FP7 Grant Agreement N° 312450

CIPRNet

Critical Infrastructure Preparedness and Resilience Research Network

Project type: Network of Excellence (NoE)

Thematic Priority: FP7 Cooperation, Theme 10: Security

Start date of project: March 1, 2013 Duration: 48 months

D6.4 Implementation of the integrated CIP MS&A based 'what if' analysis

Due date of deliverable: 31/03/2016
Actual submission date: 21/04/2016

Revision: Version 1

Fraunhofer Institute for Intelligent Analysis and Information Systems (Fraunhofer)

Project co-funded by the European Commission within the Seventh Framework Programme (2007–2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Author(s)	Jingquan Xie, Betim Sojeva, Stefan Rilling, Thomas Doll, Norman Voß, Erich Rome (Fraunhofer)
Contributor(s)	Alberto Tofani (ENEA)

Security Assessment	Dominique Sérafin, CEA
Approval Date	18.04.2016
Remarks	No issues found

The project CIPRNet has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 312450.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Purpose of this document

This deliverable belongs to work package WP6: Joint Research Activities 1: CI fMS&A and external threats modelling for 'what if' analysis [DoW]. D6.4 describes the implementation of CIPRTrainer, a technical demonstrator for the 'what if' analysis capability. It is a new application that provides a new capability for training crisis management (CM) staff, that is, exploring different courses of action and comparing their consequences ('what if' analysis), based on federated modelling, simulation and analysis (fMS&A). A first outline of CIPRTrainer had been presented in deliverable D6.3 "Scenario models" [CIPRNetD63]. In D6.4 we will present the implementation of CIPRTrainer in great detail.

The implementation is the final development phase of the four-layered architectural approach (MCRI) presented in D6.1 "Conceptual design of a federated and distributed cross-sector and threat simulator" [CIPRNetD61], the first deliverable in WP6. This deliverable D6.4 describes

- The realised technical (sub-)systems of which CIPRTrainer is composed,
- Technical solutions for communication between such (sub-)systems, including interoperability middleware for federated simulation,
- The implementation of concepts like complex event processing and their impact on the overall system implementation,
- Certain file formats, and more.

Bottom line is that D6.4 describes in detail how CIPRTrainer works internally and how it was implemented. In particular, the current implementation is related to the Emmerich train derailment scenario described in [CIPRNetD62].

1.2 Relation to other CIPRNet deliverables

The ultimate tangible outcome of WP6 is CIPRTrainer, a technical demonstrator for the 'what if' analysis capability. The development of CIPRTrainer followed the four-layered MCRI approach (mission, concepts, realisation, and implementation), introduced in deliverable D6.1 "Conceptual design of a federated and distributed cross-sector and threat simulator" [CIPRNetD61]. While D6.1 covered the top two layers, mission and concepts, and parts of the realisation layer, D6.4 covers the remainder of the realisation layer and the implementation layer.

In order to test and to use CIPRTrainer, it was also a prerequisite to develop application scenarios, that is, complex cross-border crisis scenarios that constitute the setting for the training of crisis managers. Those scenarios have been described in D6.2 "Application scenario" [CIPRNetD62]. The nature of the scenarios determines to some extent also what technical subsystems are needed, like Critical Infrastructure simulators and threat simulations. In order to make the scenarios usable inside CIPRTrainer, one more step were necessary, namely the creation of models of these scenarios. The deliverable D6.3 "Scenario models" [CIPRNetD63] describes how these models have been created, on what data they were based, and how we coped with missing data.[CIPRNetD62] D6.4 contains references to all those deliverables.

1.3 Structure of the document

The remainder of this document is structured as follows: In Section 2, we describe CIPRTrainer as a distributed technical system consisting of a front-end and a back-end that can be operated in two different modes (training and operation). We also briefly elaborate on the

method of test-driven development. Section 3 presents the **syntactical** technical basis for integration of the components of this distributed system, namely RESTful services. Section 4 describes the SyMo-based scenario development, that is, the usage of the external tool SyMo as a scenario editor and the creation of scenario files that contain descriptions of scenario events. Section 5 continues with describing the **semantic** basis for integration, namely declarative integration with Complex Event Processing. In Section 6 we present the major elements of federated simulation: simulators, simulator adapters, the time management, refined scenarios (including artificial models for parts of Critical Infrastructures for which we could not get data). In Section 7 we describe the federated simulation-based what-if analysis, including the novel rollback mechanism, and the new impact and consequence analysis module (CAM). In the final Sections 8 we conclude by summarising the main achievements, system limitations and highlight the future tasks. An overview of system components and their intended users is provided in Figure 1. System implementation details will be provided in the remaining parts of this document.

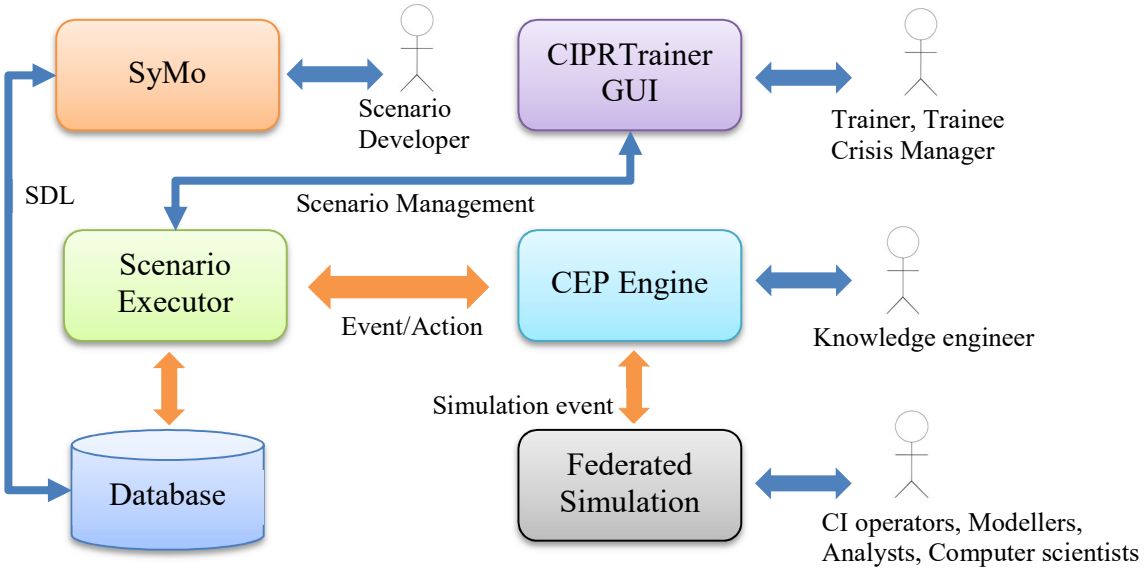


Figure 1: Overview of system components and their intended users

2 CIPRTrainer

The CIPRTrainer is a web-based training application, which is accessible through common web-browsers. It does not need further installation processes on the user machine in order to run the training application. The advantages of web-based solutions are as follows: today most end devices (mobile and desktop) provide at least one Web browser with HTML5/CSS3 support; the application is easily accessible, and no strict performance requirements of the devices are needed.

The first part in this section gives brief description about the system architecture of CIPR-Trainer followed by the development principals, which elucidate the best practice methods of how to develop a web-based application including the test-driven development approach.

The next two sections describe the architecture, techniques and frameworks used for the front and back-end development of CIPRTrainer. In addition, the *Scenario Executor* module that is responsible for running the pre-developed crisis scenario is examined in more details since this is one of core components in CIPRTrainer. This is followed by a brief description about the Open Geospatial Consortium (OGC) Standards WFS and WMS, and their advantages. Furthermore, some notable tools are presented, which are useful when dealing with spatial data. The last section gives a brief discussion of two possible running modes of using CIPR-Trainer: training and operation mode.

