaseDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Phase> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Phase> PhaseList =  session.createQuery("from Phase").list();

            for(Phase a: PhaseList){
                logger.info("Phase List::" + a);
            }
            return PhaseList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Phase findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Phase a = (Phase) session.load(Phase.class, new Integer(theId));
            logger.info("Phase loaded successfully, Phase details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
```

```
    }

    @Override
    public Phase findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Phase where name=:name");
            query.setParameter("name", theName);

            Phase a = (Phase) query.uniqueResult();
            logger.info("Phase loaded successfully, Phase details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Phase PhaseElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(PhaseElement);
            logger.info("Phase saved successfully, Phase Details="+ PhaseElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Phase Phase) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Phase);
            logger.info("Phase updated successfully, Phase Details="+ Phase);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Phase a = (Phase) session.load(Phase.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Phase deleted successfully, Phase details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }
```

```
    }
}
```

Phase Service Implementation

```
@Repository
public class PhaseServiceImpl implements PhaseService {
    @Autowired
    private PhaseDAO objectDAO ;

    @Autowired
    public PhaseServiceImpl(PhaseDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Phase aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Phase aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Phase> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Phase findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Phase findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.4.3.5 Threat Actor

Threat Actor Entity

```
@Entity
public class ThreatActor implements java.io.Serializable{


    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long threatActorID;
    @Id
    private long typeID;

    @ManyToMany(mappedBy = "threatActorID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<EventSequence> eventSequenceset;

    @Id
    public long getThreatActorID() {
        return threatActorID;
    }

    public void setThreatActorID(long threatActorID) {
        this.threatActorID = threatActorID;
    }
    @Id
    public long getTypeID() {
        return typeID;
    }

    public void setTypeID(long typeID) {
        this.typeID = typeID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ThreatActor)) return false;
        ThreatActor that = (ThreatActor) o;
        return getThreatActorID() == that.getThreatActorID() &&
                getTypeID() == that.getTypeID();
    }

    @Override
    public int hashCode() {
        return Objects.hash(getThreatActorID(), getTypeID());
    }
}
```

Threat Actor DAO Implementation

```
@Repository
public class ThreatActorDAOImpl implements ThreatActorDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
```

```
    }

    @Override
    public List<ThreatActor> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<ThreatActor> ThreatActorList =  session.createQuery("from
ThreatActor").list();

            for(ThreatActor a: ThreatActorList){
                logger.info("ThreatActor List::" + a);
            }
            return ThreatActorList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public ThreatActor findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            ThreatActor a = (ThreatActor) session.load(ThreatActor.class, new
Integer(theId));
            logger.info("ThreatActor loaded successfully, ThreatActor details="+
a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public ThreatActor findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from ThreatActor where name=:name");
            query.setParameter("name", theName);

            ThreatActor a = (ThreatActor) query.uniqueResult();
            logger.info("ThreatActor loaded successfully, ThreatActor details="+
a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(ThreatActor ThreatActorElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
```

```
            session.persist(ThreatActorElement);
            logger.info("ThreatActor saved successfully, ThreatActor Details="+
ThreatActorElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(ThreatActor ThreatActor) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(ThreatActor);
            logger.info("ThreatActor updated successfully, ThreatActor Details="+
ThreatActor);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            ThreatActor a = (ThreatActor) session.load(ThreatActor.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("ThreatActor deleted successfully, ThreatActor
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Threat Actor Service Implementation

```
@Repository
public class ThreatActorServiceImpl implements ThreatActorService {
    @Autowired
    private ThreatActorDAO objectDAO ;

    @Autowired
    public ThreatActorServiceImpl(ThreatActorDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(ThreatActor aModelElem) {
        objectDAO.insert(aModelElem);
```

```
    }

    @Override
    @Transactional
    public void update(ThreatActor aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<ThreatActor> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public ThreatActor findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public ThreatActor findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.4.3.6 Event Sequence

Event Sequence Entity

```
@Entity
public class EventSequence implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long eventSequenceID;
    @Id
    private long threatID;
    @Id
    private long likehoodID;
    private String threatEvent;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="eventsequencethreatactorjoin",
            joinColumns = @JoinColumn(name="eventSequenceID", unique = true),
            inverseJoinColumns = @JoinColumn(name="threatActorID", unique = true))
    private Set<ThreatActor> threatActorSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="eventsequencethreatjoin",
            joinColumns = @JoinColumn(name="eventSequenceID", unique = true),
            inverseJoinColumns = @JoinColumn(name="threatID", unique = true))
    private Set<Threat> threatSet;
```

```
    public long getEventSequenceID() {
        return eventSequenceID;
    }

    public void setEventSequenceID(long eventSequenceID) {
        this.eventSequenceID = eventSequenceID;
    }

    @Id
    public long getThreatID() {
        return threatID;
    }

    public void setThreatID(long threatID) {
        this.threatID = threatID;
    }

    @Id
    public long getLikehoodID() {
        return likehoodID;
    }

    public void setLikehoodID(long likehoodID) {
        this.likehoodID = likehoodID;
    }


    public String getThreatEvent() {
        return threatEvent;
    }

    public void setThreatEvent(String threatEvent) {
        this.threatEvent = threatEvent;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof EventSequence)) return false;
        EventSequence that = (EventSequence) o;
        return getEventSequenceID() == that.getEventSequenceID() &&
                getThreatID() == that.getThreatID() &&
                getLikehoodID() == that.getLikehoodID() &&
                Objects.equals(getThreatEvent(), that.getThreatEvent());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getEventSequenceID(), getThreatID(), getLikehoodID(),
getThreatEvent());
    }

}
```

Event Sequence DAO Implementation

```
@Repository
public class EventSequenceDAOImpl implements EventSequenceDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
```

```
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<EventSequence> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<EventSequence> EventSequenceList =  session.createQuery("from
EventSequence").list();

            for(EventSequence a: EventSequenceList){
                logger.info("EventSequence List::" + a);
            }
            return EventSequenceList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public EventSequence findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            EventSequence a = (EventSequence) session.load(EventSequence.class,
new Integer(theId));
            logger.info("EventSequence loaded successfully, EventSequence
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public EventSequence findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from EventSequence where name=:name");
            query.setParameter("name", theName);

            EventSequence a = (EventSequence) query.uniqueResult();
            logger.info("EventSequence loaded successfully, EventSequence
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }
```

```
    @Override
    public void insert(EventSequence EventSequenceElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(EventSequenceElement);
            logger.info("EventSequence saved successfully, EventSequence
Details="+ EventSequenceElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(EventSequence EventSequence) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(EventSequence);
            logger.info("EventSequence updated successfully, EventSequence
Details="+ EventSequence);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            EventSequence a = (EventSequence) session.load(EventSequence.class,
new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("EventSequence deleted successfully, EventSequence
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Event Sequence Service Implementation

```
@Repository
public class EventSequenceServiceImpl implements EventSequenceService {
    @Autowired
    private EventSequenceDAO objectDAO ;

    @Autowired
    public EventSequenceServiceImpl(EventSequenceDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }
```

```
    @Override
    @Transactional
    public void insert(EventSequence aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(EventSequence aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<EventSequence> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public EventSequence findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public EventSequence findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.4.4 Database

Based on the above, a database structure was created to illustrate the training database tables, as depicted in Figure 6.

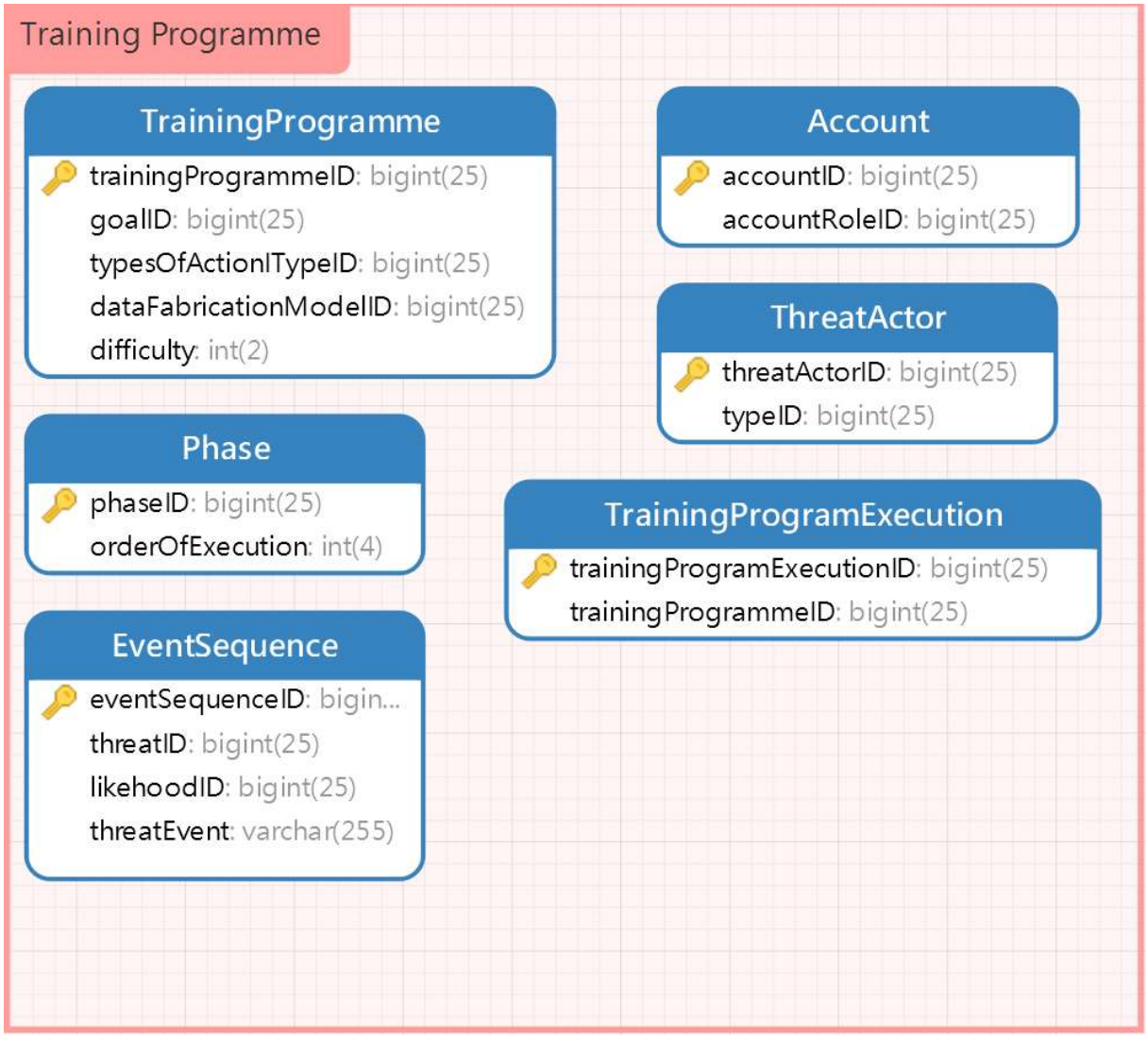

Figure 6: Training Programme Database Design

3.2.5 The Simulation parser

The subsections below present the simulation parser, the updates occurred to the components of it as described in D3.1, its grammar, and its Java classes that implements the data access objects and the services, and its database schema.

3.2.5.1 The Simulation Model changes

For the Simulation model, the following changes were made:

Renamed:

- module to *simulationComponent*;
- *simpleModule* to *simpleComponent*;
- *compoundModule* to *componentContainer*;

Deleted:

- *gate*;
- *connection*;

Added:

- *idGateway*;
- *simExpectedTrace*;

The grammar required for the parser was updated to fit the new model and the respective Java classes and database tables where created for the implementation of the parser and can be found in the Sections below.

3.2.5.2 Grammar

```
/*******CTTP Simulation Model******/

cttpSimulationModel: cttpSimulationModelTitle OPEN_PAREN  deploymentMode COMMA
simTool COMMA simTemplate COMMA simTime
COMMA (message)+ COMMA initSimTime COMMA simulationEnd COMMA executionSpeed COMMA
randomSeed COMMA (supportsPhaseRule)+
COMMA (consistsOfSimCompRule)+ COMMA (simulatesRule)+ CLOSE_PAREN;

cttpSimulationModelTitle : 'cttpSimulationmModel';
cttpSimulationModelID : Identifier;
deploymentMode: 'simTemplate' OPEN_PAREN STRING CLOSE_PAREN;
simTool: 'simTool' OPEN_PAREN STRING CLOSE_PAREN;
simTemplate: 'simTemplate' OPEN_PAREN STRING CLOSE_PAREN;
simTime: 'simTime' OPEN_PAREN DOUBLE CLOSE_PAREN;
initSimTime: 'initSimTime' OPEN_PAREN Date CLOSE_PAREN;
simulationEnd: 'simulationEnd' OPEN_PAREN Date CLOSE_PAREN;
executionSpeed: 'executionSpeed' OPEN_PAREN INT CLOSE_PAREN;
randomSeed: 'randomSpeed' OPEN_PAREN INT CLOSE_PAREN;
supportsPhaseRule: 'supportsPhase' OPEN_PAREN phaseID CLOSE_PAREN;
consistsOfSimCompRule: 'compriseOf' OPEN_PAREN simComponentID CLOSE_PAREN;
simulatesRule: 'simulations' OPEN_PAREN assetID CLOSE_PAREN;


/* Simulation Component*/
simComponent: simComponentTitle OPEN_PAREN simCommponentName COMMA
simComponentInternalID COMMA isRoot
COMMA (hasAttributeRule)+ COMMA (isPartOfCompContainerRule)* COMMA
(isPartOfCttpSimModelRule)+
(componentContainer (COMMA)*)* (simpleComponent (COMMA)*)* CLOSE_PAREN;

simComponentTitle: 'simComponent';
simComponentID: Identifier;
simCommponentName: STRING;
simComponentInternalID: STRING;
isRoot: bool;
hasAttributeRule: 'hasAttribute' OPEN_PAREN simAttributeID CLOSE_PAREN;
isPartOfCompContainerRule: 'isPartOfCompContainer' OPEN_PAREN componentContainerID
CLOSE_PAREN;
isPartOfCttpSimModelRule: 'isPartOfCttpSimModel' OPEN_PAREN cttpSimulationModelID
CLOSE_PAREN;

/* Attribute */
simAttribute: simAttributeTitle OPEN_PAREN simAttributeName COMMA
simAttributeValue CLOSE_PAREN;
simAttributeTitle: 'simAttribute';
simAttributeID: Identifier;
simAttributeName: 'name' OPEN_PAREN  STRING CLOSE_PAREN;
simAttributeValue: 'value' OPEN_PAREN STRING CLOSE_PAREN;

/* idGateway */
```

```
idGateway: idGatewayTitle OPEN_PAREN connectedTo CLOSE_PAREN;
idGatewayTitle: 'idGateway';
connectedTo: 'connectedTo' OPEN_PAREN STRING CLOSE_PAREN;

/*Component Container*/
componentContainer: componentContainerTitle OPEN_PAREN
containerConsistsOfSimCompRule CLOSE_PAREN
                |simComponent
                ;
componentContainerTitle : 'compoundModule';
componentContainerID : Identifier;
containerConsistsOfSimCompRule: 'consistsOfSimComp' OPEN_PAREN simComponentID
CLOSE_PAREN;


/*Simple Component*/
simpleComponent: simpleComponentTitle OPEN_PAREN  (handlesRule)*  CLOSE_PAREN;
simpleComponentTitle : 'simpleModule';
simpleModulebehavior : 'behavior' OPEN_PAREN STRING CLOSE_PAREN;
handlesRule : 'handles' OPEN_PAREN messageID CLOSE_PAREN;


/*Message*/
message: messageTitle OPEN_PAREN  messageName COMMA messageType COMMA
messageRecipient  COMMA messageSender
 COMMA messageArivalTime COMMA messagePriority COMMA messageStartTime CLOSE_PAREN;
messageTitle : 'message';
messageID : Identifier;
messageName: 'messageName' OPEN_PAREN STRING CLOSE_PAREN;
messageType: 'messageType' OPEN_PAREN MessageType CLOSE_PAREN;
messageRecipient: 'messageRecipient' OPEN_PAREN simComponentID CLOSE_PAREN;
messageSender: 'messageSender' OPEN_PAREN simComponentID CLOSE_PAREN;
messageArivalTime: 'messageArivalTime' OPEN_PAREN DOUBLE CLOSE_PAREN;
messagePriority: 'messagePriority' OPEN_PAREN INT CLOSE_PAREN;
messageStartTime: 'messageStartTime' OPEN_PAREN STRING CLOSE_PAREN;
```

3.2.5.3 Java Classes

3.2.5.3.1 CTTP Simulation Model

CTTP Simulation Model DAO

```
public interface CTTPSimulationModelDAO {

    public void insert(CTTPSimulationModel aModelElem);

    public void update(CTTPSimulationModel aModelElem);

    public void remove(int theId);

    public List<CTTPSimulationModel> findAll();

    public CTTPSimulationModel findById(int theId);

    public CTTPSimulationModel findByName(String theName);
}
```

CTTP Simulation Model DAO Implementation

```
@Repository
public class CTTPSimulationModelDAOImpl implements
ch.sphynx.Implementation.parser.dao.CTTPSimulationModelDAO {

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<CTTPSimulationModel> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<CTTPSimulationModel> assetList =  session.createQuery("from
CTTPSimulationModel").list();

            for(CTTPSimulationModel a: assetList){
                logger.info("CTTPSimulationModel List::" + a);
            }
            return assetList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public CTTPSimulationModel findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPSimulationModel a = (CTTPSimulationModel)
session.load(CTTPSimulationModel.class, new Integer(theId));
            logger.info("CTTPSimulationModel loaded successfully,
CTTPSimulationModel details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public CTTPSimulationModel findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from CTTPSimulationModel where name=:name");
            query.setParameter("name", theName);

            CTTPSimulationModel a = (CTTPSimulationModel) query.uniqueResult();
            logger.info("CTTPSimulationModel loaded successfully,
CTTPSimulationModel details="+ a);
            return a;
        }
```

```
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(CTTPSimulationModel CTTPSimulationModelObjectInsert) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(CTTPSimulationModelObjectInsert);
            logger.info("CTTPSimulationModel saved successfully,
CTTPSimulationModel Details="+ CTTPSimulationModelObjectInsert);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(CTTPSimulationModel CTTPSimulationModelObjectUpdate) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(CTTPSimulationModelObjectUpdate);
            logger.info("CTTPSimulationModel updated successfully,
CTTPSimulationModel Details="+ CTTPSimulationModelObjectUpdate);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPSimulationModel a = (CTTPSimulationModel)
session.load(CTTPSimulationModel.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("CTTPSimulationModel deleted successfully,
CTTPSimulationModel details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

CTTP Simulation Model Entity

```
@Entity
public class CTTPSimulationModel implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
```

```
    private long cttpSimulationModelID;
    private String tool;
    private String template;
    private long simTime;
    private long messageID;
    private Date initialSimTime;
    private Date simulationEnd;
    private int executionSpeed;
    private int randomSeed;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="cttpsimulationmodelsimulationcomponentjoin",
            joinColumns = @JoinColumn(name="cttpSimulationModelID", unique =
true),
            inverseJoinColumns = @JoinColumn(name="simulationComponentID", unique
= true))
    private Set<SimulationComponent> simulationComponentSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="eventsequencethreatjoin",
            joinColumns = @JoinColumn(name="cttpSimulationModelID", unique =
true),
            inverseJoinColumns = @JoinColumn(name="assetID", unique = true))
    private Set<Asset> assetSet;

    public long getCttpSimulationModelID() {
        return cttpSimulationModelID;
    }
    public void setCttpSimulationModelID(long cttpSimulationModelID) {
        this.cttpSimulationModelID = cttpSimulationModelID;
    }
    public String getTool() {
        return tool;
    }
    public void setTool(String tool) {
        this.tool = tool;
    }
    public String getTemplate() {
        return template;
    }
    public void setTemplate(String template) {
        this.template = template;
    }
    public long getSimTime() {
        return simTime;
    }
    public void setSimTime(long simTime) {
        this.simTime = simTime;
    }
    public long getMessageID() {
        return messageID;
    }
    public void setMessageID(long messageID) {
        this.messageID = messageID;
    }
    public Date getInitialSimTime() {
        return initialSimTime;
    }
    public void setInitialSimTime(Date initialSimTime) {
        this.initialSimTime = initialSimTime;
```

```
    }
    public Date getSimulationEnd() {
        return simulationEnd;
    }
    public void setSimulationEnd(Date simulationEnd) {
        this.simulationEnd = simulationEnd;
    }
    public int getExecutionSpeed() {
        return executionSpeed;
    }
    public void setExecutionSpeed(int executionSpeed) {
        this.executionSpeed = executionSpeed;
    }
    public int getRandomSeed() {
        return randomSeed;
    }
    public void setRandomSeed(int randomSeed) {
        this.randomSeed = randomSeed;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof CTTPSimulationModel)) return false;
        CTTPSimulationModel that = (CTTPSimulationModel) o;
        return getCttpSimulationModelID() == that.getCttpSimulationModelID() &&
                getSimTime() == that.getSimTime() &&
                getMessageID() == that.getMessageID() &&
                getExecutionSpeed() == that.getExecutionSpeed() &&
                getRandomSeed() == that.getRandomSeed() &&
                Objects.equals(getTool(), that.getTool()) &&
                Objects.equals(getTemplate(), that.getTemplate()) &&
                Objects.equals(getInitialSimTime(), that.getInitialSimTime()) &&
                Objects.equals(getSimulationEnd(), that.getSimulationEnd());
    }
    @Override
    public int hashCode() {
        return Objects.hash(getCttpSimulationModelID(), getTool(), getTemplate(),
getSimTime(), getMessageID(), getInitialSimTime(), getSimulationEnd(),
getExecutionSpeed(), getRandomSeed());
    }
}
```

CTTP Simulation Model Service

```
public interface CTTPSimulationModelService {
    void insert(CTTPSimulationModel aModelElem);

    void update(CTTPSimulationModel aModelElem);

    void remove(int theId);

    List<CTTPSimulationModel> findAll();

    CTTPSimulationModel findById(int theId);

    CTTPSimulationModel findByName(String theName);
}
```

CTTP Simulation Model Service Implementation

```
@Repository
public class CTTPSimulationModelServiceImpl implements CTTPSimulationModelService
{
    @Autowired
    private CTTPSimulationModelDAO objectDAO ;

    @Autowired
    public CTTPSimulationModelServiceImpl(CTTPSimulationModelDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(CTTPSimulationModel aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(CTTPSimulationModel aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<CTTPSimulationModel> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public CTTPSimulationModel findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public CTTPSimulationModel findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.2 Simulation Component

Simulation Component DAO Implementation

```
@Repository
public class SimulationComponentDAOImpl implements SimulationComponentDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;
```

```
    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<SimulationComponent> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<SimulationComponent> SimulationComponentList =
session.createQuery("from SimulationComponent").list();

            for(SimulationComponent a: SimulationComponentList){
                logger.info("SimulationComponent List::" + a);
            }
            return SimulationComponentList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public SimulationComponent findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SimulationComponent a = (SimulationComponent)
session.load(SimulationComponent.class, new Integer(theId));
            logger.info("SimulationComponent loaded successfully,
SimulationComponent details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public SimulationComponent findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from SimulationComponent where name=:name");
            query.setParameter("name", theName);

            SimulationComponent a = (SimulationComponent) query.uniqueResult();
            logger.info("SimulationComponent loaded successfully,
SimulationComponent details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(SimulationComponent SimulationComponentElement) {
```

```
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(SimulationComponentElement);
            logger.info("SimulationComponent saved successfully,
SimulationComponent Details="+ SimulationComponentElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(SimulationComponent SimulationComponent) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(SimulationComponent);
            logger.info("SimulationComponent updated successfully,
SimulationComponent Details="+ SimulationComponent);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SimulationComponent a = (SimulationComponent)
session.load(SimulationComponent.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("SimulationComponent deleted successfully,
SimulationComponent details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Simulation Component Entity

```
@Entity
public class SimulationComponent implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long simulationComponentID;
    private String name;
    private String id;
    private String type;
    private Boolean isRoot;
    private long typeID;
```

```
    public long getSimulationComponentID() {
        return simulationComponentID;
    }

    public void setSimulationComponentID(long simulationComponentID) {
        this.simulationComponentID = simulationComponentID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Boolean getRoot() {
        return isRoot;
    }

    public void setRoot(Boolean root) {
        isRoot = root;
    }

    public long getTypeID() {
        return typeID;
    }

    public void setTypeID(long typeID) {
        this.typeID = typeID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SimulationComponent)) return false;
        SimulationComponent that = (SimulationComponent) o;
        return getSimulationComponentID() == that.getSimulationComponentID() &&
                getTypeID() == that.getTypeID() &&
                Objects.equals(getName(), that.getName()) &&
                Objects.equals(getId(), that.getId()) &&
                Objects.equals(getType(), that.getType()) &&
                Objects.equals(isRoot, that.isRoot);
    }
```

```
    @Override
    public int hashCode() {
        return Objects.hash(getSimulationComponentID(), getName(), getId(),
getType(), isRoot, getTypeID());
    }
}
```

Simulation Component Service Implementation

```
@Repository
public class SimulationComponentServiceImpl implements SimulationComponentService
{
    @Autowired
    private SimulationComponentDAO objectDAO ;

    @Autowired
    public SimulationComponentServiceImpl(SimulationComponentDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(SimulationComponent aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(SimulationComponent aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<SimulationComponent> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public SimulationComponent findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public SimulationComponent findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.3 Simple Component

Simple Component DAO Implementation

```
@Repository
public class SimpleComponentDAOImpl implements SimpleComponentDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<SimpleComponent> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<SimpleComponent> SimpleComponentList =  session.createQuery("from
SimpleComponent").list();

            for(SimpleComponent a: SimpleComponentList){
                logger.info("SimpleComponent List::" + a);
            }
            return SimpleComponentList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public SimpleComponent findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SimpleComponent a = (SimpleComponent)
session.load(SimpleComponent.class, new Integer(theId));
            logger.info("SimpleComponent loaded successfully, SimpleComponent
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public SimpleComponent findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from SimpleComponent where name=:name");
            query.setParameter("name", theName);

            SimpleComponent a = (SimpleComponent) query.uniqueResult();
            logger.info("SimpleComponent loaded successfully, SimpleComponent
```

```
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(SimpleComponent SimpleComponentElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(SimpleComponentElement);
            logger.info("SimpleComponent saved successfully, SimpleComponent
Details="+ SimpleComponentElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(SimpleComponent SimpleComponent) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(SimpleComponent);
            logger.info("SimpleComponent updated successfully, SimpleComponent
Details="+ SimpleComponent);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SimpleComponent a = (SimpleComponent)
session.load(SimpleComponent.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("SimpleComponent deleted successfully, SimpleComponent
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Simple Component Entity

```
@Entity
public class SimpleComponent implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long simpleComponentID;
    private String behaviour;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="simplecomponentmessage",
            joinColumns = @JoinColumn(name="simpleComponentID", unique = true),
            inverseJoinColumns = @JoinColumn(name="messageID", unique = true))
    private Set<Message> messageSet;

    public long getSimpleComponentID() {
        return simpleComponentID;
    }

    public void setSimpleComponentID(long simpleComponentID) {
        this.simpleComponentID = simpleComponentID;
    }

    public String getBehaviour() {
        return behaviour;
    }

    public void setBehaviour(String behaviour) {
        this.behaviour = behaviour;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SimpleComponent)) return false;
        SimpleComponent that = (SimpleComponent) o;
        return getSimpleComponentID() == that.getSimpleComponentID() &&
                Objects.equals(getBehaviour(), that.getBehaviour());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getSimpleComponentID(), getBehaviour());
    }
}
```

Simple Component Service Implementation

```
@Repository
public class SimpleComponentServiceImpl implements SimpleComponentService {
    @Autowired
    private SimpleComponentDAO objectDAO ;

    @Autowired
    public SimpleComponentServiceImpl(SimpleComponentDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(SimpleComponent aModelElem) {
```

```
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(SimpleComponent aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<SimpleComponent> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public SimpleComponent findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public SimpleComponent findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.4 Component Container

Component Container DAO Implementation

```
@Repository
public class ComponentContainerDAOImpl implements ComponentContainerDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<ComponentContainer> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<ComponentContainer> ComponentContainerList =
session.createQuery("from ComponentContainer").list();

            for(ComponentContainer a: ComponentContainerList){
                logger.info("ComponentContainer List::" + a);
            }
            return ComponentContainerList;
```

```
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public ComponentContainer findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            ComponentContainer a = (ComponentContainer)
session.load(ComponentContainer.class, new Integer(theId));
            logger.info("ComponentContainer loaded successfully,
ComponentContainer details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public ComponentContainer findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from ComponentContainer where name=:name");
            query.setParameter("name", theName);

            ComponentContainer a = (ComponentContainer) query.uniqueResult();
            logger.info("ComponentContainer loaded successfully,
ComponentContainer details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(ComponentContainer ComponentContainerElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ComponentContainerElement);
            logger.info("ComponentContainer saved successfully, ComponentContainer
Details="+ ComponentContainerElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(ComponentContainer ComponentContainer) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
```

```
            session.update(ComponentContainer);
            logger.info("ComponentContainer updated successfully,
ComponentContainer Details="+ ComponentContainer);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            ComponentContainer a = (ComponentContainer)
session.load(ComponentContainer.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("ComponentContainer deleted successfully,
ComponentContainer details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Component Container Entity

```
@Entity
public class ComponentContainer implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long componentContainerID;

    public long getComponentContainerID() {
        return componentContainerID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ComponentContainer)) return false;
        ComponentContainer that = (ComponentContainer) o;
        return getComponentContainerID() == that.getComponentContainerID();
    }

    @Override
    public int hashCode() {
        return Objects.hash(getComponentContainerID());
    }
}
```

Component Container Service Implementation

```
@Repository
public class ComponentContainerServiceImpl implements ComponentContainerService {
    @Autowired
    private ComponentContainerDAO objectDAO ;

    @Autowired
    public ComponentContainerServiceImpl(ComponentContainerDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(ComponentContainer aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(ComponentContainer aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<ComponentContainer> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public ComponentContainer findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public ComponentContainer findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.5 Simulation Expected Trace

Simulation Expected Trace DAO Implementation

```
@Repository
public class SimExpectedTraceDAOImpl implements SimExpectedTraceDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
```

```
    }

    @Override
    public List<SimExpectedTrace> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<SimExpectedTrace> SimExpectedTraceList =
session.createQuery("from SimExpectedTrace").list();

            for(SimExpectedTrace a: SimExpectedTraceList){
                logger.info("SimExpectedTrace List::" + a);
            }
            return SimExpectedTraceList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public SimExpectedTrace findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SimExpectedTrace a = (SimExpectedTrace)
session.load(SimExpectedTrace.class, new Integer(theId));
            logger.info("SimExpectedTrace loaded successfully, SimExpectedTrace
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public SimExpectedTrace findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from SimExpectedTrace where name=:name");
            query.setParameter("name", theName);

            SimExpectedTrace a = (SimExpectedTrace) query.uniqueResult();
            logger.info("SimExpectedTrace loaded successfully, SimExpectedTrace
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(SimExpectedTrace SimExpectedTraceElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
```

```
            session.persist(SimExpectedTraceElement);
            logger.info("SimExpectedTrace saved successfully, SimExpectedTrace
Details="+ SimExpectedTraceElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(SimExpectedTrace SimExpectedTrace) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(SimExpectedTrace);
            logger.info("SimExpectedTrace updated successfully, SimExpectedTrace
Details="+ SimExpectedTrace);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SimExpectedTrace a = (SimExpectedTrace)
session.load(SimExpectedTrace.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("SimExpectedTrace deleted successfully, SimExpectedTrace
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Simulation Expected Trace Entity

```
@Entity
public class SimExpectedTrace implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long assetID;
    private String value;
    private long simModelID;

    public long getAssetID() {
        return assetID;
    }

    public void setAssetID(long assetID) {
        this.assetID = assetID;
```

```
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public long getSimModelID() {
        return simModelID;
    }

    public void setSimModelID(long simModelID) {
        this.simModelID = simModelID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SimExpectedTrace)) return false;
        SimExpectedTrace that = (SimExpectedTrace) o;
        return getAssetID() == that.getAssetID() &&
                getSimModelID() == that.getSimModelID() &&
                Objects.equals(getValue(), that.getValue());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getAssetID(), getValue(), getSimModelID());
    }
}
```

Simulation Expected Trace Service Implementation

```
@Repository
public class SimExpectedTraceServiceImpl implements SimExpectedTraceService {
    @Autowired
    private SimExpectedTraceDAO objectDAO ;

    @Autowired
    public SimExpectedTraceServiceImpl(SimExpectedTraceDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(SimExpectedTrace aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(SimExpectedTrace aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
```

```
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<SimExpectedTrace> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public SimExpectedTrace findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public SimExpectedTrace findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.6 IdGateway

IdGateway DAO Implementation

```
@Repository
public class IdGatewayDAOImpl implements IdGatewayDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<IdGateway> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<IdGateway> IdGatewayList =  session.createQuery("from
IdGateway").list();

            for(IdGateway a: IdGatewayList){
                logger.info("IdGateway List::" + a);
            }
            return IdGatewayList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public IdGateway findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
```

```
            IdGateway a = (IdGateway) session.load(IdGateway.class, new
Integer(theId));
            logger.info("IdGateway loaded successfully, IdGateway details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public IdGateway findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from IdGateway where name=:name");
            query.setParameter("name", theName);

            IdGateway a = (IdGateway) query.uniqueResult();
            logger.info("IdGateway loaded successfully, IdGateway details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(IdGateway IdGatewayElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(IdGatewayElement);
            logger.info("IdGateway saved successfully, IdGateway Details="+
IdGatewayElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(IdGateway IdGateway) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(IdGateway);
            logger.info("IdGateway updated successfully, IdGateway Details="+
IdGateway);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
```

```
            Session session = this.sessionFactory.getCurrentSession();
            IdGateway a = (IdGateway) session.load(IdGateway.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("IdGateway deleted successfully, IdGateway details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

IdGateway Entity

```
@Entity
public class IdGateway implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long idGatewayID;
    private String connectedTo;

    public long getIdGatewayID() {
        return idGatewayID;
    }

    public void setIdGatewayID(long idGatewayID) {
        this.idGatewayID = idGatewayID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof IdGateway)) return false;
        IdGateway idGateway = (IdGateway) o;
        return getIdGatewayID() == idGateway.getIdGatewayID() &&
                Objects.equals(getConnectedTo(), idGateway.getConnectedTo());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getIdGatewayID(), getConnectedTo());
    }

    public String getConnectedTo() {
        return connectedTo;
    }

    public void setConnectedTo(String connectedTo) {
        this.connectedTo = connectedTo;
    }
}
```

IdGateway Service Implementation

```
@Repository
public class IdGatewayServiceImpl implements IdGatewayService {
    @Autowired
    private IdGatewayDAO objectDAO ;

    @Autowired
    public IdGatewayServiceImpl(IdGatewayDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(IdGateway aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(IdGateway aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<IdGateway> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public IdGateway findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public IdGateway findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.7 Message

Message DAO Implementation

```
@Repository
public class MessageDAOImpl implements MessageDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
```

```
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Message> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Message> MessageList =  session.createQuery("from
Message").list();

            for(Message a: MessageList){
                logger.info("Message List::" + a);
            }
            return MessageList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Message findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Message a = (Message) session.load(Message.class, new Integer(theId));
            logger.info("Message loaded successfully, Message details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Message findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Message where name=:name");
            query.setParameter("name", theName);

            Message a = (Message) query.uniqueResult();
            logger.info("Message loaded successfully, Message details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Message MessageElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(MessageElement);
            logger.info("Message saved successfully, Message Details="+
```

```
MessageElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Message Message) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Message);
            logger.info("Message updated successfully, Message Details="+
Message);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Message a = (Message) session.load(Message.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Message deleted successfully, Message details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Message Model Entity

```
@Entity
public class Message implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long messageID;
    private String name;
    @Id
    private long messageTypeID;
    @Id
    private long sendingSimpleCompID;
    @Id
    private long receivingSimpleCompID;
    private long arrivalTime;
    private int priority;
    private String start;

    @ManyToMany(mappedBy = "messageID",cascade = CascadeType.REFRESH, fetch =
```

```
FetchType.LAZY)
    private Set<SimpleComponent> simpleComponentSet;

    public long getMessageID() {
        return messageID;
    }

    public void setMessageID(long messageID) {
        this.messageID = messageID;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public long getMessageTypeID() {
        return messageTypeID;
    }

    public void setMessageTypeID(long messageTypeID) {
        this.messageTypeID = messageTypeID;
    }

    public long getSendingSimpleCompID() {
        return sendingSimpleCompID;
    }

    public void setSendingSimpleCompID(long sendingSimpleCompID) {
        this.sendingSimpleCompID = sendingSimpleCompID;
    }

    public long getReceivingSimpleCompID() {
        return receivingSimpleCompID;
    }

    public void setReceivingSimpleCompID(long receivingSimpleCompID) {
        this.receivingSimpleCompID = receivingSimpleCompID;
    }

    public long getArrivalTime() {
        return arrivalTime;
    }

    public void setArrivalTime(long arrivalTime) {
        this.arrivalTime = arrivalTime;
    }

    public int getPriority() {
        return priority;
    }

    public void setPriority(int priority) {
        this.priority = priority;
    }

    public String getStart() {
```

```
        return start;
    }

    public void setStart(String start) {
        this.start = start;
    }

    public Set<SimpleComponent> getSimpleComponentSet() {
        return simpleComponentSet;
    }

    public void setSimpleComponentSet(Set<SimpleComponent> simpleComponentSet) {
        this.simpleComponentSet = simpleComponentSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Message)) return false;
        Message message = (Message) o;
        return getMessageID() == message.getMessageID() &&
                getMessageTypeID() == message.getMessageTypeID() &&
                getSendingSimpleCompID() == message.getSendingSimpleCompID() &&
                getReceivingSimpleCompID() == message.getReceivingSimpleCompID()
&&
                getArrivalTime() == message.getArrivalTime() &&
                getPriority() == message.getPriority() &&
                Objects.equals(getName(), message.getName()) &&
                Objects.equals(getStart(), message.getStart()) &&
                Objects.equals(getSimpleComponentSet(),
message.getSimpleComponentSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getMessageID(), getName(), getMessageTypeID(),
getSendingSimpleCompID(), getReceivingSimpleCompID(), getArrivalTime(),
getPriority(), getStart(), getSimpleComponentSet());
    }
}
```

Message Service Implementation

```
@Repository
public class MessageServiceImpl implements MessageService {
    @Autowired
    private MessageDAO objectDAO ;

    @Autowired
    public MessageServiceImpl(MessageDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Message aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
```

```
    @Transactional
    public void update(Message aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Message> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Message findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Message findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.3.8 Attribute

Attribute DAO Implementation

```
@Repository
public class AttributeDAOImpl implements AttributeDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Attribute> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Attribute> AttributeList =  session.createQuery("from
Attribute").list();

            for(Attribute a: AttributeList){
                logger.info("Attribute List::" + a);
            }
            return AttributeList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
```

```
        }
    }

    @Override
    public Attribute findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Attribute a = (Attribute) session.load(Attribute.class, new
Integer(theId));
            logger.info("Attribute loaded successfully, Attribute details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Attribute findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Attribute where name=:name");
            query.setParameter("name", theName);

            Attribute a = (Attribute) query.uniqueResult();
            logger.info("Attribute loaded successfully, Attribute details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Attribute AttributeElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(AttributeElement);
            logger.info("Attribute saved successfully, Attribute Details="+
AttributeElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Attribute Attribute) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Attribute);
            logger.info("Attribute updated successfully, Attribute Details="+
Attribute);
        }
        catch (Exception e){
            logger.error(e.getMessage());
```

```
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Attribute a = (Attribute) session.load(Attribute.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Attribute deleted successfully, Attribute details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Attribute Entity

```
@Entity
@Table(name = "attribute")
public class Attribute implements java.io.Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "attributeID", nullable = false)
    private long attributeID;
    @Id
    private long simComponentID;
    @Id
    private long gatewayID;
    private String name;
    private String value;

    @Id
    @Column(name = "attributeID", nullable = false)
    public long getAttributeID() {
        return attributeID;
    }

    public void setAttributeID(long attributeID) {
        this.attributeID = attributeID;
    }

    @Id
    @Column(name = "componentID")
    public long getSimComponentID() {
        return simComponentID;
    }

    public void setSimComponentID(long simComponentID) {
        this.simComponentID = simComponentID;
    }
```

```
    @Id
    @Column(name = "gatewayID", nullable = false)
    public long getGatewayID() {
        return gatewayID;
    }

    public void setGatewayID(long gatewayID) {
        this.gatewayID = gatewayID;
    }

    @Basic
    @Column(name = "name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Basic
    @Column(name = "value")
    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Attribute)) return false;
        Attribute attribute = (Attribute) o;
        return getAttributeID() == attribute.getAttributeID() &&
                getSimComponentID() == attribute.getSimComponentID() &&
                getGatewayID() == attribute.getGatewayID() &&
                Objects.equals(getName(), attribute.getName()) &&
                Objects.equals(value, attribute.value);
    }

    @Override
    public int hashCode() {
        return Objects.hash(getAttributeID(), getSimComponentID(), getGatewayID(),
getName(), value);
    }
}
```

Attribute Service Implementation

```
@Repository
public class AttributeServiceImpl implements AttributeService {
    @Autowired
    private AttributeDAO objectDAO ;

    @Autowired
    public AttributeServiceImpl(AttributeDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }
```

```
    @Override
    @Transactional
    public void insert(Attribute aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Attribute aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Attribute> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Attribute findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Attribute findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.5.4 Database

Based on the above, a database structure was created to illustrate the simulation database tables, as depicted in Figure 7.

Figure 7: Simulation Model Database schema

3.2.6 The Emulation parser

The subsections below present the emulation parser, the updates occurred to the components of it as described in D3.1, its grammar, and its java classes that implements the data access objects and the services, and its database schema.