2.1 CIPRTrainer in a nutshell

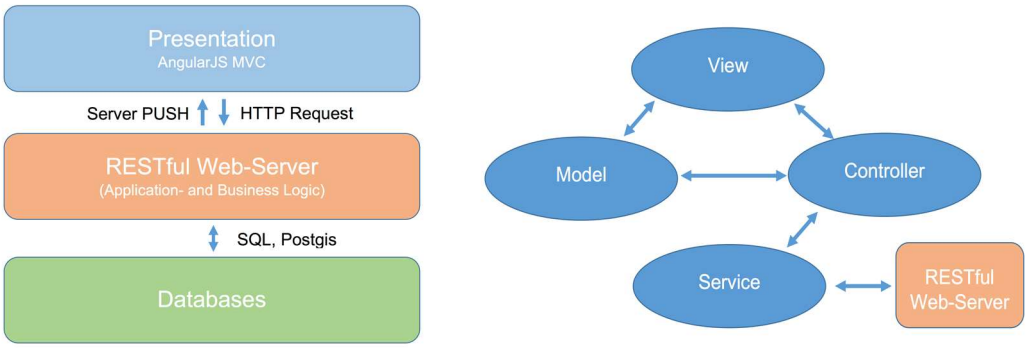


Figure 2: System architecture of the CIPRTrainer (left) and AngularJS MVC pattern (right).

The CIPRTrainer is a standard Single Page Web Application, which uses modern HTML5 Web technologies. It is accessible through the regular HTTP/HTTPS protocol (regular Web browsers). The front-end is implemented using the AngularJS MVW[1], which is a variant of MVC (Model-View-Controller) framework; see the right part of Figure 2. The content is loaded dynamically by sending AJAX-requests and receiving HTML-PUSH-notifications from the NodeJS application server[2]. The system embraces a three-tier-architecture, which contains a presentation, logic and data-tier (see Figure 2, the left). The front-end (presenta-

[1] https://angularjs.org/

[2] https://nodejs.org https://nodejs.org

tion-tier) is implemented using the AngularJS framework provided by Google Inc. AngularJS provides models that are bound to the view-layer. These models can be manipulated through AngularJS controller-functions. AngularJS service-functions handle typically the communication to the RESTful-API. The web or application-server (logic-tier) is based on NodeJS, which incorporates an event-driven JavaScript runtime environment. It incorporates the application- and business logic, and provides a RESTful Web Services (see Chapter 3) for What-if Analysis (WIA) and other capabilities. The application server also includes scenario services, and the federated simulation controller that is able to set up, start and stop the federated simulation (see Chapter 6). The web-server has access to the databases, which serialize spatial- and socio-demographic data, CI models, user configurations and training protocols.

2.2 Test-Driven Development

The application of Test-Driven Development (TDD) on complex software projects has shown significant improvements concerning the software quality [TDD03]. TDD is defined as "a software development strategy that requires automated tests be written prior to writing functional code in small, rapid iterations" [JAN2006]. In other words, when a software developer desires to add a new feature to the software system, the best practice is to first develop a test that incorporates the specification and validation of the function [TDD03]. The development of the CIPRTrainer front-end follows this approach by utilising Jasmine, "a behaviour-driven development framework for testing JavaScript code" [JASM]. For the back-end we used the same approach, but with a different test framework called MochaJS, which is efficient and lightweight and dedicated for NodeJS-based Web applications.

Jasmine is a standalone framework, which is able to test JavaScript code without using the DOM. A test suite starts with a `describe` function that contains two parameters: a title, which is a string; and a function, which implements tests. Each test is nested into a function-specific unit (`it`). These functions are executable blocks that run the tests. Behaviours can be tested by applying the `expect` function, which compares actual values with expected values. Sometimes, test suites require specific setups before executing code. For instance, a setup can be the initialisation of specific variables, etc. Therefore, Jasmine provides `beforeEach` functions within test suites that are executed before each run of a specific unit (it). For instance, the following shows a test suite containing two tests that expect the variable `val` to be true and false respectively:

```
describe("A test suite", function () {
  var val;

  beforeEach(function () {
    val = true;
  });

  it("expect val to be true", function () {
    expect(val).toBe(true);
  });

  it("expect val to be false", function () {
    expect(val).toBe(false); // throws an exception
  });
});
```

For testing server-side code we applied Mocha, which is similar to Jasmine. Test suits have the same structure like Jasmine. However, there are some differences between Mocha and Jasmine. For instance, Mocha provides more flexibility when dealing with server-side code. The developer has the opportunity to include various assertion libraries to meet the demanded requirements. Popular assertion libraries are Chai[3], Expect[4] and Should[5]. Additionally, Mocha has to integrate a mocking library (e.g. Sinon[6]), which can simulate the server behaviours of real functions. In order to run Mocha including assertions and mocking, we need to install these modules by using the Node Package Manager (npm).

```
$ npm install mocha -g
$ npm install mocha chai sinon -save-dev
```

Mocha can be easily combined with Karma, a tool that is able to execute test code for different browsers. It is capable of watching files and run the whole test suites when changes occur. Thus, the developer does not need to trigger the test run after each change, which facilitates the software development.

2.3 CIPRTrainer Front-end

This section examines all AngularJS modules of the CIPRTrainer front-end (see Table 1). A module is a set of an AngularJS specific controller, service and view or directives. AngularJS directives are pre-defined markers on the DOM tree that are being compiled by AngularJS into an actual DOM-element. The following table describes briefly each module and its function for the CIPRTrainer.

Table 1: CIPRTrainer front-end module descriptions

Module	**Sidebar**
Description	The Sidebar module provides all necessary controls in order to run the training including start, stop, pause, or continue scenario.

Module	**What-if-Analysis (WIA)**
Description	The WIA module provides capabilities of exploring different courses of action and computing the consequence analysis.

Module	**Timeline**
Description	The Timeline module enables the user to examine scenario and simulation events. Also, it provides a cursor that is able to perform a rollback on a specific time.

[3] http://chaijs.com

[4] https://github.com/Automattic/expect.js

[5] https://shouldjs.github.io/

[6] http://sinonjs.org/

Module	Scenario
Description	The Scenario module loads the scenario and provides functions for exploring the training scenario.

Module	CIMap
Description	The CIMap module provides the base map on which the CI features and events are depicted.

Module	CILayer
Description	The CILayer module provides the option to remove or add CI features as GIS overlays.

Module	Training Logs
Description	The Training Log module is able to archive and list training logs.

Module	Home
Description	The Home module provides the landing page as well as a login mask.

Module	Authentication
Description	The Authentication module enables the user to log in to the system.

Module	Notification
Description	The Notification module pushes information of scenario events as well as general information like error or confirmation notifications.

Module	i18n
Description	The i18n module enables the end-user to switch between different languages. Also, it adapts the local tactical symbol based on the chosen language.

Module	Navigation
Description	The Navigation module enables the user to switch between different views and provides a login/logout mask.

2.3.1 Logging into the system

Each training session starts with an authentication. Using the application require user authentication: users have to launch the application by opening the browser and entering the domain name on which the CIPRTrainer web-server is listening. The landing page offers a navigation-bar on which the user is able to log into the system entering username and password (see Figure 3). Based on the user role, he or she may enter the training mode (Figure 4) or the trainer dashboard (Figure 8). Further user role can easily be added.