3.2.6.1 The CTTP Emulation Model Updates

For the Simulation model, the following changes were made:

Deleted:

- *virtualNetworkModule*;

- *softwareAsset* (deleted only from the CTTP Emulation Model, not from the core model);
- *virtualNetworkAdapter*

Added:

- *networks*;
- *network*;
- *scripts*;
- *script*;
- *routers*;
- *router*;
- *customVM*;
- *emExpectedTrace*;

The grammar required for the parser was updated to fit the new model and the respective Java classes and database tables where created for the implementation of the parser and can be found in the Sections below.

3.2.6.2 The Emulation tool input parameters

The Emulation Tool has been designed to deploy the training scenario starting from the description provided by the CTTP Models. In particular, the tool will use the Emulation Sub-model described in this subsection.

The Emulation Controller and the Emulation Compiler modules included in the Emulation Tool are the modules responsible of, respectively, the communication with the other THREAT-ARREST tools, and of the parsing, compiling, and finalization of the model in the YAML file (YAML, 2019) that will be used to deploy the scenario.

The Controller provides a REST interface that will be used by the Training Tool. In particular, the method `/emulation/getVMfromXML` accepts as input the eXtensible Markup Language (XML) description of the scenario to deploy and returns the set of *username* and *password* the tool should use to access the VMs. The first version of the compiling algorithms is explained in details in the deliverables "D2.1 – Emulation Components Generator Modules v1" and "D2.3 – Interlinking od emulated components module v1".

Figure 8 presents an example of XML files containing all the elements provided in the Emulation Sub-model, which is able to deploy a network composed by a Linux and a Windows machine connected to two virtual networks. The two networks are then connected using a virtual router. Figure 8 shows this Emulation Sub-model that will be the source of the input XML.

The XML is composed by the following four sections:

- `CustomVM`, with the configuration each single virtual machine (VM). The section contains the following elements and properties:
  - `name`: Name of the VM as will be deployed in OpenStack;
  - `os`: Type of operating system (Windows or Linux);
  - `Connectionmode`: specification of the connection to be used to remotely connect to the VM; in general `port = 22` and `connectiontype=SSH` for connection with

Linux VM or port=3389 and connectiontype=rdp to connect with Windows machines;

- ram: size of the memory to allocate to the VM (in Kbyte);
- disk: size in GB of the disk to associate to the VM;
- image: name of the image to deploy, used as index in the OpenStack repository along with username and, in case of a Windows VM, password;
- network: virtual network the VM is connected to. Optionally a fixed address can be provided (fixedip); we can have one or more network elements;
- scripts: id of the script element to associate to the VM that will be run during the boot (optional).

- Networks (the definition of the virtual networks to deploy):
  - network: id of the network, to be referred in the CustomVM network element;
  - gateway: name and address of the default gateway associated to the subnet;
  - cidr: name and subnet to be associate to the subnet;
  - is_external: boolean element to declare if the network has to be considered external or internal.
- Routers, that describe the network elements that connect two networks, thus defining the underlying network topology:
  - router: id, image, flavour (OpenStack description of the virtual machine characteristic, by default m1.small), os, username, connectionport, and connectiontype to define the machine that will acts as router (only the name is requested, all the other parameters are optional, unless a special router is requested);
  - routerinterface: the two networks the router is connected to, defined in the Networks section;
  - scripts: the id of the related script (optional).
- Scripts, containing the scripts to be run at the boot of the related VM:
  - script: id of the script, to be referred in the CustomVM and Routers sections;
  - Script code, that can be a Bash script, in case of Linux machine, or a Shell or PowerShell script in case of Windows VM.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Scenario name="TA-mix">
    <CustomVM name="lamp" os="linux">
        <connectionmode port="22" connectiontype="ssh"></connectionmode>
        <ram val="2048"></ram>
        <vcpus val="1"></vcpus>
        <disk val="20"></disk>
        <image name="img1" val="ubuntu-lamp-red" username="ubuntu"></image>
        <Network idref="dmz" fixedip="10.20.30.25"/>
        <Scripts idref="install_script" />
    </CustomVM>
    <CustomVM name="win7" os="windows">
        <connectionmode port="3389" connectiontype="rdp"></connectionmode>
        <ram val="2048"></ram>
        <vcpus val="2"></vcpus>
        <disk val="20"></disk>
        <image name="img2" val="win7-red" username="win7" password="win7"></image>
        <Network idref="esternal" />
    </CustomVM>
   <Networks>
        <Network id="dmz">
            <gateway name="gateway-dmz" val="10.20.30.1"></gateway>
            <cidr name="cidr-dmz" val="10.20.30.0/24"></cidr>
            <is_external val="false"></is_external>
        </Network>
        <Network id="esternal" >
            <gateway name="gateway-esternal" val="10.20.40.1"></gateway>
            <cidr name="cidr-esternal" val="10.20.40.0/24"></cidr>
            <is_external val="true"></is_external>
        </Network>
   </Networks>
    <Routers>
        <Router id="router_1" img="ubuntu-router-red" flavor="m1.small" os="linux"
            username="ubuntu" port="22" connectiontype="ssh" >
            <routerInterface network_1="dmz" network_2="esternal"></routerInterface>
        </Router>
    </Routers>
    <Scripts>
        <Script id="install_script" ><![CDATA[
echo "start user data"
sudo wget -O /tmp/sqli_noparam.sh http://172.20.28.138:8088/demo/ex-sqli/sqli_noparam.sh
sudo chmod +x /tmp/sqli_noparam.sh
sudo /tmp/sqli_noparam.sh
echo "end user data"
]]>
        </Script>
    </Scripts>
</Scenario>
```

Figure 8: XML deployment file extracted from the Emulation Sub-model.

3.2.6.3 Grammar

```
*********CTTP Emulation Model*********


cttpEmulationModel: cttpEmulationModelTitle OPEN_PAREN (emTemplate)? COMMA
emModelName
 COMMA (facilitatesPhaseRule)+ COMMA (emulatesRule)+
(hasCustomVMRule)* COMMA (hasEmNetworksRule)* COMMA (hasEmRouterRule)* COMMA
(hasEmScriptRule)* CLOSE_PAREN;
```

```
cttpEmulationModelTitle : 'cttpEmulationModel';
cttpEmulationModelID : Identifier;
emTemplate: 'emTemplate' OPEN_PAREN STRING CLOSE_PAREN;
emModelName: 'emModelName' OPEN_PAREN STRING CLOSE_PAREN;

hasEmNetworksRule: 'hasEmNetwork' OPEN_PAREN emNetworksID CLOSE_PAREN;
hasEmRouterRule: 'hasEmmRouter' OPEN_PAREN routersID CLOSE_PAREN;
hasEmScriptRule: 'hasEmScript' OPEN_PAREN emScriptsID  CLOSE_PAREN;
hasCustomVMRule: 'hasCustomVM' OPEN_PAREN customVMID CLOSE_PAREN;

facilitatesPhaseRule: 'facilitatesPhase' OPEN_PAREN phaseID CLOSE_PAREN;
emulatesRule: 'emulates' OPEN_PAREN assetID CLOSE_PAREN;

/*Emulation networks*/
emNetworks: emNetworksTitle OPEN_PAREN (containsEmNetworkRule)+ CLOSE_PAREN;
emNetworksTitle: 'emNetworks';
emNetworksID: Identifier;
containsEmNetworkRule: 'containsEmNetwork' OPEN_PAREN emNetworkID CLOSE_PAREN;

/*Emulation network*/
emNetwork: emNetworkTitle OPEN_PAREN emNetworkInternalID COMMA emNetworkCIDR COMMA
isExternal COMMA emGateway CLOSE_PAREN;
emNetworkTitle: 'emNetworks';
emNetworkID: Identifier;
emNetworkInternalID: 'emNetworkInternalID' OPEN_PAREN STRING CLOSE_PAREN;
emNetworkCIDR: 'emNetworkCIDR' OPEN_PAREN cidr CLOSE_PAREN;
isExternal: 'isExternal' OPEN_PAREN bool CLOSE_PAREN;
emGateway: 'emGateway' OPEN_PAREN gatewayType CLOSE_PAREN;


/*Scripts*/
emScripts: emScriptsTitle OPEN_PAREN (containScriptRule)* CLOSE_PAREN;
emScriptsTitle: 'emScripts';
emScriptsID: Identifier;
containScriptRule: 'containScript' OPEN_PAREN emScriptID CLOSE_PAREN;

/*Script*/
emScript: emScriptTitle OPEN_PAREN emScriptInternalID COMMA emScriptBody
CLOSE_PAREN;
emScriptTitle: 'emScript';
emScriptID: Identifier;
emScriptInternalID: 'emScriptInternalID' OPEN_PAREN STRING CLOSE_PAREN;
emScriptBody: 'emScriptBody' OPEN_PAREN STRING CLOSE_PAREN;


/*Custom VM*/
customVM: customVMTitle OPEN_PAREN vmName COMMA vmOS COMMA vmConnectionMode COMMA
vmRam COMMA
vmDisk COMMA vmImage COMMA vmNetwork COMMA vmScriptID COMMA vcpus  CLOSE_PAREN;
customVMTitle: 'customVM';
customVMID: Identifier;
vmName: 'vmName' OPEN_PAREN STRING CLOSE_PAREN;
vmOS: 'vmOS' OPEN_PAREN STRING CLOSE_PAREN;
vmConnectionMode: 'connectionMode' OPEN_PAREN connectionMode CLOSE_PAREN;
vmRam: 'vmRam' OPEN_PAREN INT CLOSE_PAREN;
vmDisk: 'vmDisk' OPEN_PAREN INT CLOSE_PAREN;
vmImage: 'vmImage' OPEN_PAREN imageType CLOSE_PAREN;
vmNetwork: 'vmNetwork' OPEN_PAREN networkType CLOSE_PAREN;
vmScriptID: 'vmScriptID' OPEN_PAREN STRING CLOSE_PAREN;
```

```
vcpus: 'vcpus' OPEN_PAREN INT CLOSE_PAREN;


/*Routers*/
routers: routersTitle OPEN_PAREN (containsRouterRule)* CLOSE_PAREN;
routersTitle: 'routers';
routersID: Identifier;
containsRouterRule: 'containsRouter' OPEN_PAREN routerID CLOSE_PAREN;

/*Router*/
router: routersTitle OPEN_PAREN routerInternalID COMMA routerImg COMMA (flavor)?
COMMA (routerOS)? COMMA (routerUsername)? COMMA
(routerPort)? COMMA (routerConnectionType)? COMMA routerInterface COMMA
routerScriptID CLOSE_PAREN;
routerTitle: 'routers';
routerID: Identifier;
routerInternalID: 'routerInternalID' OPEN_PAREN STRING CLOSE_PAREN;
routerImg: 'routerImg' OPEN_PAREN STRING CLOSE_PAREN;
flavor: 'flavor' OPEN_PAREN STRING CLOSE_PAREN;
routerOS: 'routerOS' OPEN_PAREN STRING CLOSE_PAREN;
routerUsername: 'routerUsername' OPEN_PAREN STRING CLOSE_PAREN;
routerPort: 'routerPort' OPEN_PAREN INT CLOSE_PAREN;
routerConnectionType: 'routerConnectionType' OPEN_PAREN STRING CLOSE_PAREN;
routerInterface: 'routerInterface' OPEN_PAREN riType CLOSE_PAREN;
routerScriptID: 'vmName' OPEN_PAREN STRING CLOSE_PAREN;
```

3.2.6.4 Java Classes

3.2.6.4.1 CTTP Emulation Model

CTTP Emulation Model DAO

```
public interface CTTPEmulationModelDAO {

    public void insert(CTTPEmulationModel aModelElem);

    public void update(CTTPEmulationModel aModelElem);

    public void remove(int theId);

    public List<CTTPEmulationModel> findAll();

    public CTTPEmulationModel findById(int theId);

    public CTTPEmulationModel findByName(String theName);
}
```

CTTP Emulation Model DAO Implementation

```
@Repository
public class CTTPEmulationModelDAOImpl implements CTTPEmulationModelDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
```

```
    @Override
    public List<CTTPEmulationModel> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<CTTPEmulationModel> CTTPEmulationModelList =
session.createQuery("from CTTPEmulationModel").list();

            for(CTTPEmulationModel a: CTTPEmulationModelList){
                logger.info("CTTPEmulationModel List::" + a);
            }
            return CTTPEmulationModelList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public CTTPEmulationModel findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPEmulationModel a = (CTTPEmulationModel)
session.load(CTTPEmulationModel.class, new Integer(theId));
            logger.info("CTTPEmulationModel loaded successfully,
CTTPEmulationModel details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public CTTPEmulationModel findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from CTTPEmulationModel where name=:name");
            query.setParameter("name", theName);

            CTTPEmulationModel a = (CTTPEmulationModel) query.uniqueResult();
            logger.info("CTTPEmulationModel loaded successfully, Asset details="+
a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(CTTPEmulationModel CTTPEmulationModelObjectInsert) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(CTTPEmulationModelObjectInsert);
            logger.info("CTTPEmulationModel saved successfully, CTTPEmulationModel
```

```
Details="+ CTTPEmulationModelObjectInsert);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(CTTPEmulationModel CTTPEmulationModelObjectUpdate) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(CTTPEmulationModelObjectUpdate);
            logger.info("CTTPEmulationModel updated successfully,
CTTPEmulationModel Details="+ CTTPEmulationModelObjectUpdate);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPEmulationModel a = (CTTPEmulationModel)
session.load(CTTPEmulationModel.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Asset deleted successfully, asset details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

CTTP Emulation Model Entity

```
@Entity
public class CTTPEmulationModel implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long cttpEmulationModelID;
    private String name;
    private String template;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="cttpemmodelnetworksjoin",
            joinColumns = @JoinColumn(name="cttpEmulationModelID", unique = true),
            inverseJoinColumns = @JoinColumn(name="networksID", unique = true))
    private Set<Networks> networksSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="cttpemmodelroutersjoin",
```

```
            joinColumns = @JoinColumn(name="cttpEmulationModelID", unique = true),
            inverseJoinColumns = @JoinColumn(name="routersID", unique = true))
    private Set<Routers> routersSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="cttpemmodelcustomvmjoin",
            joinColumns = @JoinColumn(name="cttpEmulationModelID", unique = true),
            inverseJoinColumns = @JoinColumn(name="customVMID", unique = true))
    private Set<CustomVM> customVMSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="cttpemmodelscriptsjoin",
            joinColumns = @JoinColumn(name="cttpEmulationModelID", unique = true),
            inverseJoinColumns = @JoinColumn(name="scriptsID", unique = true))
    private Set<Scripts> scriptsSet;

    public long getCttpEmulationModelID() {
        return cttpEmulationModelID;
    }

    public void setCttpEmulationModelID(long cttpEmulationModelID) {
        this.cttpEmulationModelID = cttpEmulationModelID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getTemplate() {
        return template;
    }

    public void setTemplate(String template) {
        this.template = template;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof CTTPEmulationModel)) return false;
        CTTPEmulationModel that = (CTTPEmulationModel) o;
        return getCttpEmulationModelID() == that.getCttpEmulationModelID() &&
                Objects.equals(getName(), that.getName()) &&
                Objects.equals(getTemplate(), that.getTemplate());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getCttpEmulationModelID(), getName(), getTemplate());
    }
}
```

CTTP Emulation Model Service

```
public interface CTTPEmulationModelService {
    void insert(CTTPEmulationModel aModelElem);
```

```
    void update(CTTPEmulationModel aModelElem);

    void remove(int theId);

    List<CTTPEmulationModel> findAll();

    CTTPEmulationModel findById(int theId);

    CTTPEmulationModel findByName(String theName);
}
```

CTTP Emulation Model Service Implementation

```
@Repository
public class CTTPEmulationModelServiceImpl implements CTTPEmulationModelService {
    @Autowired
    private CTTPEmulationModelDAO objectDAO ;

    @Autowired
    public CTTPEmulationModelServiceImpl(CTTPEmulationModelDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(CTTPEmulationModel aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(CTTPEmulationModel aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<CTTPEmulationModel> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public CTTPEmulationModel findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public CTTPEmulationModel findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.2 Networks

Networks DAO Implementation

```
@Repository
public class NetworksDAOImpl implements NetworksDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Networks> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Networks> NetworksList =  session.createQuery("from
Networks").list();

            for(Networks a: NetworksList){
                logger.info("Networks List::" + a);
            }
            return NetworksList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Networks findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Networks a = (Networks) session.load(Networks.class, new
Integer(theId));
            logger.info("Networks loaded successfully, Networks details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Networks findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Networks where name=:name");
            query.setParameter("name", theName);

            Networks a = (Networks) query.uniqueResult();
            logger.info("Networks loaded successfully, Networks details="+ a);
            return a;
```

```
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Networks NetworksElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(NetworksElement);
            logger.info("Networks saved successfully, Networks Details="+
NetworksElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Networks Networks) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Networks);
            logger.info("Networks updated successfully, Networks Details="+
Networks);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Networks a = (Networks) session.load(Networks.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Networks deleted successfully, Networks details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Networks Entity

```
@Entity
public class Networks implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
```

```
    private long networksID;

    @ManyToMany(mappedBy = "networksID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<CTTPEmulationModel> cttpEmulationModelSet;

    public long getNetworksID() {
        return networksID;
    }

    public void setNetworksID(long networksID) {
        this.networksID = networksID;
    }

    public Set<CTTPEmulationModel> getCttpEmulationModelSet() {
        return cttpEmulationModelSet;
    }

    public void setCttpEmulationModelSet(Set<CTTPEmulationModel>
cttpEmulationModelSet) {
        this.cttpEmulationModelSet = cttpEmulationModelSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Networks)) return false;
        Networks networks = (Networks) o;
        return getNetworksID() == networks.getNetworksID() &&
                Objects.equals(getCttpEmulationModelSet(),
networks.getCttpEmulationModelSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getNetworksID(), getCttpEmulationModelSet());
    }
}
```

Networks Service Implementation

```
@Repository
public class NetworksServiceImpl implements NetworksService {
    @Autowired
    private NetworksDAO objectDAO ;

    @Autowired
    public NetworksServiceImpl(NetworksDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Networks aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Networks aModelElem) {
```

```
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Networks> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Networks findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Networks findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.3 Network

Network DAO Implementation

```
@Repository
public class NetworkDAOImpl implements NetworkDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Network> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Network> NetworkList =  session.createQuery("from
Network").list();

            for(Network a: NetworkList){
                logger.info("Network List::" + a);
            }
            return NetworkList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }
```

```
    @Override
    public Network findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Network a = (Network) session.load(Network.class, new Integer(theId));
            logger.info("Network loaded successfully, Network details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Network findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Network where name=:name");
            query.setParameter("name", theName);

            Network a = (Network) query.uniqueResult();
            logger.info("Network loaded successfully, Network details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Network NetworkElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(NetworkElement);
            logger.info("Network saved successfully, Network Details="+
NetworkElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Network Network) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Network);
            logger.info("Network updated successfully, Network Details="+
Network);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }
```

```
    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Network a = (Network) session.load(Network.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Network deleted successfully, Network details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Network *Entity*

```
@Entity
public class Network implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long networkID;
    @Id
    private long networkdsID;
    @Id
    private long cidrID;
    @Id
    private long gatewayID;
    private String id;
    private Boolean isExternal;


    public long getNetworkID() {
        return networkID;
    }

    public void setNetworkID(long networkID) {
        this.networkID = networkID;
    }

    public long getNetworkdsID() {
        return networkdsID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Network)) return false;
        Network network = (Network) o;
        return getNetworkID() == network.getNetworkID() &&
                getNetworkdsID() == network.getNetworkdsID() &&
                getCidrID() == network.getCidrID() &&
                getGatewayID() == network.getGatewayID() &&
                Objects.equals(getId(), network.getId()) &&
```

```
                Objects.equals(isExternal, network.isExternal);
    }

    @Override
    public int hashCode() {
        return Objects.hash(getNetworkID(), getNetworkdsID(), getCidrID(),
getGatewayID(), getId(), isExternal);
    }

    public void setNetworkdsID(long networkdsID) {
        this.networkdsID = networkdsID;
    }

    public long getCidrID() {
        return cidrID;
    }

    public void setCidrID(long cidrID) {
        this.cidrID = cidrID;
    }

    public long getGatewayID() {
        return gatewayID;
    }

    public void setGatewayID(long gatewayID) {
        this.gatewayID = gatewayID;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public Boolean getExternal() {
        return isExternal;
    }

    public void setExternal(Boolean external) {
        isExternal = external;
    }

}
```

Network Service Implementation

```
@Repository
public class NetworkServiceImpl implements NetworkService {
    @Autowired
    private NetworkDAO objectDAO ;

    @Autowired
    public NetworkServiceImpl(NetworkDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
```

```
    @Transactional
    public void insert(Network aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Network aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Network> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Network findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Network findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.4 Scripts

Scripts DAO Implementation

```
@Repository
public class ScriptsDAOImpl implements ScriptsDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Scripts> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Scripts> ScriptsList =  session.createQuery("from
Scripts").list();

            for(Scripts a: ScriptsList){
                logger.info("Scripts List::" + a);
```

```
            }
            return ScriptsList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Scripts findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Scripts a = (Scripts) session.load(Scripts.class, new Integer(theId));
            logger.info("Scripts loaded successfully, Scripts details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Scripts findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Scripts where name=:name");
            query.setParameter("name", theName);

            Scripts a = (Scripts) query.uniqueResult();
            logger.info("Scripts loaded successfully, Scripts details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Scripts ScriptsElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ScriptsElement);
            logger.info("Scripts saved successfully, Scripts Details="+
ScriptsElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Scripts Scripts) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Scripts);
```

```
            logger.info("Scripts updated successfully, Scripts Details="+
Scripts);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Scripts a = (Scripts) session.load(Scripts.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Scripts deleted successfully, Scripts details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Scripts *Entity*

```
@Entity
public class Scripts implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long scriptsID;

    @ManyToMany(mappedBy = "scriptsID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<CTTPEmulationModel> cttpEmulationModelSet;

    public long getScriptsID() {
        return scriptsID;
    }

    public void setScriptsID(long scriptsID) {
        this.scriptsID = scriptsID;
    }

    public Set<CTTPEmulationModel> getCttpEmulationModelSet() {
        return cttpEmulationModelSet;
    }

    public void setCttpEmulationModelSet(Set<CTTPEmulationModel>
cttpEmulationModelSet) {
        this.cttpEmulationModelSet = cttpEmulationModelSet;
    }

    @Override
    public boolean equals(Object o) {
```

```
        if (this == o) return true;
        if (!(o instanceof Scripts)) return false;
        Scripts scripts = (Scripts) o;
        return getScriptsID() == scripts.getScriptsID() &&
                Objects.equals(getCttpEmulationModelSet(),
scripts.getCttpEmulationModelSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getScriptsID(), getCttpEmulationModelSet());
    }
}
```

Scripts Service Implementation

```
@Repository
public class ScriptsServiceImpl implements ScriptsService {
    @Autowired
    private ScriptsDAO objectDAO ;

    @Autowired
    public ScriptsServiceImpl(ScriptsDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Scripts aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Scripts aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Scripts> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Scripts findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Scripts findByName(String theName) {
        return objectDAO.findByName(theName);
```

```
    }
}
```

3.2.6.4.5 Script

Script DAO Implementation

```
@Repository
public class ScriptDAOImpl implements ScriptDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Script> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Script> ScriptList =  session.createQuery("from Script").list();

            for(Script a: ScriptList){
                logger.info("Script List::" + a);
            }
            return ScriptList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Script findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Script a = (Script) session.load(Script.class, new Integer(theId));
            logger.info("Script loaded successfully, Script details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Script findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Script where name=:name");
            query.setParameter("name", theName);

            Script a = (Script) query.uniqueResult();
            logger.info("Script loaded successfully, Script details="+ a);
```

```
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Script ScriptElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ScriptElement);
            logger.info("Script saved successfully, Script Details="+
ScriptElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Script Script) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Script);
            logger.info("Script updated successfully, Script Details="+ Script);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Script a = (Script) session.load(Script.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Script deleted successfully, Script details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Script *Entity*

```
@Entity
public class Script implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long scriptID;
```

```
    @Id
    private long scriptsID;
    private String id;
    private String body;

    public long getScriptID() {
        return scriptID;
    }

    public void setScriptID(long scriptID) {
        this.scriptID = scriptID;
    }

    public long getScriptsID() {
        return scriptsID;
    }

    public void setScriptsID(long scriptsID) {
        this.scriptsID = scriptsID;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Script)) return false;
        Script script = (Script) o;
        return getScriptID() == script.getScriptID() &&
                getScriptsID() == script.getScriptsID() &&
                Objects.equals(getId(), script.getId()) &&
                Objects.equals(getBody(), script.getBody());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getScriptID(), getScriptsID(), getId(), getBody());
    }
}
```

Script Service Implementation

```
@Repository
public class ScriptServiceImpl implements ScriptService {
    @Autowired
    private ScriptDAO objectDAO ;
```

```
    @Autowired
    public ScriptServiceImpl(ScriptDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Script aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Script aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Script> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Script findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Script findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.6 Routers

Routers DAO Implementation

```
@Repository
public class RoutersDAOImpl implements RoutersDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Routers> findAll() {
```

```
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Routers> RoutersList =  session.createQuery("from
Routers").list();

            for(Routers a: RoutersList){
                logger.info("Routers List::" + a);
            }
            return RoutersList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Routers findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Routers a = (Routers) session.load(Routers.class, new Integer(theId));
            logger.info("Routers loaded successfully, Routers details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Routers findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Routers where name=:name");
            query.setParameter("name", theName);

            Routers a = (Routers) query.uniqueResult();
            logger.info("Routers loaded successfully, Routers details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Routers RoutersElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(RoutersElement);
            logger.info("Routers saved successfully, Routers Details="+
RoutersElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
```

```
    }

    @Override
    public void update(Routers Routers) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Routers);
            logger.info("Routers updated successfully, Routers Details="+
Routers);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Routers a = (Routers) session.load(Routers.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Routers deleted successfully, Routers details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Routers *Entity*

```
@Entity
public class Routers implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long routersID;

    @ManyToMany(mappedBy = "routersID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<CTTPEmulationModel> cttpEmulationModelSet;

    public long getRoutersID() {
        return routersID;
    }

    public void setRoutersID(long routersID) {
        this.routersID = routersID;
    }

    public Set<CTTPEmulationModel> getCttpEmulationModelSet() {
        return cttpEmulationModelSet;
    }

    public void setCttpEmulationModelSet(Set<CTTPEmulationModel>
```

```
cttpEmulationModelSet) {
        this.cttpEmulationModelSet = cttpEmulationModelSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Routers)) return false;
        Routers routers = (Routers) o;
        return getRoutersID() == routers.getRoutersID() &&
                Objects.equals(getCttpEmulationModelSet(),
routers.getCttpEmulationModelSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getRoutersID(), getCttpEmulationModelSet());
    }
}
```

Routers Service Implementation

```
@Repository
public class RoutersServiceImpl implements RoutersService {
    @Autowired
    private RoutersDAO objectDAO ;

    @Autowired
    public RoutersServiceImpl(RoutersDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Routers aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Routers aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Routers> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Routers findById(int theId) {
        return objectDAO.findById(theId);
```

```
    }

    @Override
    @Transactional
    public Routers findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.7 Router

Router DAO Implementation

```
@Repository
public class RouterDAOImpl implements RouterDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Router> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Router> RouterList =  session.createQuery("from Router").list();

            for(Router a: RouterList){
                logger.info("Router List::" + a);
            }
            return RouterList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Router findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Router a = (Router) session.load(Router.class, new Integer(theId));
            logger.info("Router loaded successfully, Router details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Router findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
```

```
            Query query= session.
                    createQuery("from Router where name=:name");
            query.setParameter("name", theName);

            Router a = (Router) query.uniqueResult();
            logger.info("Router loaded successfully, Router details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Router RouterElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(RouterElement);
            logger.info("Router saved successfully, Router Details="+
RouterElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Router Router) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Router);
            logger.info("Router updated successfully, Router Details="+ Router);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Router a = (Router) session.load(Router.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Router deleted successfully, Router details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Router Entity

```
@Entity
public class Router implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long routerID;
    @Id
    private long routersID;
    private String id;
    private String img;
    private String flavor;
    private String os;
    private String username;
    private String password;
    private long routerInterfaceID;
    private String ScriptID;

    public long getRouterID() {
        return routerID;
    }

    public void setRouterID(long routerID) {
        this.routerID = routerID;
    }

    public long getRoutersID() {
        return routersID;
    }

    public void setRoutersID(long routersID) {
        this.routersID = routersID;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getImg() {
        return img;
    }

    public void setImg(String img) {
        this.img = img;
    }

    public String getFlavor() {
        return flavor;
    }

    public void setFlavor(String flavor) {
        this.flavor = flavor;
    }

    public String getOs() {
        return os;
```

```
    }

    public void setOs(String os) {
        this.os = os;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public long getRouterInterfaceID() {
        return routerInterfaceID;
    }

    public void setRouterInterfaceID(long routerInterfaceID) {
        this.routerInterfaceID = routerInterfaceID;
    }

    public String getScriptID() {
        return ScriptID;
    }

    public void setScriptID(String scriptID) {
        ScriptID = scriptID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Router)) return false;
        Router router = (Router) o;
        return getRouterID() == router.getRouterID() &&
                getRoutersID() == router.getRoutersID() &&
                getRouterInterfaceID() == router.getRouterInterfaceID() &&
                Objects.equals(getId(), router.getId()) &&
                Objects.equals(getImg(), router.getImg()) &&
                Objects.equals(getFlavor(), router.getFlavor()) &&
                Objects.equals(getOs(), router.getOs()) &&
                Objects.equals(getUsername(), router.getUsername()) &&
                Objects.equals(getPassword(), router.getPassword()) &&
                Objects.equals(getScriptID(), router.getScriptID());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getRouterID(), getRoutersID(), getId(), getImg(),
getFlavor(), getOs(), getUsername(), getPassword(), getRouterInterfaceID(),
getScriptID());
```

```
    }
}
```

Router Service Implementation

```
@Repository
public class RouterServiceImpl implements RouterService {
    @Autowired
    private RouterDAO objectDAO ;

    @Autowired
    public RouterServiceImpl(RouterDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Router aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Router aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Router> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Router findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Router findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.8 CustomVM

CustomVM DAO Implementation

```
@Repository
public class CustomVMDAOImpl implements CustomVMDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
```

```
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<CustomVM> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<CustomVM> CustomVMList =  session.createQuery("from
CustomVM").list();

            for(CustomVM a: CustomVMList){
                logger.info("CustomVM List::" + a);
            }
            return CustomVMList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public CustomVM findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CustomVM a = (CustomVM) session.load(CustomVM.class, new
Integer(theId));
            logger.info("CustomVM loaded successfully, CustomVM details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public CustomVM findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from CustomVM where name=:name");
            query.setParameter("name", theName);

            CustomVM a = (CustomVM) query.uniqueResult();
            logger.info("CustomVM loaded successfully, CustomVM details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(CustomVM CustomVMElement) {
```

```
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(CustomVMElement);
            logger.info("CustomVM saved successfully, CustomVM Details="+
CustomVMElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(CustomVM CustomVM) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(CustomVM);
            logger.info("CustomVM updated successfully, CustomVM Details="+
CustomVM);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CustomVM a = (CustomVM) session.load(CustomVM.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("CustomVM deleted successfully, CustomVM details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

CustomVM Entity

```
@Entity
public class CustomVM implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long customVMID;
    private String name;
    private long connectionModeTypeID;
    private int ram;
    private int disk;
    private long imageTypeID;
    private long networkTypeID;
    private String scriptiD;
```

```
    private int cvpus;

    @ManyToMany(mappedBy = "customVMID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<CTTPEmulationModel> cttpEmulationModelSet;

    public long getCustomVMID() {
        return customVMID;
    }

    public void setCustomVMID(long customVMID) {
        this.customVMID = customVMID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getConnectionModeTypeID() {
        return connectionModeTypeID;
    }

    public void setConnectionModeTypeID(long connectionModeTypeID) {
        this.connectionModeTypeID = connectionModeTypeID;
    }

    public int getRam() {
        return ram;
    }

    public void setRam(int ram) {
        this.ram = ram;
    }

    public int getDisk() {
        return disk;
    }

    public void setDisk(int disk) {
        this.disk = disk;
    }

    public long getImageTypeID() {
        return imageTypeID;
    }

    public void setImageTypeID(long imageTypeID) {
        this.imageTypeID = imageTypeID;
    }

    public long getNetworkTypeID() {
        return networkTypeID;
    }

    public void setNetworkTypeID(long networkTypeID) {
        this.networkTypeID = networkTypeID;
```

```
    }

    public String getScriptiD() {
        return scriptiD;
    }

    public void setScriptiD(String scriptiD) {
        this.scriptiD = scriptiD;
    }

    public int getCvpus() {
        return cvpus;
    }

    public void setCvpus(int cvpus) {
        this.cvpus = cvpus;
    }

    public Set<CTTPEmulationModel> getCttpEmulationModelSet() {
        return cttpEmulationModelSet;
    }

    public void setCttpEmulationModelSet(Set<CTTPEmulationModel>
cttpEmulationModelSet) {
        this.cttpEmulationModelSet = cttpEmulationModelSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof CustomVM)) return false;
        CustomVM customVM = (CustomVM) o;
        return getCustomVMID() == customVM.getCustomVMID() &&
                getConnectionModeTypeID() == customVM.getConnectionModeTypeID() &&
                getRam() == customVM.getRam() &&
                getDisk() == customVM.getDisk() &&
                getImageTypeID() == customVM.getImageTypeID() &&
                getNetworkTypeID() == customVM.getNetworkTypeID() &&
                getCvpus() == customVM.getCvpus() &&
                Objects.equals(getName(), customVM.getName()) &&
                Objects.equals(getScriptiD(), customVM.getScriptiD()) &&
                Objects.equals(getCttpEmulationModelSet(),
customVM.getCttpEmulationModelSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getCustomVMID(), getName(), getConnectionModeTypeID(),
getRam(), getDisk(), getImageTypeID(), getNetworkTypeID(), getScriptiD(),
getCvpus(), getCttpEmulationModelSet());
    }
}
```

CustomVM Service Implementation

```
@Repository
public class CustomVMServiceImpl implements CustomVMService {
    @Autowired
    private CustomVMDAO objectDAO ;
```

```
    @Autowired
    public CustomVMServiceImpl(CustomVMDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(CustomVM aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(CustomVM aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<CustomVM> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public CustomVM findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public CustomVM findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.4.9 Emulation Expected Trace

EmExpectedTrace DAO Implementation

```
@Repository
public class EmExpectedTraceDAOImpl implements EmExpectedTraceDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<EmExpectedTrace> findAll() {
        try{
```

```
            Session session = this.sessionFactory.getCurrentSession();
            List<EmExpectedTrace> EmExpectedTraceList =  session.createQuery("from
EmExpectedTrace").list();

            for(EmExpectedTrace a: EmExpectedTraceList){
                logger.info("EmExpectedTrace List::" + a);
            }
            return EmExpectedTraceList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public EmExpectedTrace findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            EmExpectedTrace a = (EmExpectedTrace)
session.load(EmExpectedTrace.class, new Integer(theId));
            logger.info("EmExpectedTrace loaded successfully, EmExpectedTrace
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public EmExpectedTrace findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from EmExpectedTrace where name=:name");
            query.setParameter("name", theName);

            EmExpectedTrace a = (EmExpectedTrace) query.uniqueResult();
            logger.info("EmExpectedTrace loaded successfully, EmExpectedTrace
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(EmExpectedTrace EmExpectedTraceElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(EmExpectedTraceElement);
            logger.info("EmExpectedTrace saved successfully, EmExpectedTrace
Details="+ EmExpectedTraceElement);
        }
        catch(Exception e){
```

```
            e.printStackTrace();
        }
    }

    @Override
    public void update(EmExpectedTrace EmExpectedTrace) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(EmExpectedTrace);
            logger.info("EmExpectedTrace updated successfully, EmExpectedTrace
Details="+ EmExpectedTrace);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            EmExpectedTrace a = (EmExpectedTrace)
session.load(EmExpectedTrace.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("EmExpectedTrace deleted successfully, EmExpectedTrace
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

EmExpectedTrace Entity

```
@Entity
public class EmExpectedTrace implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long emExpectedTraceID;
    private String value;
    private long emModelID;

    public long getEmExpectedTraceID() {
        return emExpectedTraceID;
    }

    public void setEmExpectedTraceID(long emExpectedTraceID) {
        this.emExpectedTraceID = emExpectedTraceID;
    }

    public String getValue() {
        return value;
    }
```

```
    public void setValue(String value) {
        this.value = value;
    }

    public long getEmModelID() {
        return emModelID;
    }

    public void setEmModelID(long emModelID) {
        this.emModelID = emModelID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof EmExpectedTrace)) return false;
        EmExpectedTrace that = (EmExpectedTrace) o;
        return getEmExpectedTraceID() == that.getEmExpectedTraceID() &&
                getEmModelID() == that.getEmModelID() &&
                getValue().equals(that.getValue());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getEmExpectedTraceID(), getValue(), getEmModelID());
    }

}
```

EmExpectedTrace Service Implementation

```
@Repository
public class EmExpectedTraceServiceImpl implements EmExpectedTraceService {
    @Autowired
    private EmExpectedTraceDAO objectDAO ;

    @Autowired
    public EmExpectedTraceServiceImpl(EmExpectedTraceDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(EmExpectedTrace aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(EmExpectedTrace aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }
```

```
    @Override
    @Transactional
    public List<EmExpectedTrace> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public EmExpectedTrace findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public EmExpectedTrace findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.6.5 Database

Based on the above, a database structure was created to illustrate the emulation database tables, as depicted in Figure 9.

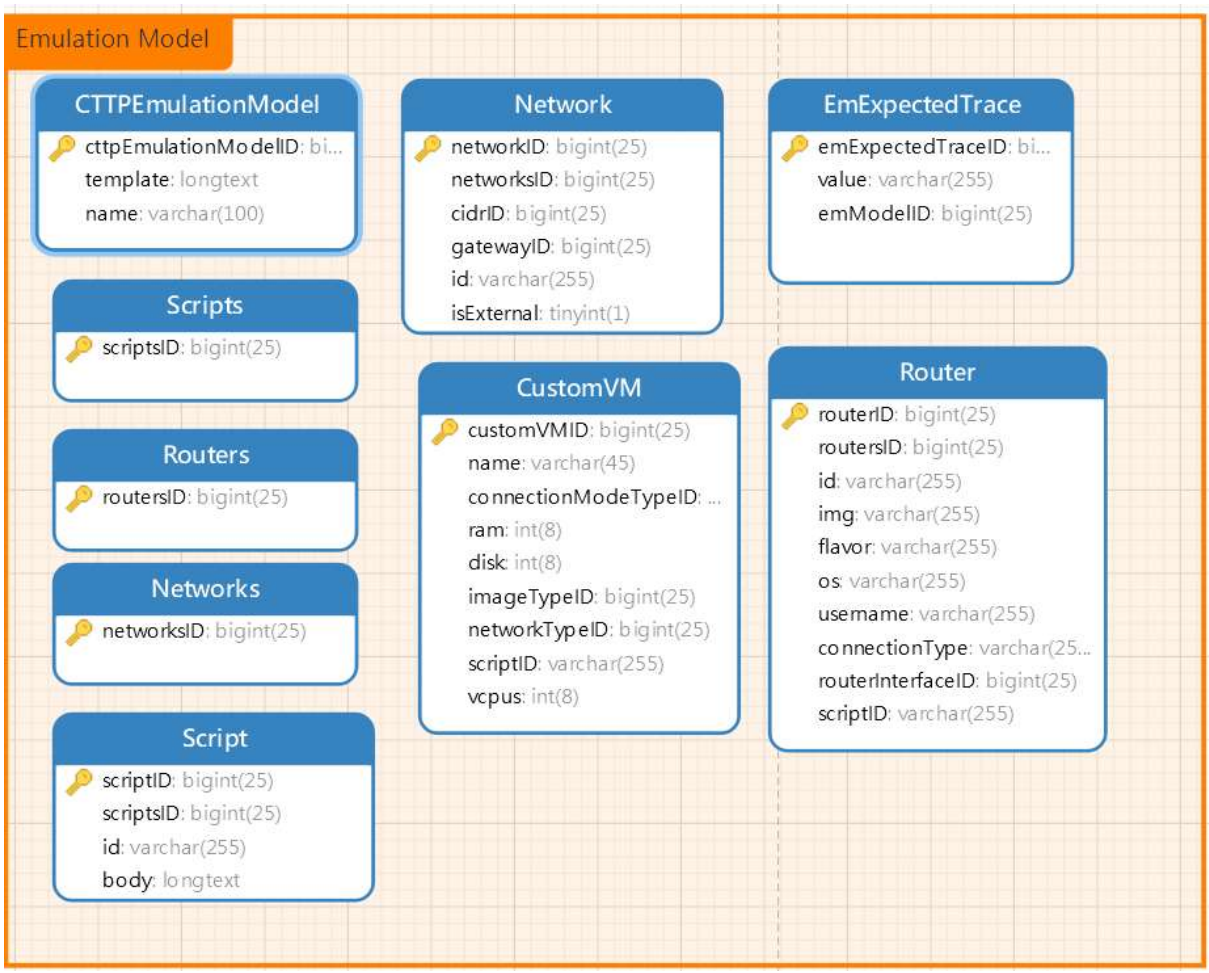

Figure 9: Emulation Model Database Schema

3.2.7 The Gamification parser

The subsections below present the gamification parser, its grammar, and its Java classes that implements the data access objects and the services, and its database schema. The gamification grammar was not described in D3.1 as its need was identified later in the project.