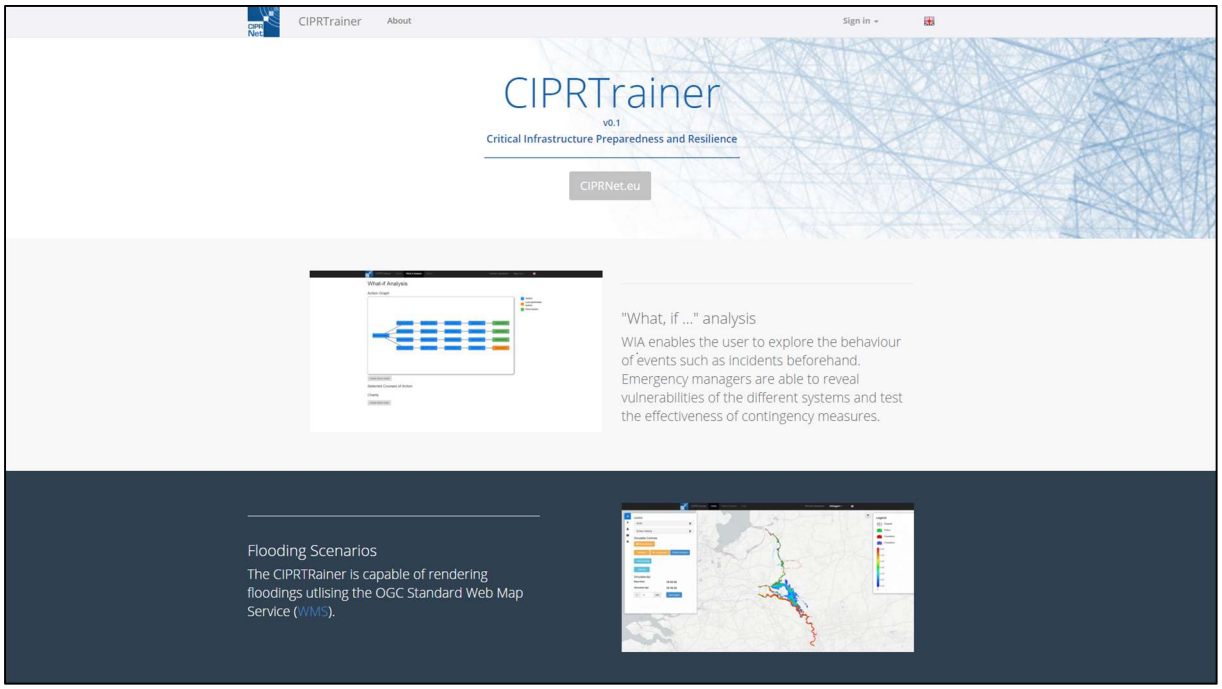


Figure 3: The landing page includes a navigation bar on the top and a main view depicting core features of the CIPRTrainer.

2.3.2 Trainee view, CIMap and CILayer

The CIPRTrainer is a typical GIS application where the end user can navigate through different layers or overlays. Two types of layers can be distinguished: Reference and Thematic layers [Tomasz2015]. Reference layers build the background of GIS and do not deliver any specific message but it gives an orientation to the user. Reference layers are also referred to as base layers or base maps. A typical base map is a satellite image as depicted in Figure 4. The more relevant layers are the thematic layers, of which there are several types: Choropleth Maps, Dot Density Maps, Proportional Symbol Maps and Isarithmic Maps [Tomasz2015]. “Proportional symbol maps use symbols of varying sizes that are proportional to the value or magnitude being shown” [Tomasz2015]. Dot density maps illustrate the distribution of behaviour patterns or conditions by marking these observations as dots on the map [Tomasz2015]. Examples are the usage of Twitter or the dissemination of a disease across the globe. Isarithmic maps are used to show continuous data like elevations, temperature or rainfall commonly illustrated on a contour map [Tomasz2015]. In the CIPRTrainer, the user is able to visualise or hide layers by clicking on the listed CI on the right sidebar and thus examine important CIs or resources (see Figure 7, right). A section for managing user account data such as changing password, email, and name is provided in the setting view (see Figure 7, left). Classic map controls are provided like zooming and panning for gaining more or less information on the panel. CIs or resources are represented as GIS markers containing CI- or resource-specific icons. Icons are carefully chosen in order to avoid misinterpretations. Crisis managers use specific tactical symbols that represent events and current states on the map. Moreover, for a crisis manager it is important to immediately know the operational status of CIs. Every GIS entity contains a small LED light on the upper right corner (see Figure 5). A green light illustrates an operational status without any interference ($x = 1$), whereas a red light symbolises a complete loss of operability or functionality ($x = 0$). The crisis manager can immediately interpret multiple statuses of CIs based on their LED light. Each GIS element contains additional CI-specific properties, which can be seen by clicking on the element. A pop-up appears containing the most important information about the entity. More information can be received

by clicking on the provided 'Read More' button on the bottom of the pop-up window. Since CIs can contain multiple elements, it is likely that the user loses track of the depicted CIs on the map. A well-known practice in GIS is clustering elements. It can be used to group CIs, which helps the user to maintain overview of the CIs. When the user zooms out of the map and passes a predefined zoom level, the system clusters CI elements into groups showing the number of the elements that are being clustered. Furthermore, when the user hovers over the cluster, the convex hull of the clustered elements is being showed. This helps the user to examine the distribution of the clustered GIS elements.

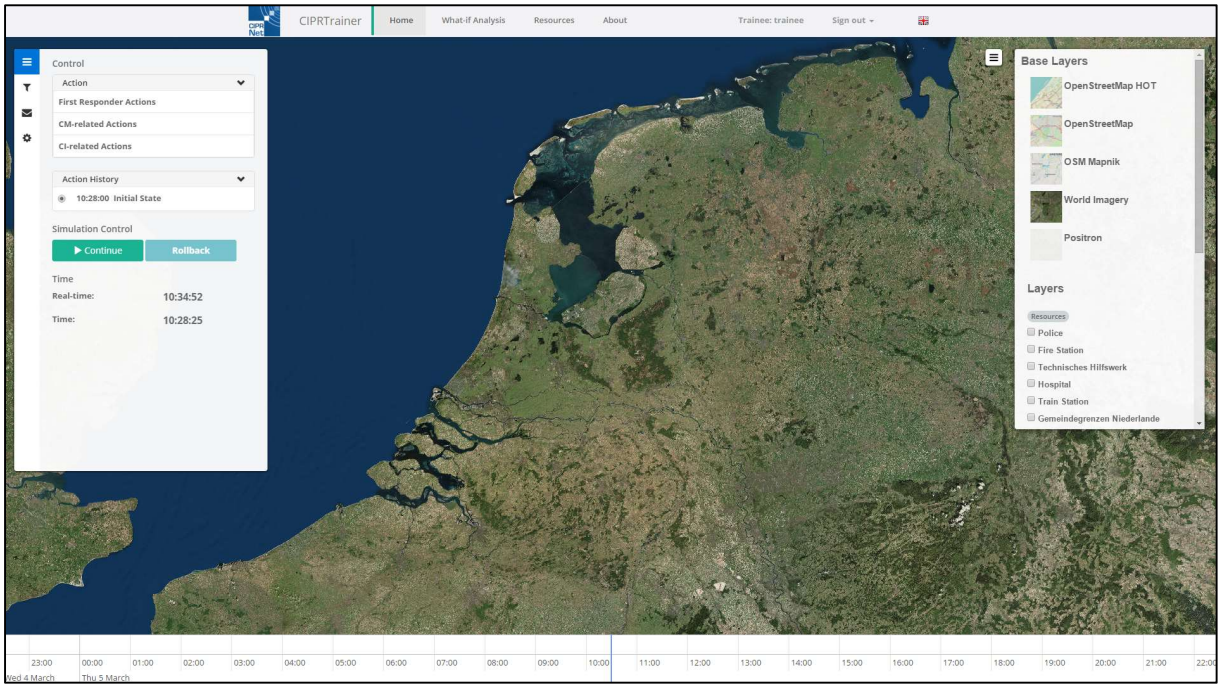


Figure 4: The trainee view consists of two side panels (left and right side) containing control functions and layer information, a main view showing the GIS map, a timeline on the bottom and the navigation bar on the top side.

2.3.3 Trainer Dashboard

The trainer is able to log into the dashboard (Figure 8) that monitors the evolution of the running training session including information about the trainee, the trainee's actions, scenario state, and CA results (see 7.2 Impact and Consequence Analysis Module (CAM)). The user also has the possibility to choose, start and stop scenario, and assign a participant to a training session. Moreover, the computed CA and training protocols are downloadable in CSV or JSON format.

2.3.4 Timeline

The timeline displays a set of different events in a chronological order (see Figure 9). These events can be pre-defined scenario events (accident, explosion, etc.) or events that are calculated by the federated simulators. Also, it can display different kind of actions, which can be performed by the crisis manager. Lastly, external sources of information are displayed on the timeline, which can origin from other agencies such as police or firefighters. A crucial advantage of displaying a chronological set of events is that the crisis manager can keep track of all kind of information sources. The user is able to focus on specific time intervals and thereby focus on important events and hide less important ones by dragging and zooming onto the

timeline. Therefore, the user is able to comprehend the complete scenario and thus make better decisions.

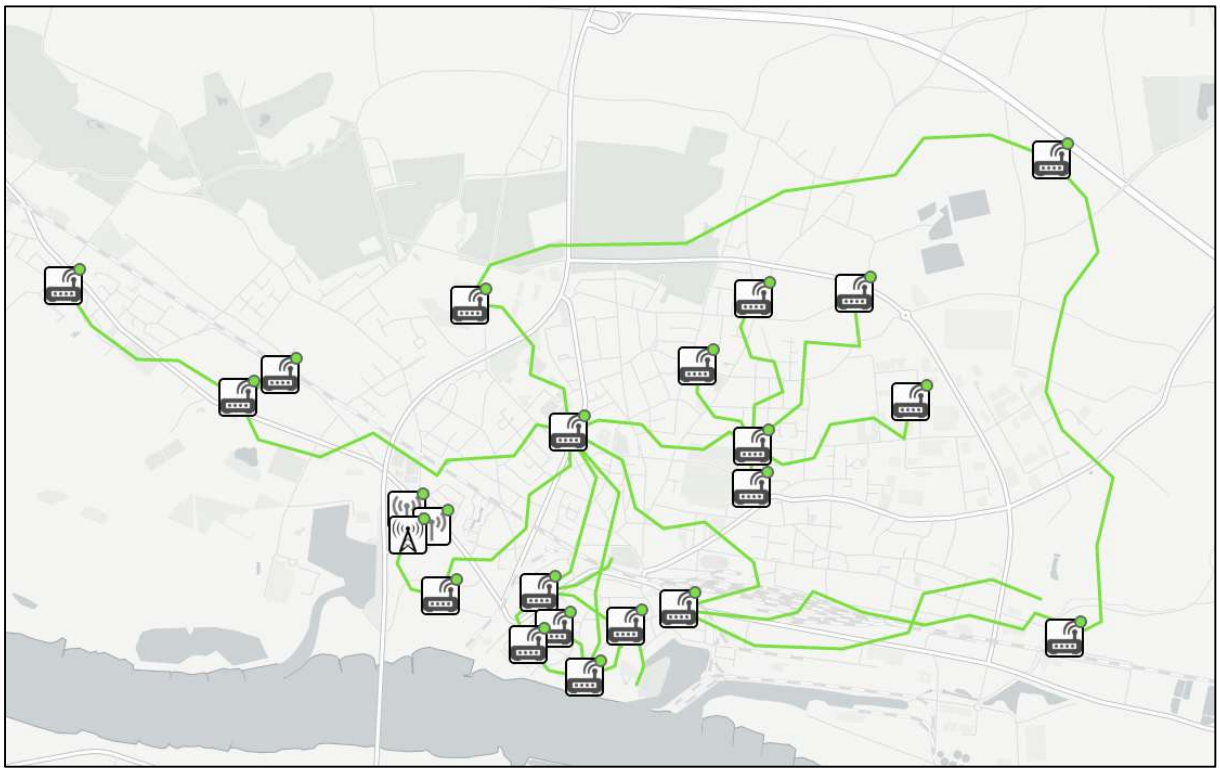


Figure 5: Telecommunication network that reflects the CI model in ns-3 in Emmerich city (see chapter 6). LED lights indicate the operational status of the CI.

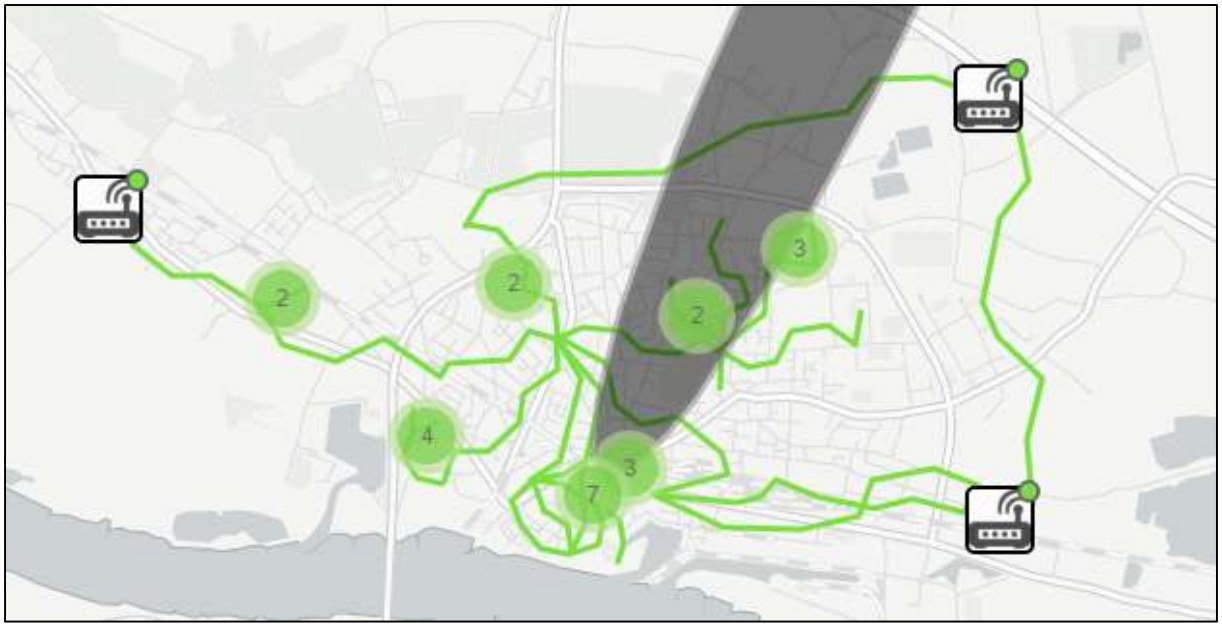


Figure 6: Clustered elements of the telecommunication network of ns-3.