3.2.7.1 Grammar

```
*********CTTP Gamification Model*********

cttpGamificationModel: cttpGamificationModelTitle OPEN_PAREN isPartOfTpRule COMMA
(containsGameRule)+
COMMA (setsGamificationExpectedTraceRule)+ CLOSE_PAREN;
cttpGamificationModelTitle: 'cttpGamificationModel';
cttpGamificationModelID: Identifier;
containsGameRule: 'containsGame' OPEN_PAREN gameID CLOSE_PAREN;
setsGamificationExpectedTraceRule: 'setsGamificationExpectedTrace' OPEN_PAREN
gamificationExpectedTraceID CLOSE_PAREN;

/*Game*/
game: gameTitle (protect (COMMA)*)* (awarenessQuest (COMMA)*)* OPEN_PAREN
difficultyLevel COMMA gameTime ;
gameTitle: 'game';
gameID: Identifier;
difficultyLevel: 'difficultyLevel' OPEN_PAREN INT CLOSE_PAREN;
gameTime: 'gameTime'OPEN_PAREN INT CLOSE_PAREN;

/*PROTECT*/
protect: protectTitle OPEN_PAREN cardDeckInternalID COMMA specialPractice
CLOSE_PAREN;
protectTitle: 'protect';
cardDeckInternalID: 'cardDeckID' OPEN_PAREN STRING CLOSE_PAREN;
specialPractice: 'specialPractice'OPEN_PAREN bool CLOSE_PAREN;

/*AwarenessQuest*/
awarenessQuest: awarenessQuestTitle OPEN_PAREN questionSetID CLOSE_PAREN;
awarenessQuestTitle: 'awarenessQuest';
questionSetID: 'questionSetID'OPEN_PAREN STRING CLOSE_PAREN;

/*GamificationExpectedTrace*/
gamificationExpectedTrace: gamificationExpectedTraceTitle;
gamificationExpectedTraceTitle: 'gamificationExpectedTrace';
gamificationExpectedTraceID: Identifier;
```

3.2.7.2 Java Classes

3.2.7.2.1 CTTP Gamification Model

CTTP Gamification Model DAO

```
public interface CTTPGamificationModelDAO {

    void insert(CTTPGamificationModel aModelElem);

    void update(CTTPGamificationModel aModelElem);

    void remove(int theId);

    List<CTTPGamificationModel> findAll();
```

```
    CTTPGamificationModel findById(int theId);

    CTTPGamificationModel findByName(String theName);
}
```

CTTP Gamification Model DAO Implementation

```
@Repository
public class CTTPGamificationModelDAOImpl implements
ch.sphynx.Implementation.parser.dao.CTTPGamificationModelDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<CTTPGamificationModel> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<CTTPGamificationModel> assetList =  session.createQuery("from
CTTPGamification").list();

            for(CTTPGamificationModel a: assetList){
                logger.info("CTTPGamification List::" + a);
            }
            return assetList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public CTTPGamificationModel findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPGamificationModel a = (CTTPGamificationModel)
session.load(CTTPGamificationModel.class, new Integer(theId));
            logger.info("Asset loaded successfully, Asset details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public CTTPGamificationModel findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Asset where name=:name");
            query.setParameter("name", theName);
```

```
            CTTPGamificationModel a = (CTTPGamificationModel)
query.uniqueResult();
            logger.info("Asset loaded successfully, Asset details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(CTTPGamificationModel CTTPGamificationModelObjectInsert) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(CTTPGamificationModelObjectInsert);
            logger.info("Asset saved successfully, Asset Details="+
CTTPGamificationModelObjectInsert);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(CTTPGamificationModel CTTPGamificationModelObjectUpdate) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(CTTPGamificationModelObjectUpdate);
            logger.info("Asset updated successfully, Asset Details="+
CTTPGamificationModelObjectUpdate);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPGamificationModel a = (CTTPGamificationModel)
session.load(CTTPGamificationModel.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Asset deleted successfully, asset details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

CTTP Gamification Model Entity

```
@Entity
public class CTTPGamificationModel implements java.io.Serializable{
    @Id
    @Column(name="cttpGamificationModelID", nullable = false)
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long cttpGamificationModelID;

    @ManyToMany(mappedBy = "cttpGamificationModelID",cascade =
CascadeType.REFRESH, fetch = FetchType.LAZY)
    private Set<TrainingProgramme> trainingProgrammeSet;

    @Id
    public long getCttpGamificationModelID() {
        return cttpGamificationModelID;
    }
    public void setCttpGamificationModelID(long cttpGamificationModelID) {
        this.cttpGamificationModelID = cttpGamificationModelID;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof CTTPGamificationModel)) return false;
        CTTPGamificationModel that = (CTTPGamificationModel) o;
        return getCttpGamificationModelID() == that.getCttpGamificationModelID();
    }
    @Override
    public int hashCode() {
        return Objects.hash(getCttpGamificationModelID());
    }
}
```

CTTP Gamification Model Service

```
public interface CTTPGamificationModelService {
    void insert(CTTPGamificationModel aModelElem);

    void update(CTTPGamificationModel aModelElem);

    void remove(int theId);

    List<CTTPGamificationModel> findAll();

    CTTPGamificationModel findById(int theId);

    CTTPGamificationModel findByName(String theName);
}
```

CTTP Gamification Model Service Implementation

```
@Repository
public class CTTPGamificationModelServiceImpl implements
CTTPGamificationModelService {
    @Autowired
    private CTTPGamificationModelDAO objectDAO ;

    @Autowired
    public CTTPGamificationModelServiceImpl(CTTPGamificationModelDAO
constructorDAO) {
        this.objectDAO = constructorDAO;
    }
```

```
    @Override
    @Transactional
    public void insert(CTTPGamificationModel aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(CTTPGamificationModel aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<CTTPGamificationModel> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public CTTPGamificationModel findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public CTTPGamificationModel findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.7.2.2 Game

Game DAO Implementation

```
@Repository
public class GameDAOImpl implements GameDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Game> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Game> GameList =  session.createQuery("from Game").list();

            for(Game a: GameList){
```

```
                logger.info("Game List::" + a);
            }
            return GameList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Game findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Game a = (Game) session.load(Game.class, new Integer(theId));
            logger.info("Game loaded successfully, Game details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Game findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Game where name=:name");
            query.setParameter("name", theName);

            Game a = (Game) query.uniqueResult();
            logger.info("Game loaded successfully, Game details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Game GameElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(GameElement);
            logger.info("Game saved successfully, Game Details="+ GameElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Game Game) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Game);
```

```
            logger.info("Game updated successfully, Game Details="+ Game);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Game a = (Game) session.load(Game.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Game deleted successfully, Game details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Game Entity

```
@Entity
public class Game implements java.io.Serializable{
    @Id
    @Column(name="gameID", nullable = false)
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long gameID;
    private long gameTypeID;
    private int difficultyLevel;

    @Id
    public long getGameID() {
        return gameID;
    }
    public void setGameID(long gameID) {
        this.gameID = gameID;
    }

    @Id
    public long getGameTypeID() {
        return gameTypeID;
    }
    public void setGameTypeID(long gameTypeID) {
        this.gameTypeID = gameTypeID;
    }

    @Basic
    public int getDifficultyLevel() {
        return difficultyLevel;
    }
    public void setDifficultyLevel(int difficultyLevel) {
        this.difficultyLevel = difficultyLevel;
```

```
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Game)) return false;
        Game game = (Game) o;
        return getGameID() == game.getGameID() &&
                getGameTypeID() == game.getGameTypeID() &&
                getDifficultyLevel() == game.getDifficultyLevel();
    }

    @Override
    public int hashCode() {
        return Objects.hash(getGameID(), getGameTypeID(), getDifficultyLevel());
    }
}
```

Game Service Implementation

```
@Repository
public class GameServiceImpl implements GameService {
    @Autowired
    private GameDAO objectDAO ;

    @Autowired
    public GameServiceImpl(GameDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Game aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Game aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Game> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Game findById(int theId) {
        return objectDAO.findById(theId);
    }
```

```
    @Override
    @Transactional
    public Game findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.7.2.3 The PROTECT game

Protect DAO Implementation

```
@Repository
public class ProtectDAOImpl implements ProtectDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Protect> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Protect> ProtectList =  session.createQuery("from
Protect").list();

            for(Protect a: ProtectList){
                logger.info("Protect List::" + a);
            }
            return ProtectList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Protect findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Protect a = (Protect) session.load(Protect.class, new Integer(theId));
            logger.info("Protect loaded successfully, Protect details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Protect findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
```

```
                    createQuery("from Protect where name=:name");
            query.setParameter("name", theName);

            Protect a = (Protect) query.uniqueResult();
            logger.info("Protect loaded successfully, Protect details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Protect ProtectElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ProtectElement);
            logger.info("Protect saved successfully, Protect Details="+
ProtectElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Protect Protect) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Protect);
            logger.info("Protect updated successfully, Protect Details="+
Protect);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Protect a = (Protect) session.load(Protect.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Protect deleted successfully, Protect details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Protect Entity

```
@Entity
public class Protect implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long protectID;
    private String cardDeckID;
    private int specialPractice;

    @Id
    public long getProtectID() {
        return protectID;
    }

    public void setProtectID(long protectID) {
        this.protectID = protectID;
    }

    public String getCardDeckID() {
        return cardDeckID;
    }

    public void setCardDeckID(String cardDeckID) {
        this.cardDeckID = cardDeckID;
    }

    public int getSpecialPractice() {
        return specialPractice;
    }

    public void setSpecialPractice(int specialPractice) {
        this.specialPractice = specialPractice;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Protect)) return false;
        Protect protect = (Protect) o;
        return getProtectID() == protect.getProtectID() &&
                getSpecialPractice() == protect.getSpecialPractice() &&
                Objects.equals(getCardDeckID(), protect.getCardDeckID());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getProtectID(), getCardDeckID(),
getSpecialPractice());
    }
}
```

Protect Service Implementation

```
@Repository
public class ProtectServiceImpl implements ProtectService {
    @Autowired
    private ProtectDAO objectDAO ;

    @Autowired
```

```
    public ProtectServiceImpl(ProtectDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Protect aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Protect aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Protect> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Protect findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Protect findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.7.2.4 The Awareness Quest game

Awareness Quest DAO Implementation

```
@Repository
public class AwarenessQuestDAOImpl implements AwarenessQuestDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<AwarenessQuest> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
```

```
            List<AwarenessQuest> AwarenessQuestList =  session.createQuery("from
AwarenessQuest").list();

            for(AwarenessQuest a: AwarenessQuestList){
                logger.info("AwarenessQuest List::" + a);
            }
            return AwarenessQuestList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public AwarenessQuest findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            AwarenessQuest a = (AwarenessQuest) session.load(AwarenessQuest.class,
new Integer(theId));
            logger.info("AwarenessQuest loaded successfully, AwarenessQuest
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public AwarenessQuest findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from AwarenessQuest where name=:name");
            query.setParameter("name", theName);

            AwarenessQuest a = (AwarenessQuest) query.uniqueResult();
            logger.info("AwarenessQuest loaded successfully, AwarenessQuest
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(AwarenessQuest AwarenessQuestElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(AwarenessQuestElement);
            logger.info("AwarenessQuest saved successfully, AwarenessQuest
Details="+ AwarenessQuestElement);
        }
        catch(Exception e){
            e.printStackTrace();
```

```
        }
    }

    @Override
    public void update(AwarenessQuest AwarenessQuest) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(AwarenessQuest);
            logger.info("AwarenessQuest updated successfully, AwarenessQuest
Details="+ AwarenessQuest);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            AwarenessQuest a = (AwarenessQuest) session.load(AwarenessQuest.class,
new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("AwarenessQuest deleted successfully, AwarenessQuest
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Awareness Quest Entity

```
@Entity
public class AwarenessQuest implements java.io.Serializable{
    @Id
    @Column(name="awarenessQuestiID", nullable = false)
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long awarenessQuestID;
    private String questionSetID;

    @Id
    public long getAwarenessQuestID() {
        return awarenessQuestID;
    }

    public void setAwarenessQuestID(long awarenessQuestID) {
        this.awarenessQuestID = awarenessQuestID;
    }
    @Basic

    public String getQuestionSetID() {
        return questionSetID;
```

```
    }

    public void setQuestionSetID(String questionSetID) {
        this.questionSetID = questionSetID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof AwarenessQuest)) return false;
        AwarenessQuest that = (AwarenessQuest) o;
        return getAwarenessQuestID() == that.getAwarenessQuestID() &&
                Objects.equals(getQuestionSetID(), that.getQuestionSetID());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getAwarenessQuestID(), getQuestionSetID());
    }
}
```

Awareness Quest Service Implementation

```
@Repository
public class AwarenessQuestServiceImpl implements AwarenessQuestService {
    @Autowired
    private AwarenessQuestDAO objectDAO ;

    @Autowired
    public AwarenessQuestServiceImpl(AwarenessQuestDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(AwarenessQuest aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(AwarenessQuest aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<AwarenessQuest> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public AwarenessQuest findById(int theId) {
```

```
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public AwarenessQuest findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.7.2.5 Gamification Expected Trace

Gamification Expected Trace DAO Implementation

```
@Repository
public class GamificationExpectedTraceDAOImpl implements
GamificationExpectedTraceDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<GamificationExpectedTrace> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<GamificationExpectedTrace> GamificationExpectedTraceList =
session.createQuery("from GamificationExpectedTrace").list();

            for(GamificationExpectedTrace a: GamificationExpectedTraceList){
                logger.info("GamificationExpectedTrace List::" + a);
            }
            return GamificationExpectedTraceList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public GamificationExpectedTrace findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            GamificationExpectedTrace a = (GamificationExpectedTrace)
session.load(GamificationExpectedTrace.class, new Integer(theId));
            logger.info("GamificationExpectedTrace loaded successfully,
GamificationExpectedTrace details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }
```

```
    @Override
    public GamificationExpectedTrace findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from GamificationExpectedTrace where
name=:name");
            query.setParameter("name", theName);

            GamificationExpectedTrace a = (GamificationExpectedTrace)
query.uniqueResult();
            logger.info("GamificationExpectedTrace loaded successfully,
GamificationExpectedTrace details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(GamificationExpectedTrace GamificationExpectedTraceElement)
{
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(GamificationExpectedTraceElement);
            logger.info("GamificationExpectedTrace saved successfully,
GamificationExpectedTrace Details="+ GamificationExpectedTraceElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(GamificationExpectedTrace GamificationExpectedTrace) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(GamificationExpectedTrace);
            logger.info("GamificationExpectedTrace updated successfully,
GamificationExpectedTrace Details="+ GamificationExpectedTrace);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            GamificationExpectedTrace a = (GamificationExpectedTrace)
session.load(GamificationExpectedTrace.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("GamificationExpectedTrace deleted successfully,
```

```
GamificationExpectedTrace details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Gamification Expected Trace Entity

```
@Entity
public class GamificationExpectedTrace implements java.io.Serializable{
    @Id
    @Column(name = "gamificationExpectedTraceID", nullable = false)
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long gamificationExpectedTraceID;
    private String value;
    @Id
    private long gamificationModelID;

    public long getGamificationExpectedTraceID() {
        return gamificationExpectedTraceID;
    }

    @Id
    public void setGamificationExpectedTraceID(long gamificationExpectedTraceID) {
        this.gamificationExpectedTraceID = gamificationExpectedTraceID;
    }

    public String getValue() {
        return value;
    }

    @Basic
    public void setValue(String value) {
        this.value = value;
    }

    @Id
    public long getGamificationModelID() {
        return gamificationModelID;
    }

    public void setGamificationModelID(long gamificationModelID) {
        this.gamificationModelID = gamificationModelID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof GamificationExpectedTrace)) return false;
        GamificationExpectedTrace that = (GamificationExpectedTrace) o;
        return getGamificationExpectedTraceID() ==
that.getGamificationExpectedTraceID() &&
                getGamificationModelID() == that.getGamificationModelID() &&
                Objects.equals(getValue(), that.getValue());
    }
```

```
    @Override
    public int hashCode() {
        return Objects.hash(getGamificationExpectedTraceID(), getValue(),
getGamificationModelID());
    }
}
```

Gamification Expected Trace Service Implementation

```
@Repository
public class GamificationExpectedTraceServiceImpl implements
GamificationExpectedTraceService {
    @Autowired
    private GamificationExpectedTraceDAO objectDAO ;

    @Autowired
    public GamificationExpectedTraceServiceImpl(GamificationExpectedTraceDAO
constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(GamificationExpectedTrace aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(GamificationExpectedTrace aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<GamificationExpectedTrace> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public GamificationExpectedTrace findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public GamificationExpectedTrace findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.7.3 Database

Based on the above, a database structure was created to illustrate the gamification database tables, as depicted in Figure 10.

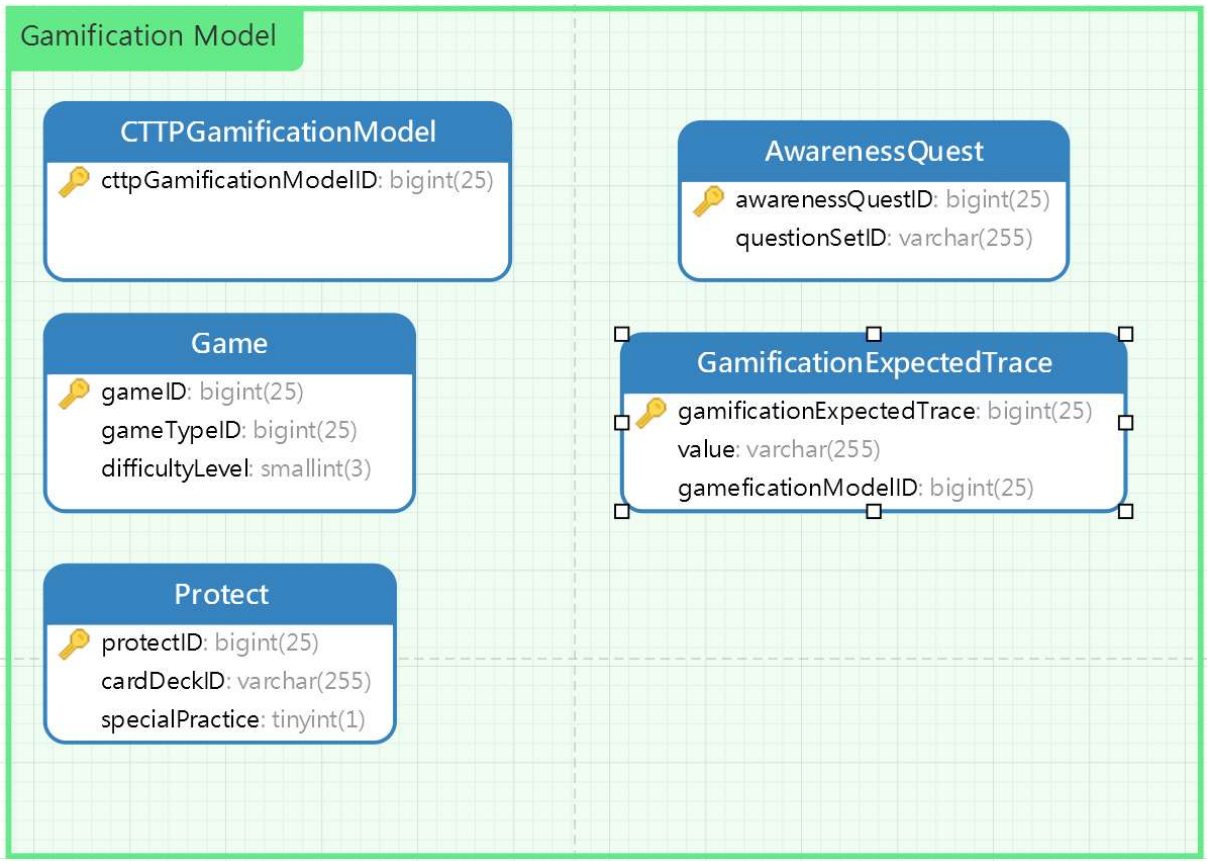

Figure 10: Gamification Model database schema

3.2.8 The Data Fabrication parser

The subsections below present the data fabrication parser, its grammar, and its java classes that implements the data access objects and the services, and its database schema. The data fabrication grammar was not described in D3.1 as its need was identified later in the project.

3.2.8.1 Grammar

```
cttpDataFabricationModel: cttpDataFabricationModelTitle OPEN_PAREN dfname COMMA
hasNetworkRule COMMA hasScenarioRule CLOSE_PAREN;
cttpDataFabricationModelTitle: 'cttpDataFabricationModel';
cttpDataFabricationModelID: Identifier;
dfname: 'dfName' OPEN_PAREN STRING CLOSE_PAREN;
hasNetworkRule: 'hasNetwork' OPEN_PAREN dfNetworkID CLOSE_PAREN;
hasScenarioRule: 'hasScenario' OPEN_PAREN dfScenarioID CLOSE_PAREN;

/* DF Network */
dfNetwork: dfNetworkTitle OPEN_PAREN hasSubnetArrayRule CLOSE_PAREN;
dfNetworkTitle: 'dfNetwork';
dfNetworkID: Identifier;
hasSubnetArrayRule: 'hasSubnetArray' OPEN_PAREN subnetArrayID CLOSE_PAREN;

/* subnetArray */
subnetArray: subnetArrayTitle OPEN_PAREN subnetArrayName COMMA subnetArrayType
COMMA (subnetObjectRule)+ COMMA (subnetArray)+
COMMA (subnetObjectRule)+ CLOSE_PAREN;
```

```
subnetArrayTitle: 'subnetArrayTitle' ;
subnetArrayID: Identifier;
subnetArrayName: 'subnetArrayName' OPEN_PAREN STRING CLOSE_PAREN;
subnetArrayType: 'subnetArrayType' OPEN_PAREN STRING CLOSE_PAREN;
subnetArrayRule: 'subnetArray' OPEN_PAREN subnetArrayID CLOSE_PAREN;
subnetObjectRule: 'subnetObject' OPEN_PAREN subnetID CLOSE_PAREN;

/* subnet */
subnet:subnetTitle OPEN_PAREN subnetName COMMA subnetType COMMA
(subnetConnection)+ COMMA stereotype COMMA hasAppArrayRule
 CLOSE_PAREN;
subnetTitle: 'subnet';
subnetID: Identifier;
subnetName: 'subnetName' OPEN_PAREN STRING CLOSE_PAREN;
subnetType: 'subnetType' OPEN_PAREN STRING CLOSE_PAREN;
subnetConnection: 'subnetConnection' OPEN_PAREN STRING CLOSE_PAREN;
stereotype: 'stereotype' OPEN_PAREN STRING CLOSE_PAREN;
hasAppArrayRule: 'hasAppArray' OPEN_PAREN appsArrayID CLOSE_PAREN;

/* appsArray */
appsArray: appsArrayTitle OPEN_PAREN (hasAppsRule)+ CLOSE_PAREN;
appsArrayTitle: 'appsArray';
appsArrayID: Identifier;
hasAppsRule: 'hasApps' OPEN_PAREN app;

/* app */
app: appTitle OPEN_PAREN appName CLOSE_PAREN;
appTitle: 'app';
appID: Identifier;
appName: STRING;

/* Scenario */
dfScenario: dfScenarioTitle OPEN_PAREN (scenariosRule)+ COMMA isPartOfDFModelRule
CLOSE_PAREN;
dfScenarioTitle: 'dfScenario';
dfScenarioID: Identifier;
scenariosRule: 'scenarios' OPEN_PAREN dfScenarioObjectID CLOSE_PAREN;
isPartOfDFModelRule: 'isPartOfDFModel' OPEN_PAREN cttpDataFabricationModelID
CLOSE_PAREN;

/* Scenario object */
dfScenarioObject: dfScenarioObjectTitle OPEN_PAREN isPartOfScenarioRule  COMMA
scenarioObjectHasConstraintRule CLOSE_PAREN;
dfScenarioObjectTitle: 'dfScenarioObject';
dfScenarioObjectID: Identifier;
isPartOfScenarioRule: 'isPartOfScenario' OPEN_PAREN dfScenarioID CLOSE_PAREN;
scenarioObjectHasConstraintRule: 'hasConstraint' OPEN_PAREN dfConstraintID
CLOSE_PAREN;

/* constraint  */
dfConstraint: dfConstraintTitle OPEN_PAREN dfConstraintName COMMA
dfConstraintDescription COMMA dfConstraintBody COMMA
 CLOSE_PAREN;
dfConstraintTitle: 'dfConstraint';
dfConstraintID: Identifier;
dfConstraintName: 'dfConstraintName' OPEN_PAREN STRING CLOSE_PAREN;
dfConstraintDescription: 'dfConstraintName' OPEN_PAREN STRING CLOSE_PAREN;
dfConstraintBody: 'dfConstraintName' OPEN_PAREN STRING CLOSE_PAREN;

/* Data fabrication expectedTrace */
```

```
dfExpectedTrace: dfExpectedTraceTitle OPEN_PAREN CLOSE_PAREN;
dfExpectedTraceTitle: 'dfExpectedTrace';
dfExpectedTraceID: Identifier;
```

3.2.8.2 Java Classes

3.2.8.2.1 CTTP Data Fabrication Model

CTTP Data Fabrication Model DAO

```
public interface CTTPDataFabricationModelDAO {

    public void insert(CTTPDataFabricationModel aModelElem);

    public void update(CTTPDataFabricationModel aModelElem);

    public void remove(int theId);

    public List<CTTPDataFabricationModel> findAll();

    public CTTPDataFabricationModel findById(int theId);

    public CTTPDataFabricationModel findByName(String theName);
}
```

CTTP Data Fabrication Model DAO Implementation

```
@Repository
public class CTTPDataFabricationModelDAOImpl implements
CTTPDataFabricationModelDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<CTTPDataFabricationModel> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<CTTPDataFabricationModel> CTTPDataFabricationModelList =
session.createQuery("from CTTPDataFabricationModel").list();

            for(CTTPDataFabricationModel a: CTTPDataFabricationModelList){
                logger.info("CTTPDataFabricationModel List::" + a);
            }
            return CTTPDataFabricationModelList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public CTTPDataFabricationModel findById(int theId) {
        try{
```

```
            Session session = this.sessionFactory.getCurrentSession();
            CTTPDataFabricationModel a = (CTTPDataFabricationModel)
session.load(CTTPDataFabricationModel.class, new Integer(theId));
            logger.info("CTTPDataFabricationModel loaded successfully,
CTTPDataFabricationModel details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public CTTPDataFabricationModel findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from CTTPDataFabricationModel where name=:name");
            query.setParameter("name", theName);

            CTTPDataFabricationModel a = (CTTPDataFabricationModel)
query.uniqueResult();
            logger.info("CTTPDataFabricationModel loaded successfully,
CTTPDataFabricationModel details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(CTTPDataFabricationModel
CTTPDataFabricationModelObjectInsert) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(CTTPDataFabricationModelObjectInsert);
            logger.info("CTTPDataFabricationModel saved successfully,
CTTPDataFabricationModel Details="+ CTTPDataFabricationModelObjectInsert);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(CTTPDataFabricationModel
CTTPDataFabricationModelObjectUpdate) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(CTTPDataFabricationModelObjectUpdate);
            logger.info("CTTPDataFabricationModel updated successfully,
CTTPDataFabricationModel Details="+ CTTPDataFabricationModelObjectUpdate);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
```

```
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            CTTPDataFabricationModel a = (CTTPDataFabricationModel)
session.load(CTTPDataFabricationModel.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("CTTPDataFabricationModel deleted successfully, asset
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

CTTP Data Fabrication Model Entity

```
@Entity
public class CTTPDataFabricationModel implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long cttpDataFabricationModelID;
    @Id
    private long scenarioID;
    @Id
    private long dfNetworkID;
    private String name;

    public long getCttpDataFabricationModelID() {
        return cttpDataFabricationModelID;
    }

    public void setCttpDataFabricationModelID(long cttpDataFabricationModelID) {
        this.cttpDataFabricationModelID = cttpDataFabricationModelID;
    }
    public long getScenarioID() {
        return scenarioID;
    }
    public void setScenarioID(long scenarioID) {
        this.scenarioID = scenarioID;
    }
    public long getDfNetworkID() {
        return dfNetworkID;
    }
    public void setDfNetworkID(long dfNetworkID) {
        this.dfNetworkID = dfNetworkID;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```
        this.name = name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof CTTPDataFabricationModel)) return false;
        CTTPDataFabricationModel that = (CTTPDataFabricationModel) o;
        return getCttpDataFabricationModelID() ==
that.getCttpDataFabricationModelID() &&
                getScenarioID() == that.getScenarioID() &&
                getDfNetworkID() == that.getDfNetworkID() &&
                Objects.equals(getName(), that.getName());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getCttpDataFabricationModelID(), getScenarioID(),
getDfNetworkID(), getName());
    }
```

CTTP Data Fabrication Model Service

```
public interface CTTPDataFabricationModelService {
    void insert(CTTPDataFabricationModel aModelElem);

    void update(CTTPDataFabricationModel aModelElem);

    void remove(int theId);

    List<CTTPDataFabricationModel> findAll();

    CTTPDataFabricationModel findById(int theId);

    CTTPDataFabricationModel findByName(String theName);
}
```

CTTP Data Fabrication Model Service Implementation

```
@Repository
public class CTTPDataFabricationModelServiceImpl implements
CTTPDataFabricationModelService {
    @Autowired
    private CTTPDataFabricationModelDAO objectDAO ;

    @Autowired
    public CTTPDataFabricationModelServiceImpl(CTTPDataFabricationModelDAO
constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(CTTPDataFabricationModel aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(CTTPDataFabricationModel aModelElem) {
        objectDAO.update(aModelElem);
```

```
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<CTTPDataFabricationModel> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public CTTPDataFabricationModel findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public CTTPDataFabricationModel findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.2 Scenario

Scenario DAO Implementation

```
@Repository
public class ScenarioDAOImpl implements ScenarioDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Scenario> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Scenario> ScenarioList =  session.createQuery("from
Scenario").list();

            for(Scenario a: ScenarioList){
                logger.info("Scenario List::" + a);
            }
            return ScenarioList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }
```

```
    @Override
    public Scenario findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Scenario a = (Scenario) session.load(Scenario.class, new
Integer(theId));
            logger.info("Scenario loaded successfully, Scenario details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Scenario findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Scenario where name=:name");
            query.setParameter("name", theName);

            Scenario a = (Scenario) query.uniqueResult();
            logger.info("Scenario loaded successfully, Scenario details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Scenario ScenarioElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ScenarioElement);
            logger.info("Scenario saved successfully, Scenario Details="+
ScenarioElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Scenario Scenario) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Scenario);
            logger.info("Scenario updated successfully, Scenario Details="+
Scenario);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }
```

```
    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Scenario a = (Scenario) session.load(Scenario.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Scenario deleted successfully, Scenario details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Scenario Entity

```
@Entity
public class Scenario implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long scenarioID;
    @Id
    private long cttpDFModelID;

    @ManyToMany(mappedBy = "scenarioID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<Constraint> constraintSet;

    public long getScenarioID() {
        return scenarioID;
    }

    public void setScenarioID(long scenarioID) {
        this.scenarioID = scenarioID;
    }

    public long getCttpDFModelID() {
        return cttpDFModelID;
    }

    public void setCttpDFModelID(long cttpDFModelID) {
        this.cttpDFModelID = cttpDFModelID;
    }

    public Set<Constraint> getConstraintSet() {
        return constraintSet;
    }

    public void setConstraintSet(Set<Constraint> constraintSet) {
        this.constraintSet = constraintSet;
    }

    @Override
```

```
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Scenario)) return false;
        Scenario scenario = (Scenario) o;
        return getScenarioID() == scenario.getScenarioID() &&
                getCttpDFModelID() == scenario.getCttpDFModelID() &&
                Objects.equals(getConstraintSet(), scenario.getConstraintSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getScenarioID(), getCttpDFModelID(),
getConstraintSet());
    }
}
```

Scenario Service Implementation

```
@Repository
public class ScenarioServiceImpl implements ScenarioService {
    @Autowired
    private ScenarioDAO objectDAO ;

    @Autowired
    public ScenarioServiceImpl(ScenarioDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Scenario aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Scenario aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Scenario> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Scenario findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
```

```
    public Scenario findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.3 Scenario Object

Scenario DAO Implementation

```
@Repository
public class ScenarioObjectDAOImpl implements ScenarioObjectDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<ScenarioObject> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<ScenarioObject> ScenarioObjectList =  session.createQuery("from
ScenarioObject").list();

            for(ScenarioObject a: ScenarioObjectList){
                logger.info("ScenarioObject List::" + a);
            }
            return ScenarioObjectList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public ScenarioObject findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            ScenarioObject a = (ScenarioObject) session.load(ScenarioObject.class,
new Integer(theId));
            logger.info("ScenarioObject loaded successfully, ScenarioObject
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public ScenarioObject findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
```

```
                    createQuery("from ScenarioObject where name=:name");
            query.setParameter("name", theName);

            ScenarioObject a = (ScenarioObject) query.uniqueResult();
            logger.info("ScenarioObject loaded successfully, ScenarioObject
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(ScenarioObject ScenarioObjectElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ScenarioObjectElement);
            logger.info("ScenarioObject saved successfully, ScenarioObject
Details="+ ScenarioObjectElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(ScenarioObject ScenarioObject) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(ScenarioObject);
            logger.info("ScenarioObject updated successfully, ScenarioObject
Details="+ ScenarioObject);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            ScenarioObject a = (ScenarioObject) session.load(ScenarioObject.class,
new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("ScenarioObject deleted successfully, ScenarioObject
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }
```

```
    }
}
```

Scenario Entity

```
@Entity
public class ScenarioObject implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long scenarioObjectID;
    private long scenarioID;
    private String node;
    private String function;
    private String name;
    private String type;

    @ManyToMany(mappedBy = "scenarioObjectID",cascade = CascadeType.REFRESH, fetch
= FetchType.LAZY)
    private Set<Constraint> constraintSet;

    public long getScenarioObjectID() {
        return scenarioObjectID;
    }

    public void setScenarioObjectID(long scenarioObjectID) {
        this.scenarioObjectID = scenarioObjectID;
    }

    public long getScenarioID() {
        return scenarioID;
    }

    public void setScenarioID(long scenarioID) {
        this.scenarioID = scenarioID;
    }

    public String getNode() {
        return node;
    }

    public void setNode(String node) {
        this.node = node;
    }

    public String getFunction() {
        return function;
    }

    public void setFunction(String function) {
        this.function = function;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
```

```
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Set<Constraint> getConstraintSet() {
        return constraintSet;
    }

    public void setConstraintSet(Set<Constraint> constraintSet) {
        this.constraintSet = constraintSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ScenarioObject)) return false;
        ScenarioObject that = (ScenarioObject) o;
        return getScenarioObjectID() == that.getScenarioObjectID() &&
                getScenarioID() == that.getScenarioID() &&
                Objects.equals(getNode(), that.getNode()) &&
                Objects.equals(getFunction(), that.getFunction()) &&
                Objects.equals(getName(), that.getName()) &&
                Objects.equals(getType(), that.getType()) &&
                Objects.equals(getConstraintSet(), that.getConstraintSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getScenarioObjectID(), getScenarioID(), getNode(),
getFunction(), getName(), getType(), getConstraintSet());
    }
}
```

Scenario Service Implementation

```
@Repository
public class ScenarioObjectServiceImpl implements ScenarioObjectService {
    @Autowired
    private ScenarioObjectDAO objectDAO ;

    @Autowired
    public ScenarioObjectServiceImpl(ScenarioObjectDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(ScenarioObject aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(ScenarioObject aModelElem) {
```

```
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<ScenarioObject> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public ScenarioObject findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public ScenarioObject findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.4 Network

Network DAO Implementation

```
@Repository
public class DFNetworkDAOImpl implements DFNetworkDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<DFNetwork> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<DFNetwork> DFNetworkList =  session.createQuery("from
DFNetwork").list();

            for(DFNetwork a: DFNetworkList){
                logger.info("DFNetwork List::" + a);
            }
            return DFNetworkList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }
```

```
    @Override
    public DFNetwork findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            DFNetwork a = (DFNetwork) session.load(DFNetwork.class, new
Integer(theId));
            logger.info("DFNetwork loaded successfully, DFNetwork details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public DFNetwork findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from DFNetwork where name=:name");
            query.setParameter("name", theName);

            DFNetwork a = (DFNetwork) query.uniqueResult();
            logger.info("DFNetwork loaded successfully, DFNetwork details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(DFNetwork DFNetworkElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(DFNetworkElement);
            logger.info("DFNetwork saved successfully, DFNetwork Details="+
DFNetworkElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(DFNetwork DFNetwork) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(DFNetwork);
            logger.info("DFNetwork updated successfully, DFNetwork Details="+
DFNetwork);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
```

```
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            DFNetwork a = (DFNetwork) session.load(DFNetwork.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("DFNetwork deleted successfully, DFNetwork details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Network Entity

```
@Entity
public class DFNetwork implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long dfNetworkID;
    @Id
    private long cttpDFModelID;
    @Id
    private long subnetArrayID;

    public long getDfNetworkID() {
        return dfNetworkID;
    }

    public void setDfNetworkID(long dfNetworkID) {
        this.dfNetworkID = dfNetworkID;
    }

    public long getCttpDFModelID() {
        return cttpDFModelID;
    }

    public void setCttpDFModelID(long cttpDFModelID) {
        this.cttpDFModelID = cttpDFModelID;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof DFNetwork)) return false;
        DFNetwork dfNetwork = (DFNetwork) o;
        return getDfNetworkID() == dfNetwork.getDfNetworkID() &&
                getCttpDFModelID() == dfNetwork.getCttpDFModelID() &&
                getSubnetArrayID() == dfNetwork.getSubnetArrayID();
    }
```

```
    @Override
    public int hashCode() {
        return Objects.hash(getDfNetworkID(), getCttpDFModelID(),
getSubnetArrayID());
    }

    public long getSubnetArrayID() {
        return subnetArrayID;
    }

    public void setSubnetArrayID(long subnetArrayID) {
        this.subnetArrayID = subnetArrayID;
    }
}
```

Network Service Implementation

```
@Repository
public class DFNetworkServiceImpl implements DFNetworkService {
    @Autowired
    private DFNetworkDAO objectDAO ;

    @Autowired
    public DFNetworkServiceImpl(DFNetworkDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(DFNetwork aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(DFNetwork aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<DFNetwork> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public DFNetwork findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public DFNetwork findByName(String theName) {
```

```
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.5 Subnet

Subnet DAO Implementation

```
@Repository
public class SubnetDAOImpl implements SubnetDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Subnet> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Subnet> SubnetList =  session.createQuery("from Subnet").list();

            for(Subnet a: SubnetList){
                logger.info("Subnet List::" + a);
            }
            return SubnetList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Subnet findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Subnet a = (Subnet) session.load(Subnet.class, new Integer(theId));
            logger.info("Subnet loaded successfully, Subnet details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Subnet findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Subnet where name=:name");
            query.setParameter("name", theName);

            Subnet a = (Subnet) query.uniqueResult();
```

```
            logger.info("Subnet loaded successfully, Subnet details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Subnet SubnetElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(SubnetElement);
            logger.info("Subnet saved successfully, Subnet Details="+
SubnetElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Subnet Subnet) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(Subnet);
            logger.info("Subnet updated successfully, Subnet Details="+ Subnet);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Subnet a = (Subnet) session.load(Subnet.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Subnet deleted successfully, Subnet details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Subnet Entity

```
@Entity
public class Subnet implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
```

```
    private long subnetID;
    @Id
    private long subnetArrayID;
    @Id
    private long appArrayID;
    @Id
    private long dfConnectionID;
    private String name;
    private String type;
    private String sterotype;

    @ManyToMany(mappedBy = "subnetID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<Constraint> constraintSet;

    public long getSubnetID() {
        return subnetID;
    }

    public void setSubnetID(long subnetID) {
        this.subnetID = subnetID;
    }

    public long getSubnetArrayID() {
        return subnetArrayID;
    }

    public void setSubnetArrayID(long subnetArrayID) {
        this.subnetArrayID = subnetArrayID;
    }

    public long getAppArrayID() {
        return appArrayID;
    }

    public void setAppArrayID(long appArrayID) {
        this.appArrayID = appArrayID;
    }

    public long getDfConnectionID() {
        return dfConnectionID;
    }

    public void setDfConnectionID(long dfConnectionID) {
        this.dfConnectionID = dfConnectionID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
```

```
        this.type = type;
    }

    public String getSterotype() {
        return sterotype;
    }

    public void setSterotype(String sterotype) {
        this.sterotype = sterotype;
    }

    public Set<Constraint> getConstraintSet() {
        return constraintSet;
    }

    public void setConstraintSet(Set<Constraint> constraintSet) {
        this.constraintSet = constraintSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Subnet)) return false;
        Subnet subnet = (Subnet) o;
        return getSubnetID() == subnet.getSubnetID() &&
                getSubnetArrayID() == subnet.getSubnetArrayID() &&
                getAppArrayID() == subnet.getAppArrayID() &&
                getDfConnectionID() == subnet.getDfConnectionID() &&
                Objects.equals(getName(), subnet.getName()) &&
                Objects.equals(getType(), subnet.getType()) &&
                Objects.equals(getSterotype(), subnet.getSterotype()) &&
                Objects.equals(getConstraintSet(), subnet.getConstraintSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getSubnetID(), getSubnetArrayID(), getAppArrayID(),
getDfConnectionID(), getName(), getType(), getSterotype(), getConstraintSet());
    }
}
```

Subnet Service Implementation

```
@Repository
public class SubnetServiceImpl implements SubnetService {
    @Autowired
    private SubnetDAO objectDAO ;

    @Autowired
    public SubnetServiceImpl(SubnetDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Subnet aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
```

```
    @Transactional
    public void update(Subnet aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Subnet> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Subnet findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Subnet findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.6 Subnet Array

Subnet Array DAO Implementation

```
@Repository
public class SubnetArrayDAOImpl implements SubnetArrayDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<SubnetArray> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<SubnetArray> SubnetArrayList =  session.createQuery("from
SubnetArray").list();

            for(SubnetArray a: SubnetArrayList){
                logger.info("SubnetArray List::" + a);
            }
            return SubnetArrayList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
```

```
        }
    }

    @Override
    public SubnetArray findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SubnetArray a = (SubnetArray) session.load(SubnetArray.class, new
Integer(theId));
            logger.info("SubnetArray loaded successfully, SubnetArray details="+
a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public SubnetArray findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from SubnetArray where name=:name");
            query.setParameter("name", theName);