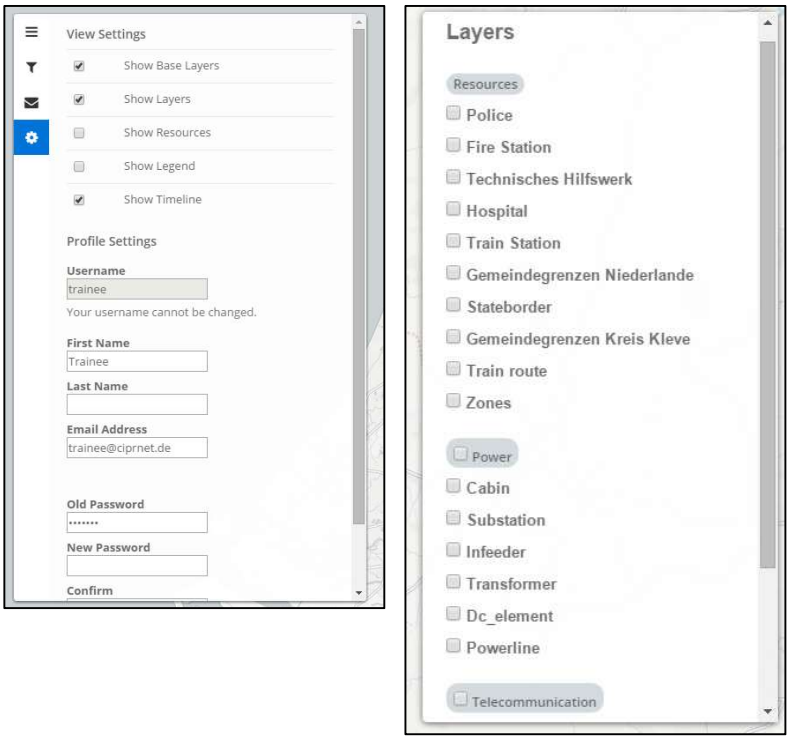


Figure 7: Managing the user account includes changing E-Mail, password, first- and last name. The username and password are used for launching the application. The username cannot be changed. The E-Mail must be provided for resolving cases like forgetting password or username. The password must have at least 8 characters containing at least one number, one uppercase letter, and one special character. In order to apply changes, the password has to be re-entered in an additional confirmation field. The CIPR-Trainer offers front-end customisation. He can show or hide several layers (base layer, CI layer, resources, legend and timeline). The left sidebar contain all sources of layer information including CI layers, base layers and so on.

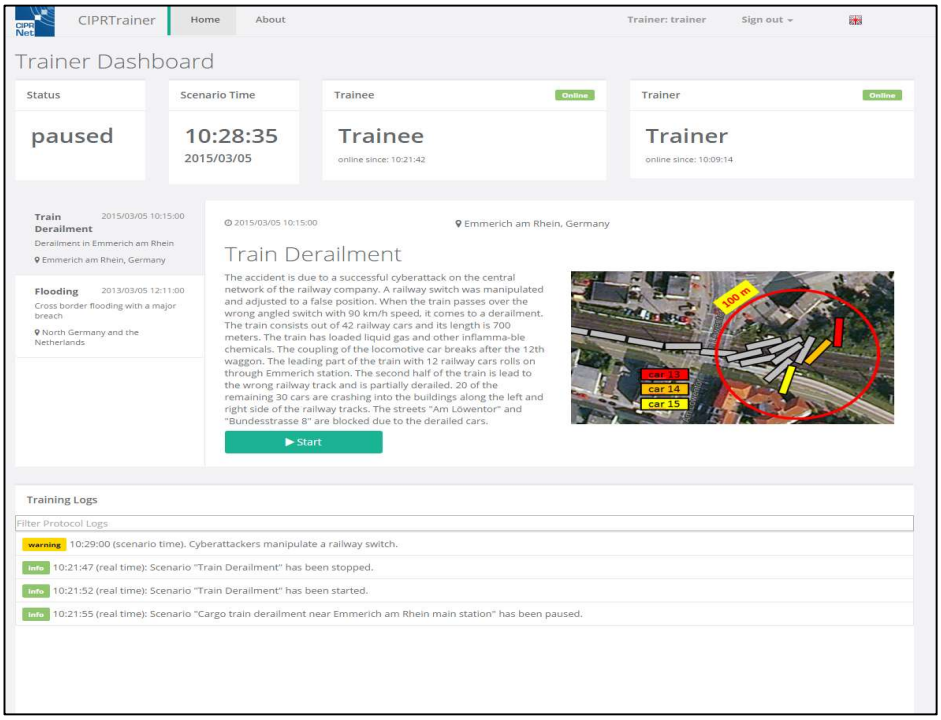


Figure 8: The trainer dashboard contains information about the running scenario, the statuses of the participants, a list of possible scenarios, of which the trainer can choose one to run; control functions for starting and stopping the scenario, and the possibility to download results of the CA and training logs.

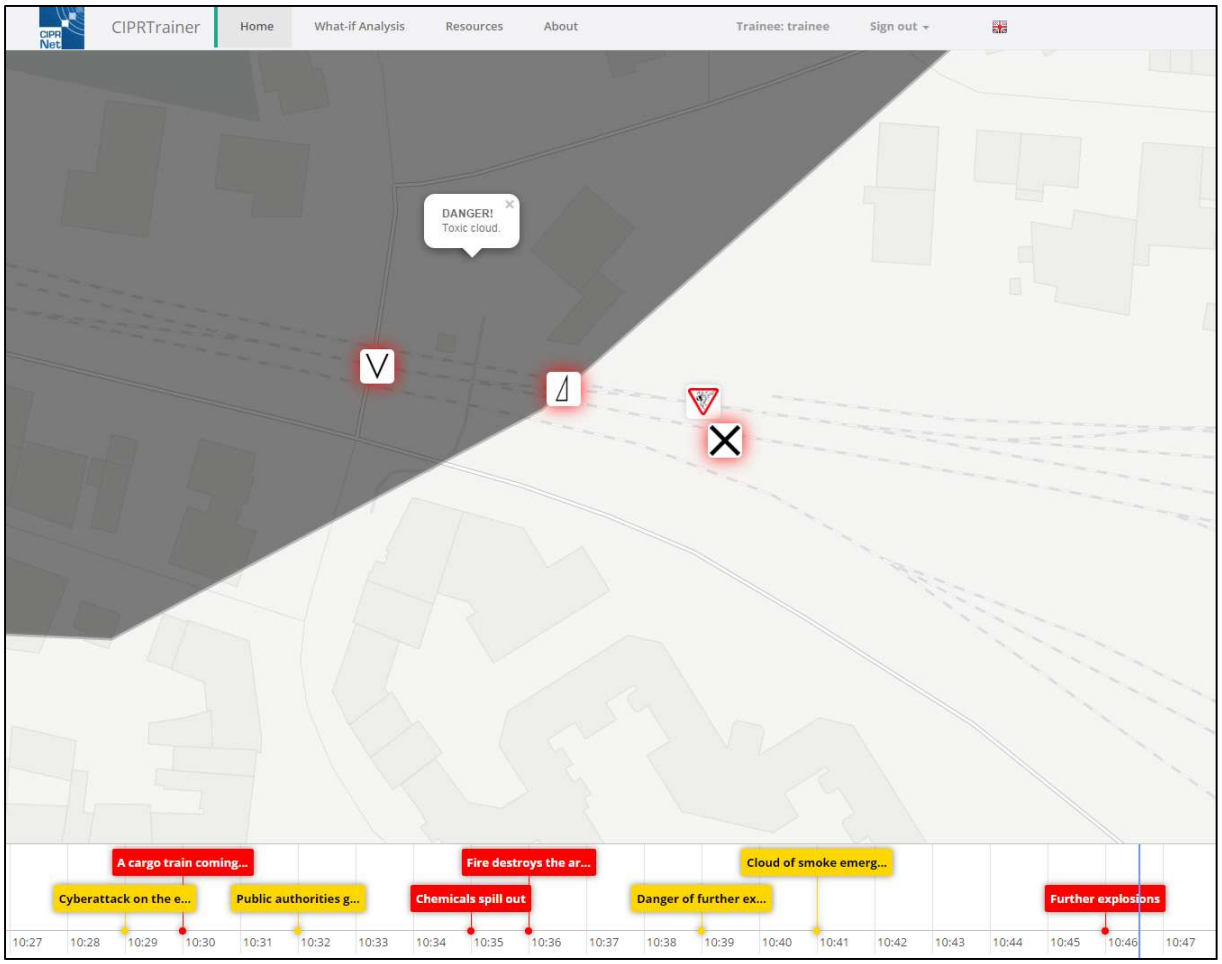


Figure 9: Specific events, such as explosion or fire, are depicted on the map using tactical symbols such that crisis managers can better understand the current situation of the scenario. The timeline provides a temporal context of the scenario. Moreover, it shows events (green, blue or red), which are listed in a chronological order. "Red events" are part of the pre-designed scenario and cannot be avoided (Explosion at the railway station in Emmerich). "Yellow events" are sources of information from third-party agencies (Telephone service and emergency call numbers are not available). "Blue events" (not in the picture) are actions that are performed by the trainee (Recruits forces to evacuate the area).