            SubnetArray a = (SubnetArray) query.uniqueResult();
            logger.info("SubnetArray loaded successfully, SubnetArray details="+
a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(SubnetArray SubnetArrayElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(SubnetArrayElement);
            logger.info("SubnetArray saved successfully, SubnetArray Details="+
SubnetArrayElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(SubnetArray SubnetArray) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(SubnetArray);
            logger.info("SubnetArray updated successfully, SubnetArray Details="+
SubnetArray);
        }
```

```
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            SubnetArray a = (SubnetArray) session.load(SubnetArray.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("SubnetArray deleted successfully, SubnetArray
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Subnet Array Entity

```
@Entity
public class SubnetArray implements java.io.Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long subnetArrayID;
    @Id
    private long dfNetworkID;
    @Id
    private long containsSubnetArrayID;
    private String name;
    private String type;
    @ManyToMany(mappedBy = "subnetArrayID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<Constraint> constraintSet;

    public long getSubnetArrayID() {
        return subnetArrayID;
    }

    public void setSubnetArrayID(long subnetArrayID) {
        this.subnetArrayID = subnetArrayID;
    }

    public long getDfNetworkID() {
        return dfNetworkID;
    }

    public void setDfNetworkID(long dfNetworkID) {
        this.dfNetworkID = dfNetworkID;
    }
```

```
    public long getContainsSubnetArrayID() {
        return containsSubnetArrayID;
    }

    public void setContainsSubnetArrayID(long containsSubnetArrayID) {
        this.containsSubnetArrayID = containsSubnetArrayID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SubnetArray)) return false;
        SubnetArray that = (SubnetArray) o;
        return getSubnetArrayID() == that.getSubnetArrayID() &&
                getDfNetworkID() == that.getDfNetworkID() &&
                getContainsSubnetArrayID() == that.getContainsSubnetArrayID() &&
                Objects.equals(getName(), that.getName()) &&
                Objects.equals(getType(), that.getType());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getSubnetArrayID(), getDfNetworkID(),
getContainsSubnetArrayID(), getName(), getType());
    }
}
```

Subnet Array Service Implementation

```
@Repository
public class SubnetArrayServiceImpl implements SubnetArrayService {
    @Autowired
    private SubnetArrayDAO objectDAO ;

    @Autowired
    public SubnetArrayServiceImpl(SubnetArrayDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(SubnetArray aModelElem) {
        objectDAO.insert(aModelElem);
```

```
    }

    @Override
    @Transactional
    public void update(SubnetArray aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<SubnetArray> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public SubnetArray findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public SubnetArray findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.7 App

App DAO Implementation

```
@Repository
public class AppDAOImpl implements AppDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<App> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<App> AppList =  session.createQuery("from App").list();

            for(App a: AppList){
                logger.info("App List::" + a);
            }
            return AppList;
        }
        catch (Exception e) {
```

```
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public App findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            App a = (App) session.load(App.class, new Integer(theId));
            logger.info("App loaded successfully, App details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public App findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from App where name=:name");
            query.setParameter("name", theName);

            App a = (App) query.uniqueResult();
            logger.info("App loaded successfully, App details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(App AppElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(AppElement);
            logger.info("App saved successfully, App Details="+ AppElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(App App) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(App);
            logger.info("App updated successfully, App Details="+ App);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
```

```
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            App a = (App) session.load(App.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("App deleted successfully, App details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

App Entity

```
@Entity
public class App implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long appID;
    private String name;

    @ManyToMany(mappedBy = "appID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<Constraint> constraintSet;

    public long getAppID() {
        return appID;
    }

    public void setAppID(long appID) {
        this.appID = appID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Set<Constraint> getConstraintSet() {
        return constraintSet;
    }

    public void setConstraintSet(Set<Constraint> constraintSet) {
        this.constraintSet = constraintSet;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof App)) return false;
        App app = (App) o;
        return getAppID() == app.getAppID() &&
                Objects.equals(getName(), app.getName()) &&
                Objects.equals(getConstraintSet(), app.getConstraintSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getAppID(), getName(), getConstraintSet());
    }
}
```

App Service Implementation

```
@Repository
public class AppServiceImpl implements AppService {
    @Autowired
    private AppDAO objectDAO ;

    @Autowired
    public AppServiceImpl(AppDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(App aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(App aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<App> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public App findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
```

```
    @Transactional
    public App findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.8 Apps Array

Apps Array DAO Implementation

```
@Repository
public class AppsArrayDAOImpl implements AppsArrayDAO {

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<AppsArray> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<AppsArray> AppsArrayList =  session.createQuery("from
AppsArray").list();

            for(AppsArray a: AppsArrayList){
                logger.info("AppsArray List::" + a);
            }
            return AppsArrayList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public AppsArray findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            AppsArray a = (AppsArray) session.load(AppsArray.class, new
Integer(theId));
            logger.info("AppsArray loaded successfully, AppsArray details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public AppsArray findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
```

```
                createQuery("from AppsArray where name=:name");
            query.setParameter("name", theName);

            AppsArray a = (AppsArray) query.uniqueResult();
            logger.info("AppsArray loaded successfully, AppsArray details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(AppsArray AppsArrayElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(AppsArrayElement);
            logger.info("AppsArray saved successfully, AppsArray Details="+
AppsArrayElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(AppsArray AppsArray) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(AppsArray);
            logger.info("AppsArray updated successfully, AppsArray Details="+
AppsArray);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            AppsArray a = (AppsArray) session.load(AppsArray.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("AppsArray deleted successfully, AppsArray details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Apps Array Entity

```
@Entity
public class AppsArray implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long appArrayID;

    @ManyToMany(mappedBy = "appArrayID",cascade = CascadeType.REFRESH, fetch =
FetchType.LAZY)
    private Set<Constraint> constraintSet;

    public long getAppArrayID() {
        return appArrayID;
    }

    public void setAppArrayID(long appArrayID) {
        this.appArrayID = appArrayID;
    }

    public Set<Constraint> getConstraintSet() {
        return constraintSet;
    }

    public void setConstraintSet(Set<Constraint> constraintSet) {
        this.constraintSet = constraintSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof AppsArray)) return false;
        AppsArray appsArray = (AppsArray) o;
        return getAppArrayID() == appsArray.getAppArrayID() &&
                Objects.equals(getConstraintSet(), appsArray.getConstraintSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getAppArrayID(), getConstraintSet());
    }
}
```

Apps Array Service Implementation

```
@Repository
public class AppsArrayServiceImpl implements AppsArrayService {
    @Autowired
    private AppsArrayDAO objectDAO ;

    @Autowired
    public AppsArrayServiceImpl(AppsArrayDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(AppsArray aModelElem) {
        objectDAO.insert(aModelElem);
    }
```

```
    @Override
    @Transactional
    public void update(AppsArray aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<AppsArray> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public AppsArray findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public AppsArray findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.9 Connection

Connection DAO Implementation

```
@Repository
public class DFConnectionDAOImpl implements DFConnectionDAO {

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<DFConnection> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<DFConnection> DFConnectionList =  session.createQuery("from
DFConnection").list();

            for(DFConnection a: DFConnectionList){
                logger.info("DeploymentMode List::" + a);
            }
            return DFConnectionList;
        }
        catch (Exception e) {
```

```
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public DFConnection findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            DFConnection a = (DFConnection) session.load(DFConnection.class, new
Integer(theId));
            logger.info("DFConnection loaded successfully, DFConnection details="+
a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public DFConnection findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from DeploymentMode where name=:name");
            query.setParameter("name", theName);

            DFConnection a = (DFConnection) query.uniqueResult();
            logger.info("DeploymentMode loaded successfully, DeploymentMode
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(DFConnection DFConnectionElement) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(DFConnectionElement);
            logger.info("DFConnection saved successfully, DFConnection Details="+
DFConnectionElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(DFConnection DFConnection) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(DFConnection);
            logger.info("DFConnection updated successfully, DFConnection
```

```
Details="+ DFConnection);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            DFConnection a = (DFConnection) session.load(DFConnection.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("DFConnection deleted successfully, DFConnection
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Connection Entity

```
@Entity
public class DFConnection implements java.io.Serializable{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long dfConnectionID;
    private String connectionBody;

    public long getDfConnectionID() {
        return dfConnectionID;
    }

    public void setDfConnectionID(long dfConnectionID) {
        this.dfConnectionID = dfConnectionID;
    }

    public String getConnectionBody() {
        return connectionBody;
    }

    public void setConnectionBody(String connectionBody) {
        this.connectionBody = connectionBody;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof DFConnection)) return false;
        DFConnection that = (DFConnection) o;
```

```
        return getDfConnectionID() == that.getDfConnectionID() &&
                Objects.equals(getConnectionBody(), that.getConnectionBody());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getDfConnectionID(), getConnectionBody());
    }
}
```

Connection Service Implementation

```
@Repository
public class DFConnectionServiceImpl implements DFConnectionService {
    @Autowired
    private DFConnectionDAO objectDAO ;

    @Autowired
    public DFConnectionServiceImpl(DFConnectionDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(DFConnection aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(DFConnection aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<DFConnection> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public DFConnection findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public DFConnection findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.10 Constraint

Constraint DAO Implementation

```
@Repository
public class ConstraintDAOImpl implements ConstraintDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<Constraint> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<Constraint> ConstraintList =  session.createQuery("from
Constraint").list();

            for(Constraint a: ConstraintList){
                logger.info("Constraint List::" + a);
            }
            return ConstraintList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public Constraint findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Constraint a = (Constraint) session.load(Constraint.class, new
Integer(theId));
            logger.info("Constraint loaded successfully, Constraint details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public Constraint findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from Constraint where name=:name");
            query.setParameter("name", theName);

            Constraint a = (Constraint) query.uniqueResult();
            logger.info("Constraint loaded successfully, Constraint details="+ a);
            return a;
```

```
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(Constraint ConstraintElementObjectInsert) {
        try {
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(ConstraintElementObjectInsert);
            logger.info("Constraint saved successfully, Constraint Details="+
ConstraintElementObjectInsert);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(Constraint ConstraintobjectUpdate) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(ConstraintobjectUpdate);
            logger.info("Constraint updated successfully, Constraint Details="+
ConstraintobjectUpdate);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Constraint a = (Constraint) session.load(Constraint.class, new
Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("Constraint deleted successfully, Constraint details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Constraint Entity

```
@Entity
public class Constraint implements java.io.Serializable{

    @Id
```

```
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long constraintID;
    private String name;
    private String description;
    private String constraint;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="constraitscenarioobjectjoin",
            joinColumns = @JoinColumn(name="constraintID", unique = true),
            inverseJoinColumns = @JoinColumn(name="scenarioObjectID", unique =
true))
    private Set<ScenarioObject> scenarioObjectSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="constraintsubnetjoin",
            joinColumns = @JoinColumn(name="constraintID", unique = true),
            inverseJoinColumns = @JoinColumn(name="subnetID", unique = true))
    private Set<Subnet> subnetSet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="constraintsubnetarrayjoin",
            joinColumns = @JoinColumn(name="constraintID", unique = true),
            inverseJoinColumns = @JoinColumn(name="subnetArrayID", unique = true))
    private Set<SubnetArray> subnetArraySet;

    @ManyToMany(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinTable(name ="constraintappjoin",
            joinColumns = @JoinColumn(name="constraintID", unique = true),
            inverseJoinColumns = @JoinColumn(name="appID", unique = true))
    private Set<App> appSet;

    public long getConstraintID() {
        return constraintID;
    }

    public void setConstraintID(long constraintID) {
        this.constraintID = constraintID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getConstraint() {
        return constraint;
    }

    public void setConstraint(String constraint) {
```

```
        this.constraint = constraint;
    }

    public Set<ScenarioObject> getScenarioObjectSet() {
        return scenarioObjectSet;
    }

    public void setScenarioObjectSet(Set<ScenarioObject> scenarioObjectSet) {
        this.scenarioObjectSet = scenarioObjectSet;
    }

    public Set<Subnet> getSubnetSet() {
        return subnetSet;
    }

    public void setSubnetSet(Set<Subnet> subnetSet) {
        this.subnetSet = subnetSet;
    }

    public Set<SubnetArray> getSubnetArraySet() {
        return subnetArraySet;
    }

    public void setSubnetArraySet(Set<SubnetArray> subnetArraySet) {
        this.subnetArraySet = subnetArraySet;
    }

    public Set<App> getAppSet() {
        return appSet;
    }

    public void setAppSet(Set<App> appSet) {
        this.appSet = appSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Constraint)) return false;
        Constraint that = (Constraint) o;
        return getConstraintID() == that.getConstraintID() &&
                Objects.equals(getName(), that.getName()) &&
                Objects.equals(getDescription(), that.getDescription()) &&
                Objects.equals(getConstraint(), that.getConstraint()) &&
                Objects.equals(getScenarioObjectSet(),
that.getScenarioObjectSet()) &&
                Objects.equals(getSubnetSet(), that.getSubnetSet()) &&
                Objects.equals(getSubnetArraySet(), that.getSubnetArraySet()) &&
                Objects.equals(getAppSet(), that.getAppSet());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getConstraintID(), getName(), getDescription(),
getConstraint(), getScenarioObjectSet(), getSubnetSet(), getSubnetArraySet(),
getAppSet());
    }
}
```

Constraint Service Implementation

```
@Repository
public class ConstraintServiceImpl implements ConstraintService {
    @Autowired
    private ConstraintDAO objectDAO ;

    @Autowired
    public ConstraintServiceImpl(ConstraintDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(Constraint aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(Constraint aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<Constraint> findAll() {
        return objectDAO.findAll();
    }

    @Override
    @Transactional
    public Constraint findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public Constraint findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.2.11 Data Fabrication Expected Trace

Data Fabrication Expected DAO Implementation

```
@Repository
public class DFExpectedTraceDAOImpl implements DFExpectedTraceDAO{

    private Logger logger   = LoggerFactory.getLogger(this.getClass());
    @Autowired
    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sessionFactory) {
```

```
        this.sessionFactory = sessionFactory;
    }

    @Override
    public List<DFExpectedTrace> findAll() {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            List<DFExpectedTrace> DFExpectedTraceList =  session.createQuery("from
DFExpectedTrace").list();

            for(DFExpectedTrace a: DFExpectedTraceList){
                logger.info("DFExpectedTrace List::" + a);
            }
            return DFExpectedTraceList;
        }
        catch (Exception e) {
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public DFExpectedTrace findById(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            DFExpectedTrace a = (DFExpectedTrace)
session.load(DFExpectedTrace.class, new Integer(theId));
            logger.info("DFExpectedTrace loaded successfully, DFExpectedTrace
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }

    }

    @Override
    public DFExpectedTrace findByName(String theName) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            Query query= session.
                    createQuery("from DFExpectedTrace where name=:name");
            query.setParameter("name", theName);

            DFExpectedTrace a = (DFExpectedTrace) query.uniqueResult();
            logger.info("DFExpectedTrace loaded successfully, DFExpectedTrace
details="+ a);
            return a;
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void insert(DFExpectedTrace DFExpectedTraceElement) {
        try {
```

```
            Session session = this.sessionFactory.getCurrentSession();
            session.persist(DFExpectedTraceElement);
            logger.info("DFExpectedTrace saved successfully, DFExpectedTrace
Details="+ DFExpectedTraceElement);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    @Override
    public void update(DFExpectedTrace DFExpectedTrace) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            session.update(DFExpectedTrace);
            logger.info("DFExpectedTrace updated successfully, DFExpectedTrace
Details="+ DFExpectedTrace);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;
        }
    }

    @Override
    public void remove(int theId) {
        try{
            Session session = this.sessionFactory.getCurrentSession();
            DFExpectedTrace a = (DFExpectedTrace)
session.load(DFExpectedTrace.class, new Integer(theId));
            if(null != a){
                session.delete(a);
            }
            logger.info("DFExpectedTrace deleted successfully, DFExpectedTrace
details="+a);
        }
        catch (Exception e){
            logger.error(e.getMessage());
            throw e;

        }

    }
}
```

Data Fabrication Expected Entity

```
@Entity
public class DFExpectedTrace implements java.io.Serializable{

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private long dfExpectedTraceID;
    private String value;
    @Id
    private long dfModelID;

    public long getDfExpectedTraceID() {
        return dfExpectedTraceID;
    }
```

```
    public void setDfExpectedTraceID(long dfExpectedTraceID) {
        this.dfExpectedTraceID = dfExpectedTraceID;
    }

    public String getValue() {
        return value;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof DFExpectedTrace)) return false;
        DFExpectedTrace that = (DFExpectedTrace) o;
        return getDfExpectedTraceID() == that.getDfExpectedTraceID() &&
                dfModelID == that.dfModelID &&
                Objects.equals(getValue(), that.getValue());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getDfExpectedTraceID(), getValue(), dfModelID);
    }
}
```

Data Fabrication Expected Service Implementation

```
@Repository
public class DFExpectedTraceServiceImpl implements DFExpectedTraceService {
    @Autowired
    private DFExpectedTraceDAO objectDAO ;

    @Autowired
    public DFExpectedTraceServiceImpl(DFExpectedTraceDAO constructorDAO) {
        this.objectDAO = constructorDAO;
    }

    @Override
    @Transactional
    public void insert(DFExpectedTrace aModelElem) {
        objectDAO.insert(aModelElem);
    }

    @Override
    @Transactional
    public void update(DFExpectedTrace aModelElem) {
        objectDAO.update(aModelElem);
    }

    @Override
    @Transactional
    public void remove(int theId) {
        objectDAO.remove(theId);
    }

    @Override
    @Transactional
    public List<DFExpectedTrace> findAll() {
        return objectDAO.findAll();
    }
```

```
    @Override
    @Transactional
    public DFExpectedTrace findById(int theId) {
        return objectDAO.findById(theId);
    }