2.3.5 I18n support

The CIPRTrainer supports various languages (currently Dutch, German, and English). The user can choose a desired language by clicking on the listed flag on the navigation bar. As depicted in Figure 10 and Figure 11, the CIPRTrainer is able to depict tactical symbols of resources like police, fire-fighters or hospitals based on the end-user's localisation. Crisis managers from Netherlands utilise different tactical symbols than German crisis managers. The CIPRTrainer includes a set of tactical symbols for each country. Currently, German and Dutch tactical symbols are incorporated into the CIPRTrainer.

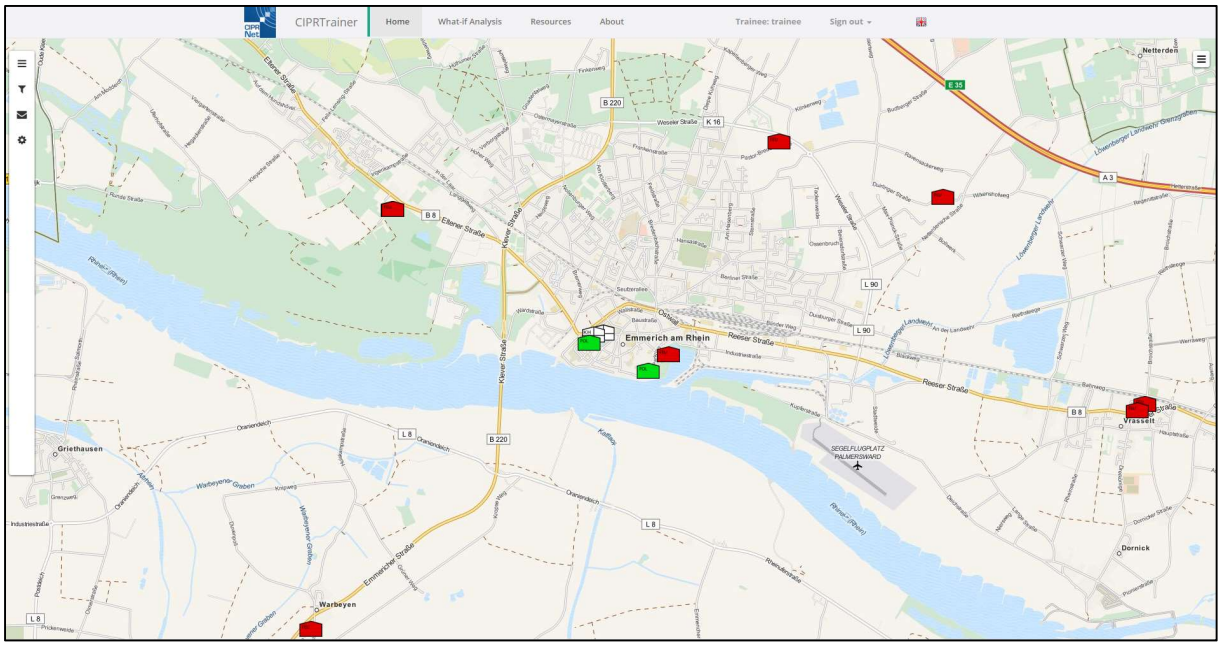


Figure 10: I18n for German crisis managers. This picture illustrates police stations (green objects), fire stations (red objects) and hospitals (white objects) depicting the recommended tactical symbols for German crisis manager.

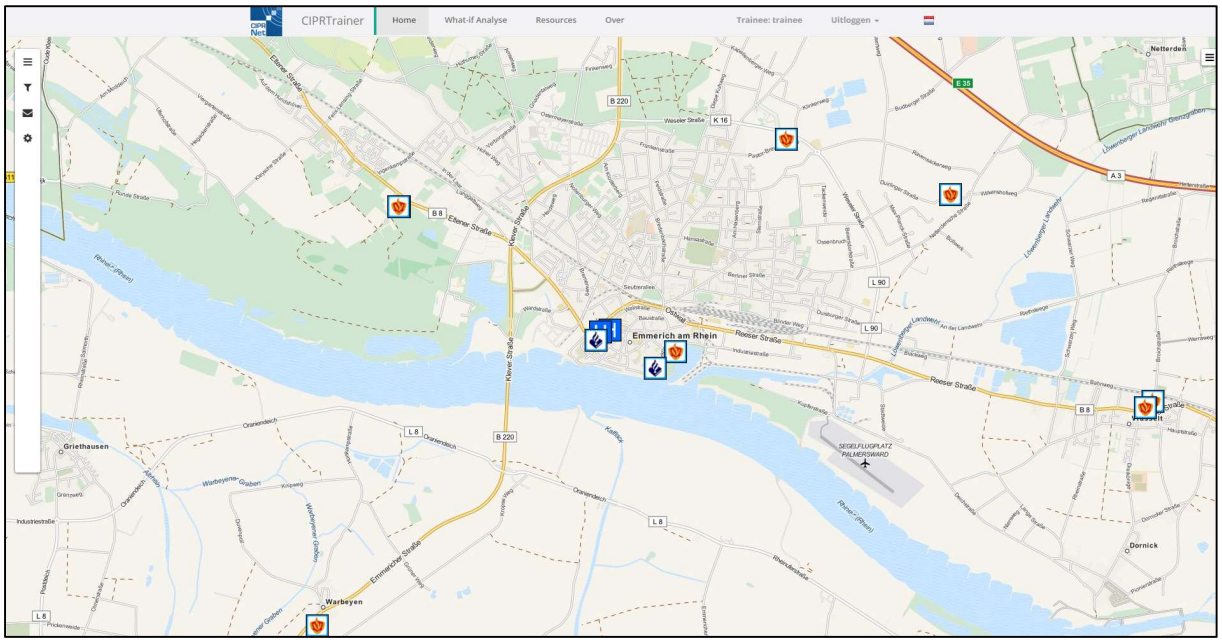


Figure 11: I18n for Dutch crisis managers. Police stations, fire stations and hospitals contain the recommended tactical symbols for Dutch crisis managers.

2.3.6 Performing an action

Actions influence the state of the critical infrastructure and the result of the consequence analysis. The trainee has two types of actions:

- First responder actions
- Crisis management actions

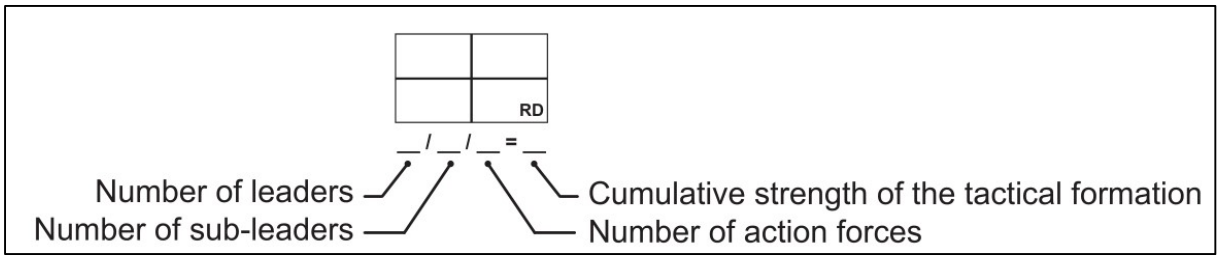


Figure 12: Tactical symbol notation for strength details of units.

The first type of action involves actual forces/resources, which are spread in the region of Emmerich. As depicted in Figure 13 (right) the CIPRTrainer provides a map that depicts the different resources and their capacities. The capacities are presented as triple (leader, sub-leader and forces, see Figure 12). The accumulation of these three values refers to the capacity strength of a specific unit. The user is able to send resources with a certain capacity to the crisis region. In addition to that, the trainee can select an activity that the forces perform in the crisis zone (e.g. fight the fire). The CIPRTrainer provides three types of forces: fire-fighters, police and the federal agency of technical relief. Each resource has a specific set of tasks or activities that can be performed. These activities influence the evolution of the scenario, CI models as well as the result of the consequence analysis. Specific rules are designed that interpret an activity and apply these on the socioeconomic data and the CI models.

The second type of actions is suited for crisis managers. A crisis manager is able to alarm public authorities and the general public as well as evacuate critical regions (see Figure 13, left). Each action influences the consequences in the scenario. For instance, when the crisis manager decides to not inform the general public, the number of injured people will rise. On the other side, if the crisis manager performs this action in an early state of the scenario, then less people will be injured, which mitigates the consequences. Each action triggers predefined rules based on the action-function, the time and the underlying socio-economic data (see section 5.3).

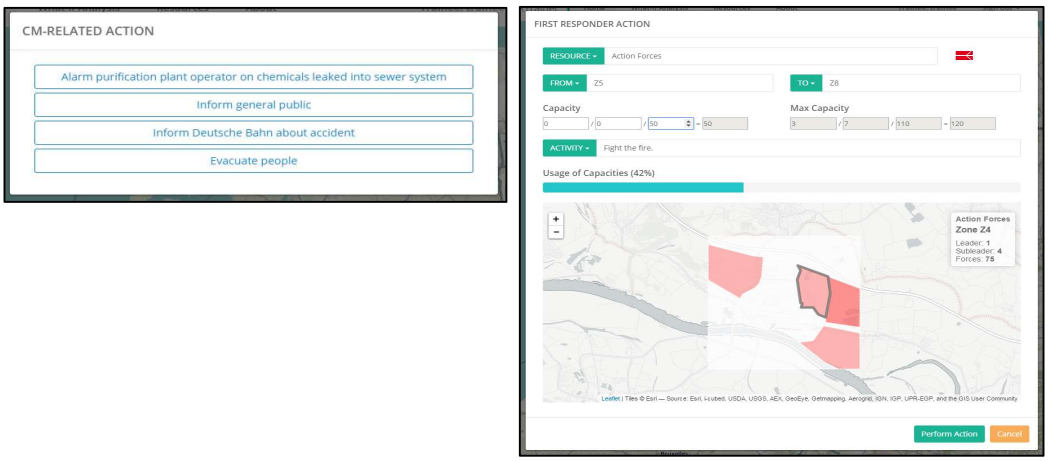


Figure 13: Interface for performing Crisis Management Actions (left) and First Responder Actions (right).

2.3.7 WIA action tree

The CIPRTrainer allows the end-user to select and compare consequences of different courses of action. Performed actions and their chronological order are visualized as a multidimensional tree (see Figure 14). Each node represents a performed action. Whenever the user jumps

back to a prior action x_t (perform a rollback), the CIPRTrainer automatically creates a new branch on action x_t, on which upcoming actions will be added. The result is a multidimensional tree that reflects the trainee's decisions throughout the entire training session. The leafs of the n-dimensional tree are the last performed actions of which each the user is able to acquire the consequence analysis of the entire action chain back to the initial state node of the tree. The result will be shown after a short while as a table and a GIS map on the web interface. The table incorporates quantified damages and other consequences such as reconstruction cost for business and residential building, infrastructure elements as well as value of lost loads for households (see section 7.2.2.7.1), emergency forces cost, number of injured and dead humans. In addition, a map depicts thematic layers that give an overview of the *distribution* of the consequences meaning that there is a spatial context attached to the consequences (see section 7.2.2). The main capability of GIS is the accumulation of various sets of geographical layers containing data that can be shown onto one display. For instance, Figure 15 shows a choropleth map that reflects the distribution of injured people in the Emmerich area. Choropleth maps are used for classifying homogeneous and areal entities using specific colours [Tomasz2015]. For quantitative classifications like for this instance, colours should be used in short hue ranges with different saturation levels that are proportional to the classified entities [Tomasz2015]. Usually the entity with highest quantitative value has the darkest colour. The CIPRTrainer acquires CA results as a GeoJSON, which incorporates the spatial context data.

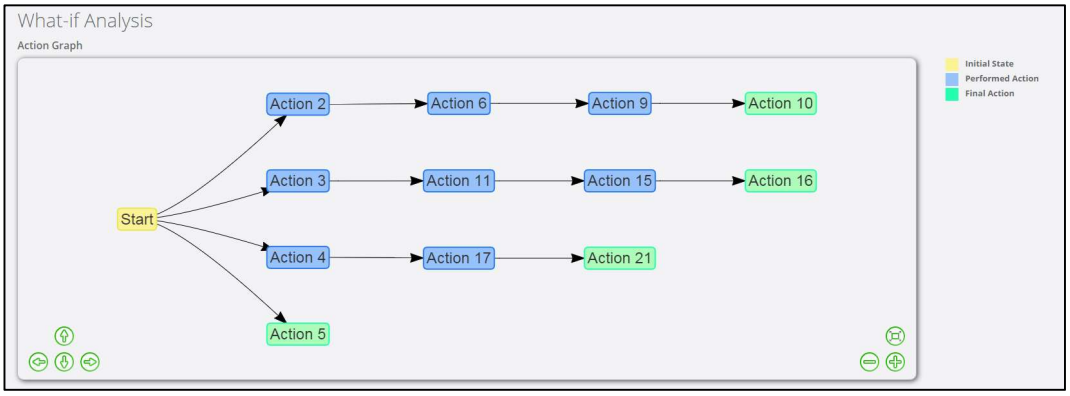


Figure 14: Each node of the n-dimensional tree refers to an action. The rollback capability creates an additional branch, on which following actions can be added. Final actions are marked as green nodes. The root node corresponds to the initial state of the scenario. In this example, the end-user performed four rollbacks (*Action 10* → *Start*, *Action 16* → *Start*, and so on).

2.4 CIPRTrainer Back-end

NodeJS is a JavaScript runtime environment for implementing server-side applications. The platform runs on Windows and UNIX-systems like OS X or Linux. The CIPRTrainer back-end is mainly a node application server. User sends requests to the web server (Nginx), which is then redirected to a private IP address on which the node application server listens. Figure 16 represents the workflow of the user-server communication. Typically, a node application consists of various configuration files (server and database configuration, etc.), a front-end implementation, set of views (HTTP-files or EJS templates) and routes (RESTful endpoints), and a logic tier. Figure 17 depicts the actual file structure of the CIPRTrainer implementation. The main configuration of the server is located in `app.js`, which incorporates HTTP web-server definitions and references to the RESTful endpoints. Any other sort of configurations that do not deal with Node, such as database connections and WFS configurations, are located in `config.js`. The front-end implementation lies in the public folder. The content is acces-

sible to the end-user. The server implementation including route end-points, views, and server specific services are located in the server folder.