    @Override
    @Transactional
    public DFExpectedTrace findByName(String theName) {
        return objectDAO.findByName(theName);
    }
}
```

3.2.8.3 Database

Based on the above, a database structure was created to illustrate the database tables for data fabrication, as depicted in Figure 11.

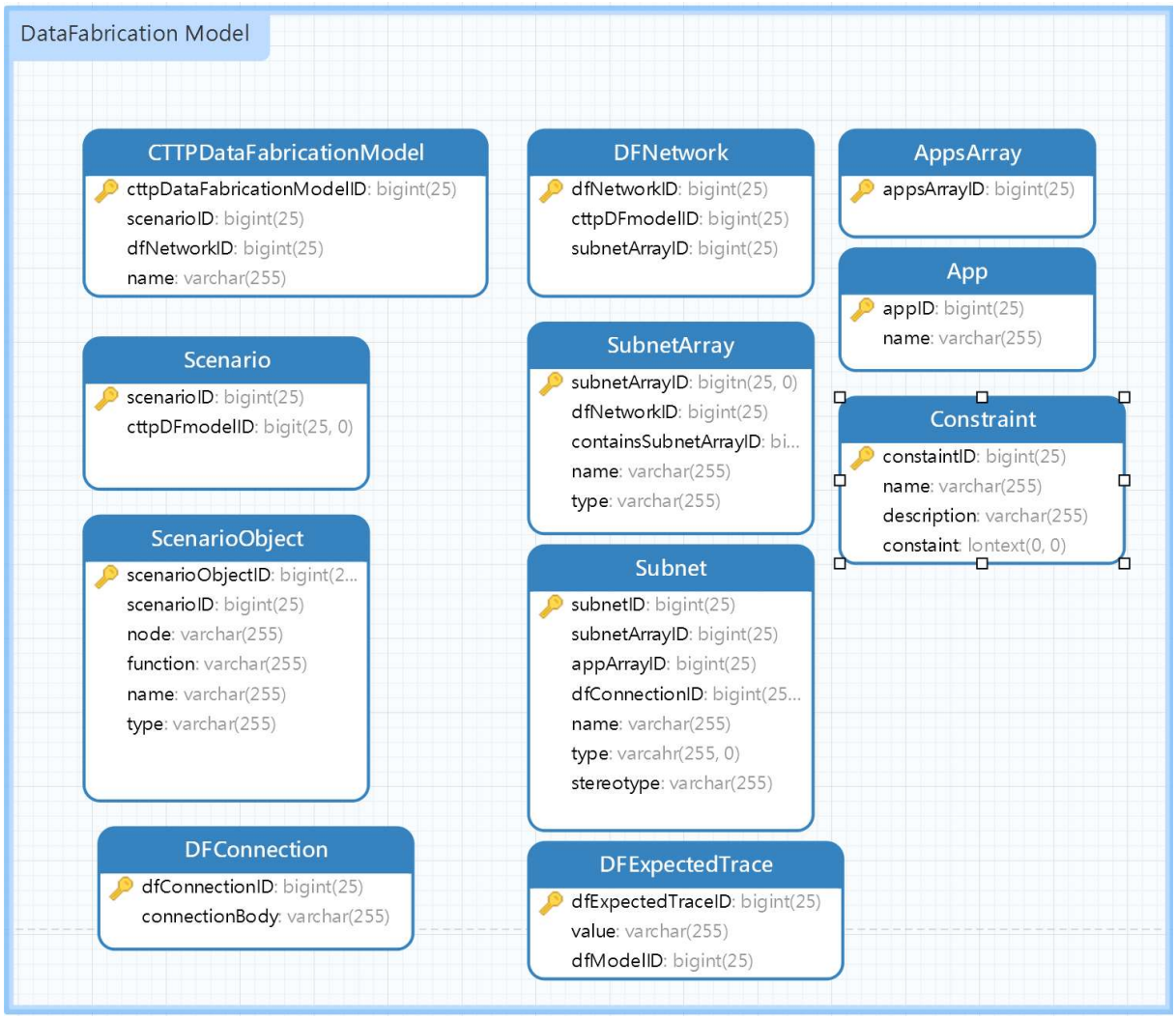

Figure 11: Data Fabrication Model database schema

4 Deployment guide

4.1 The CTTP creator and editor

In order to create and modify the CTTP language, we used the ANTLR plugins (ANTLR Development Tools, 2019) described in the subsections below.

4.1.1 The IntelliJ Plugin for ANTLR4

This tool provided: (i) syntax highlighting and syntax error checking, (ii) code completion for tokens and rule names, and (iii) live grammar interpreter for grammar preview. More specifically, upon completing the CTTP grammar, we utilised this tool to create the CTTP model and edit it, if needed. This plugin allowed us to lively interact with the rules specified in the grammar in order to produce a well-formatted output.

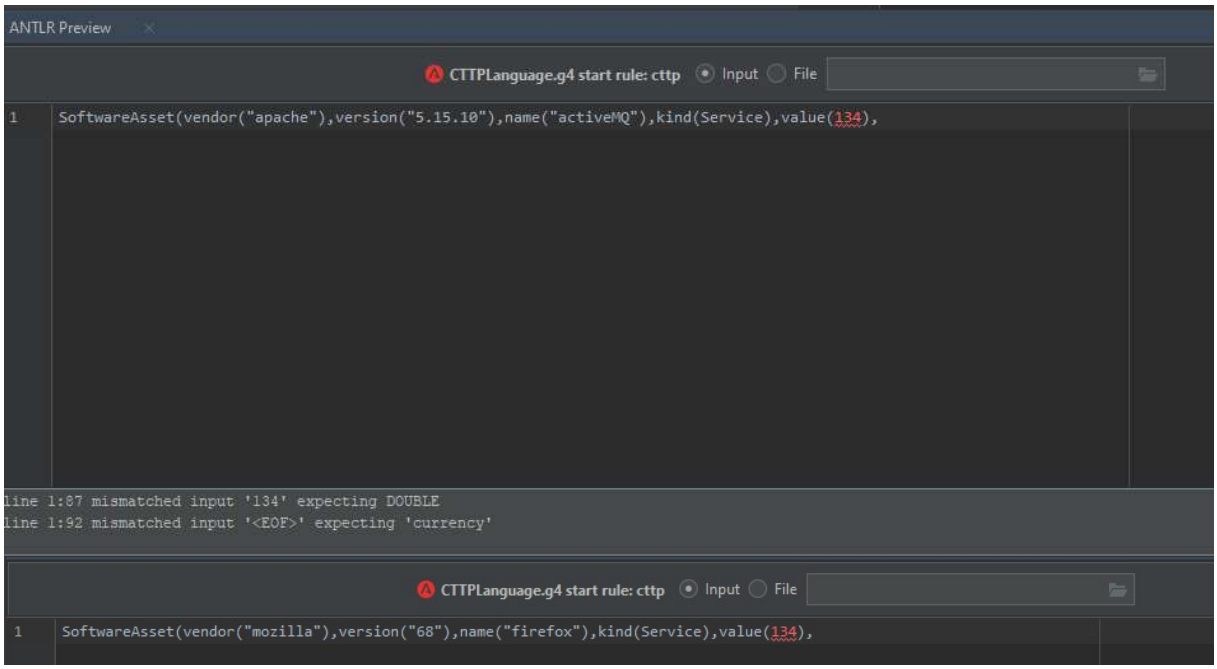

Figure 12: CTTP creator error

Figure 12 presents the live notification feature of the plugin. Given the CTTP grammar, the editor expects a set of rules to create a well-structure document. In this example, we tried to add an integer as the value of the software asset while the grammar expected a double.

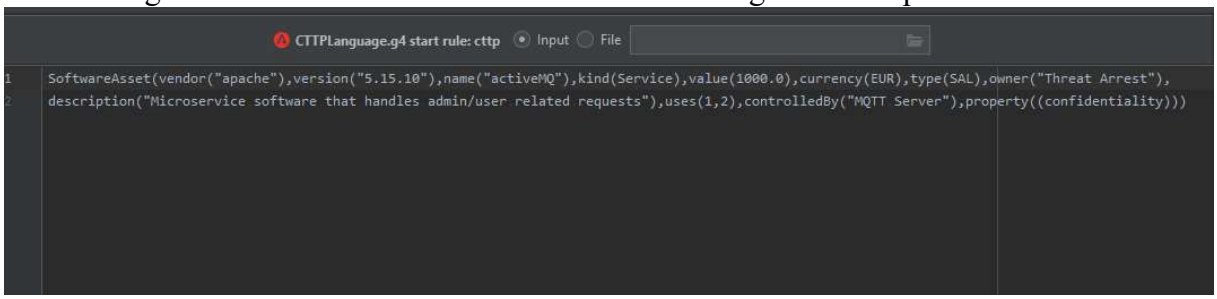

Figure 13: Software asset creation

Figure 13 presents the creation of a Software Asset that follows the CTTP grammar format and produces a correct output. The described asset is the apache ActiveMQ message broker (Snyder et al., 2011) that acts as service and needs to be assessed against the confidentiality of the data that it processes.

4.1.2 Alternative ANTLR4 Plugins

Instead of the plugins presented above, there are a number of alternatives depending on the development environment.

4.1.3 AntlrDT Tools Suite for Eclipse

The AntlrDT suite (AntlrDT, 2019) contains the XVisitorDT grammar and editor builder that provides the following capabilities (ANTLR Development Tools, 2019):

- full syntax-directed editor with outline view
- hyperlinked navigation between rules
- automatic builder with real-time problem feedback markers
- builder performs automatic visitor code generation
- utilizes fully symbolic XPath-style rule path references
- ANTLR grammar and Java native code formatter

4.1.4 Visual Studio IDE extension for ANTLR 4 (ANTLR for Visual Studio IDE, 2019)

The visual studio extensions provide (ANTLR Development Tools, 2019):

- Syntax highlighting, including within actions and semantic predicates
- Outlining support for quickly collapsing rules (v3 only)
- Dropdown bars listing parser and lexer rules within the current grammar (v3 only)
- Support for Autocomplete, Quick Info, and Go to Definition (v3 only)

4.2 CTTP Models and Programmes Specification Tool example

The CTTP parser was exposed as a RESTful API. An example of a creation of a software asset using the CTTP editor is presented in Figure 13. Figure 14 presents the POST method that accepts the CTTP Model and distributes the attributes to its corresponding tables. The tool we use to test the API is Postman. Upon successful insertion, the method returns a *HTTP OK status (200)*, elsewhere it prints an error code that notifies the user for the exact reason the parser failed to store the model to the tables.

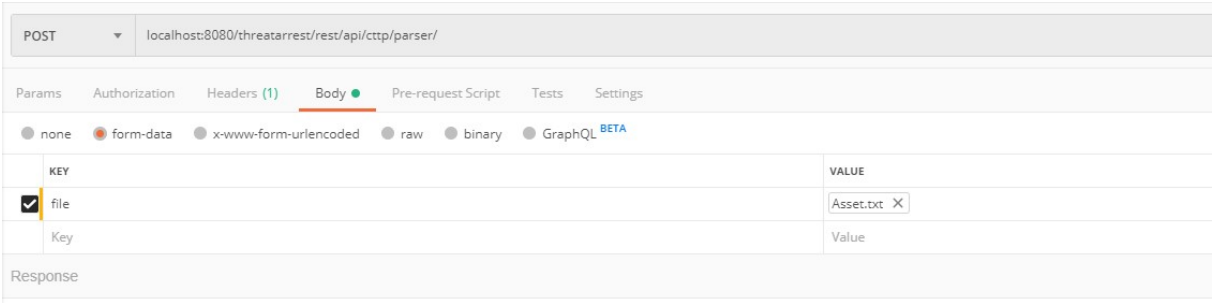

Figure 14: Grammar RESTful API

The *asset.txt* file contained the MySQL database described as a software asset of type SAL. The MySQL was used as a service, thus the *kindID* is equal to *1* while the status of the asset is set to final (*statusID=2*) because it contained all the information required by the tool (see Figure 15).

	softwareassetID	assetName	assetVendor	assetVersion	assetValue	currencyID	typeID	assetID	kindID	statusID
▶	1	MySQL	MySQL	5.7	NULL	NULL	1	2	1	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 15: Asset insertion to the database

5 How to Establish the CTTP Programmes and Models

In this Section, we will present how to collect the appropriate information for the various concepts of the CTTP programmes and models that were described earlier, and how to establish them.

5.1 Lifecycle

The THREAT-ARREST approach (Hatzivasilis et al., 2020) is composed of 4 main phases (see Figure 16): i) analysis, ii) programme establishment, iii) training and user feedback, and iv) post-training monitoring and security evaluation.

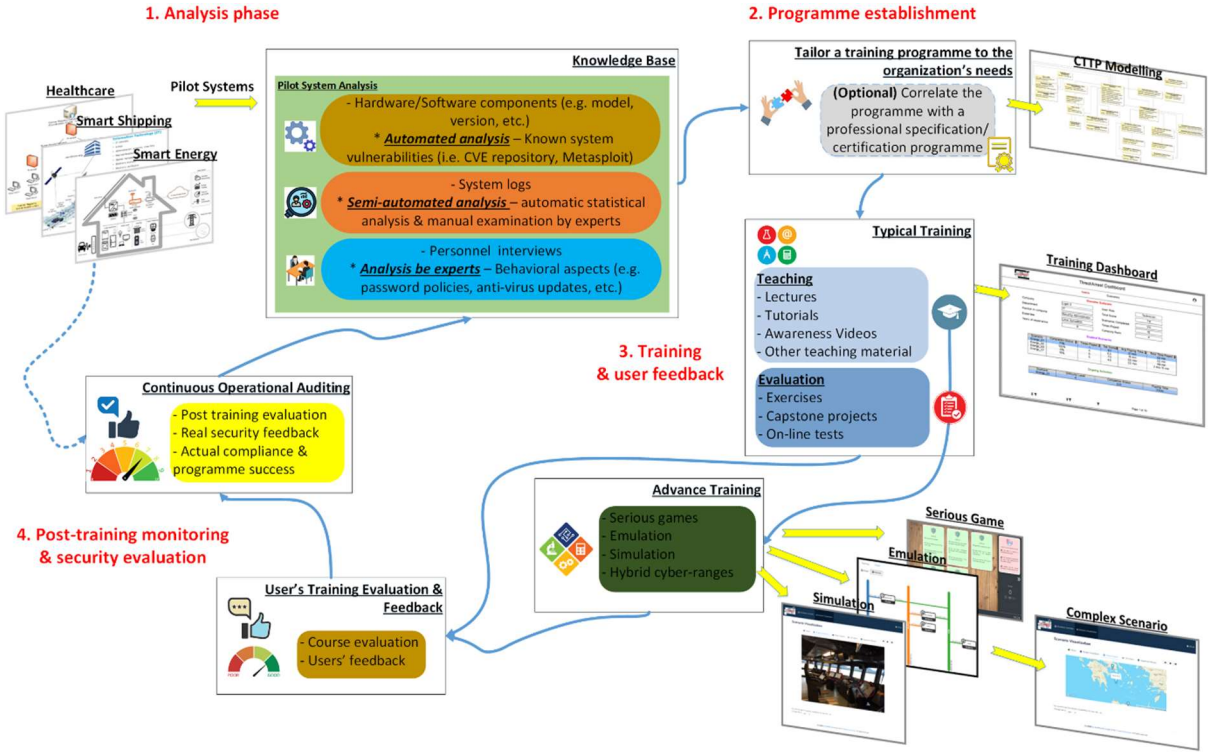

Figure 16 The THREAT-ARREST lifecycle

5.2 Initial Analysis of a Pilot System

At first, we analyse the customer organization system and establish the knowledge base for the training programme. In our case, this includes the three pilot systems for the maritime, healthcare, and smart energy use cases, respectively. The goal is to estimate the current security status and specify the weak points (e.g. system or behavioural vulnerabilities). The platform's Assurance Tool installs monitoring modules in the pilot system that capture its technical aspects (like the type and version of the running software of the installed hardware components) and check if it operates in a secure manner. Then, it searches to widely-known security repositories (i.e. CVE) and automatically discovers the active vulnerabilities of the system (e.g. if a server uses MSQL 5.5.35, then it is vulnerable to buffer overflow attacks based on the CVE-2014-0001). The vulnerabilities set is evaluated in a semi-automated fashion by the experts, who identify the most significant of them for the examined organization. Based on this information, we define the ***Core Assurance Model*** (subsection 3.2.3).

Experts also interview the organizations personnel and record the followed operational procedures (e.g. password-update policy, anti-virus updates, etc.). The training programme is developed afterwards based on the overall outcomes of the initial analysis.

Also, during this phase, the experts collect real-operational log or other data files from the examined system. This knowledge is further processed in order to enhance the advance training procedures of the THREAT-ARREST platform. At first, we perform statistical analysis on the original data in order to disclose the statistical patterns of each file. This is performed either through manual examination by experts or via an automatic statistical analysis module. The goal is to produce synthetic events (i.e. a series with legitimate and/or phishing emails) or other data (i.e. a database's content with dummy but realistic entries) via our Data Fabrication Tool that will be later used in order to provide advance training under realistic conditions.

5.3 CTTP Programme Establishment

Then, based on the initial analysis results, we tailor a CTTP programme to the organization's special needs, which could also be combined and cover the training for a professional certification programme (e.g. Certified Information Security Manager (CISM) by ISACA or Certified Information Systems Security Professional (CISSP) by ISC2), in order to increase the THREAT-ARREST's efficiency. Therefore, we define the main parameters of the ***Training Programme*** (subsection 3.2.4), like the programme's goals, actuators, trainee rules, etc.

Afterwards, we collect the related teaching material for the typical training (e.g. lectures, tutorials, awareness videos, etc.) and model the advance training scenarios based on ***Simulation Model*** (subsection 3.2.5), ***Emulation Model*** (subsection 3.2.6), and ***Gamification Model*** (subsection 3.2.7), as well as the ***Data Fabrication Model*** (subsection 3.2.8) for the required synthetic data.

5.4 Training and User Feedback

Once the trainee has completed the basic training for a learning unit, the accompanied CTTP models are activated in the Dashboard and the trainee can now proceed with the advanced training. The CTTP models describe a virtual system and how to instantiate it via the Emulation, Simulation, and Gamification Tools, respectively.

These virtual labs and digital twins, which could resemble the organization's actual system and followed procedures, offer hands-on experience to the trainees/personnel. Thus, they can test and evaluate new policies and technologies, break-down the system, restore the default state and start over again, without affecting the real system. The trainees start the programme, consume the teaching material and are evaluated against the desired learning goals.

After the completion of the training, the platform displays the results for each trainee and the programme as a whole. This process indicates the scores of the trained personnel and their achievements regarding the educational procedures. Finally, the trainees can also complete questionnaires and provide feedback to the THREAT-ARREST operator, e.g. for the platform modules, the programme, etc., in order to update and improve our system. All these form ordinary features of training platforms.

5.5 Post-Training Monitoring and Security Evaluation

However, the successful completion of a programme does not always reflect to the improvement of the pilot organization's security in a straightforward manner. The security level can be increased only if the trainees apply what they have learnt in the actual system (Manifavas et al., 2014). The evaluation of this phase is one of the THREAT-ARREST's novelties in contrast to other competitive solutions.

Thus, our platform continues to audit the pilot system for a determined period after the training phases. The deployed controls from the initial phase continuously assure the organizations security-sensitive components. The goal is to capture if the trainees really applied what they were toughed.

For example, in the analysis phases we discover that the trainees do not update their email passwords in a regular basis, i.e. by examining the log-file of the mailing server (Assurance Model). Thus, we tailor a programme to include the learning topic of password management (Training Programme and Simulation, Emulation, Gamification, and Data Fabrication Models). When the programme is finished, we inspect the servers log and check if the password-update entries have been increased or not.

The confirmation that the personnel adheres with the learned features, and thus the system's security is really improved, constitutes the actual evaluation that the programme was successful. This process is facilitated by the Assurance Tool and the related model. Feedback is gathered from this phase in order to improve the THREAT-ARREST's operation for future training iterations and new programmes.

6 Conclusion

This deliverable documented the guideline of the THREAT-ARREST CTTP Model component. This deliverable is the last deliverable of the task "T3.1 – CTTP Language definition and tool support". As such, it presents the implementation approach, the updates that occurred in the language and the deployment guide in order to utilize it.

In more details, we described the two main technologies that we utilized to create the parser and call the RESTful API to store the model to the database, the parser for the core assurance model and for each sub model described in the deliverable "D3.1 – CTTP Models and Programmes Specification Language" and provided an example for defining a CTTP element by using the editor.

Based on the editor, a CTTP model can be created and stored to the database. The component allows the end-user to edit the CTTP model, if needed, and exposes a number of RESTful APIs to distribute the sub-models to the different THREAT-ARREST tools.

This deliverable "D3.2 – CTTP Models and Programmes Specification Tool" is the final outcome of the task "T3.1 – CTTP Language definition and tool support". As documented, the editor will now be used for the development of reference CTTP models and programs for all the three pilots of THREAT-ARREST.

7 References

[1] ANTLR Development Tools, 2019. *ANTLR Development Tools.* [Online] Available at: https://www.antlr.org/tools.html

[2] ANTLR for Visual Studio IDE, 2019. *ANTLR Language Support.* [Online] Available at: https://marketplace.visualstudio.com/items?itemName=SamHarwell.ANTLRLanguageSupport

[3] Antlr.org, 2019. *ANTLR.* [Online] Available at: https://www.antlr.org/

[4] Antlr4 - Visitor vs Listener Pattern, 2019. *Antlr4 - Visitor vs Listener Pattern.* [Online] Available at: https://saumitra.me/blog/antlr4-visitor-vs-listener-pattern/

[5] AntlrDT, 2019. *AntlrDT.* [Online] Available at: https://marketplace.eclipse.org/content/antlrdt

[6] CUMULUS Project, 2012. *Certification infrastructure for multi-layer cloud services project*. D2.2 Certification models. [Online] Available at: http://cordis.europa.eu/docs/projects/cnect/0/318580/080/deliverables/001-D22Certificationmodelsv1.pdf

[7] Du Hu, et al., 2018. *Development of a REST API for obtaining site-specific historical and near-future weather data in EPW format*. Building Simulation and Optimization, Emmanuel College, University of Cambridge, 11-12 September, pp. 1-6.

[8] Hatzivasilis, G., et al., 2019a. Towards the Insurance of Healthcare Systems. 1st Model-driven Simulation and Training Environments for Cybersecurity (MSTEC), ESORICS, Springer, LNCS, vol. 11981, Luxembourg, 27 September 2019, pp. 1-14.

[9] Hatzivasilis, G., et al., 2019b. Cyber Insurance of Information Systems. 24th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2019), IEEE, Limassol, Cyprus, 11-13 September 2019, pp. 1-7.

[10] Hatzivasilis, G., et al., 2020. Cyber-Ranges as a Mean of Security Culture Establishment. ERCIM News – Special Theme: The Climate Action, ERCIM, April 2020, issue 121, Article no. 36, pp. 36-37.

[11] Hibernate, 2019. *Hibernate.* [Online] Available at: https://hibernate.org/

[12] ISO/IEC, 1996. *Information technology - Syntactic metalanguage - Extended BNF*. ISO/IEC 14977, pp. 1-14.

[13] Krotsiani M., Kloukinas C., Spanoudakis G., 2017. *Cloud Certification Process Validation using Formal Methods*. 15th International Conference on Service Oriented Computing (ICSOC 2017), Malaga, Spain, November.

[14] Manifavas, C., et al., 2014. DSAPE – Dynamic Security Awareness Program Evaluation. Human Aspects of Information Security, Privacy and Trust (HCI International 2014), 22-27 June, 2014, Creta Maris, Heraklion, Crete, Greece, Springer, LNCS, vol. 8533, pp. 258-269.

[15] Palacios M., Garcia-Fanjul J., Tuya J., Spanoudakis G., 2015. *Coverage Based Testing for Service Level Agreements*. IEEE Transactions on Services Computing, vol. 8, issue 2, March, pp. 299 – 313.

[16] Postman, 2019. *The Collaboration Platform for API Development*. Online] Available at: https://www.getpostman.com/

[17] Scowen R. S., 1993. *Extended BNF - A generic base standard*. Software Engineering Standards Symposium, 30 August, pp. 1-10.

[18] Snyder Bruce, Bosanac Dejan, and Davies Rob, 2011. ActiveMQ in Action. Manning Publications, pp. 1-408.

[19] Spring Boot, 2019. *Spring Boot.* [Online] Available at: https://spring.io/projects/spring-boot

[20] YAML, 2019. *The Official YAML website*. [Online] Available at: https://yaml.org/