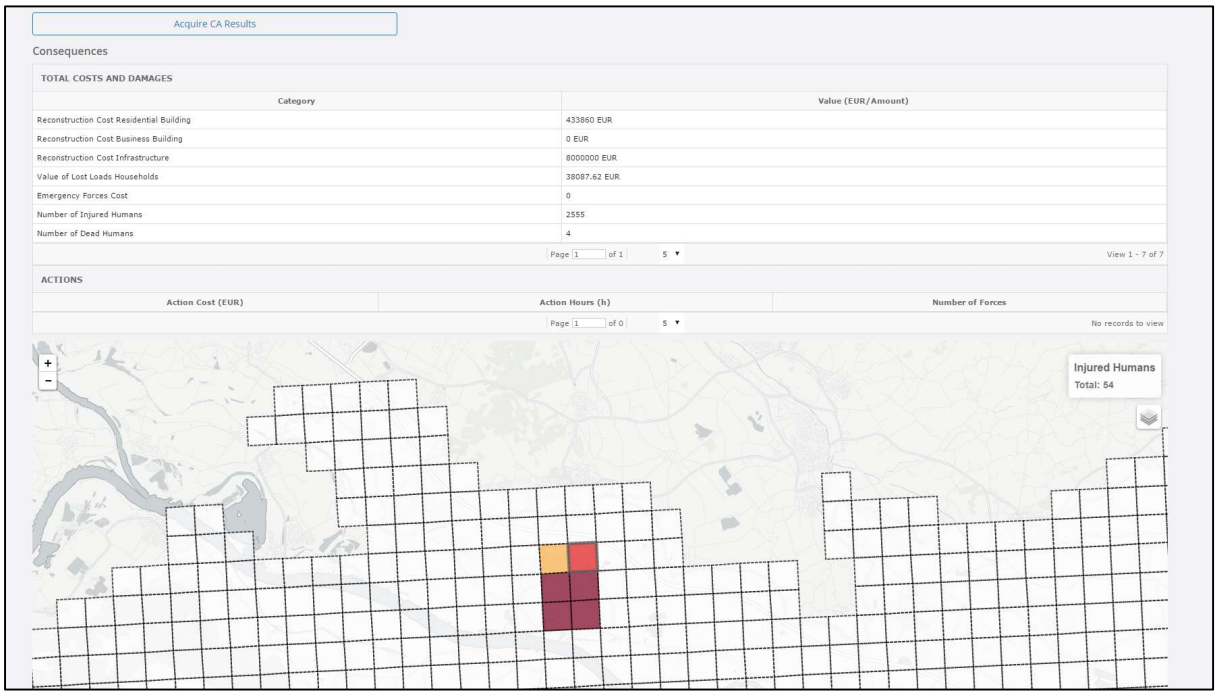


Figure 15: CA result table and visualisation.

Bower and Gulp

To facilitate server-side implementation, two helpful tools were included in our development process: Bower and Gulp. Bower manages JavaScript and CSS files for the front-end implementation. It installs packages and dependencies in a desired version. All dependencies are annotated in `bower.json`. The developer can install a component, for instance AngularJS, by running the following instruction:

```
$ bower install <package_name> --save
```

`bower.json` will be automatically extended by this package. If the package name is not provided in the Bower address repository, the user can enter the URL of the repository manually. All dependencies that are annotated in `bower.json` can be installed by the following instruction:

```
$ bower install
```

Gulp is a powerful tool to automate build processes. The user defines tasks that can be run by triggering gulp. It also is capable of watching files such that triggers a task whenever changes happen. Gulp handles any kind of document manipulation (minification, concatenation, etc.). Moreover, it is able to compile TypeScript into JavaScript. All defined tasks are located in `gulpfile.js`. For instance, the following task compiles any file with the ending ".ts" that is located in the server folder:

```
gulp.task("compile_ts_server", function () {
  return gulp.src('./server/**/*.ts')
```

```
  .pipe(tsc({       // tsc TypeScript compiler
    target: 'ES5'   // set JavaScript mode to ES5
  }))
  .pipe(gulp.dest('server'));
});
```

Figure 16: Server-user communication workflow with NGINX reverse proxy [NODE].

```
ciprtrainer/
     |_app.js # ExpressJS
     |_config.js # Database, WFS, ets. configuration file
     |_gulpfile.js # GulpJS
     |_bower.json # BowerJS
     |_package.json # Node dependecies
     |_Dockerfile # Docker configuration file
     |_install {.sh, .bat} # Install scripts
     |_start {.sh, .bat} # Run scripts
     |_node_modules/ # Automatically loaded dependencies by npm
     |_public/ # CIPRTrainer front-end
     |_test/ # Test specifictions
     |_server/ # NodeJS server components
          |_models/
          |_services/
          |_routes/
          |_views/
```

Figure 17: CIPRTrainer implementation file structure.

2.5 Scenario Executor

Scenarios allow crisis managers to outline a sequence of events and provide the basis for the CA, thus evaluating susceptibilities of CIs by revealing dependencies, interdependencies and

cascading effects. Part of the scenario is the scenario executor, which incorporates the storyline, a set of unavoidable events. Scenario executor controls the heartbeat of the whole system. It maps the simulation time and real wall time. For instance, to accelerate the simulation, the scale can be 60:1, i.e. 60 simulation seconds should be done within one real time second. Under this setting, two situations can happen:

- The system is fast enough and the actual execution time is less than one real time second. Therefore some kind of `sleep` mechanisms will be introduced before the simulation for the next 60 simulation seconds is started.
- The system is not fast enough to finish the simulation within the given time frame. That means, it is not possible to simulate 60 seconds within one real time second. The scaling factor will be modified based on the best system performance, e.g. 10:1 – just simulate 10 simulation seconds instead of 60.

The trainer can initialise (load the storyline of the scenario) and start or stop the scenario executor. Each event in the storyline is annotated with a timestamp t_i. Once the simulation time t_s passes t_i, the scenario executor notifies CEP-Engine and the CIPRTrainer by sending a HTTP push-request containing event-specific data (see Figure 18). This way, the trainee can see events on the map including GPS coordinates, event-specific information and a tactical symbol describing the event. The trainee can pause and continue the training.

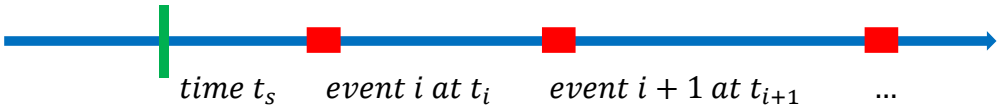


Figure 18: Once the simulation time t_s passes t_i, CEP-Engine and CIPRTrainer will be notified by sending a HTTP push-request containing event-specific data.

2.6 Mapping Services

The Open Geospatial Consortium (**OCG**) is an international organisation that provides spatial data and service standards for various GIS applications [OCG]. One of these standards is the Web Map Service Interface Standard (**WMS**) that "provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases" [WMS]. The WMS standard is used for acquiring rasterised spatial data such as base layer maps. Another important standard is the Web Feature Service Interface Standard (**WFS**) that provides an interface specification for requesting spatial features [WFS]. It provides vectorised data in various formats such as ESRI Shapefile, GeoJSON, GML, CSV, etc. *MapServer* is an open source platform of providing spatial features for GIS applications that incorporates both OGC standards WMS and WFS (see Figure 19). CIPRTrainer uses *MapServer* to receive vectorised features like the CI models of the simulators and resources (police, firefighters, hospitals, etc.). We use the WMS standard to show flooding on the base layer.

The Geospatial Data Abstraction Library (GDAL) is a library for manipulating raster and vector spatial features, which is developed and maintained by non-profit organisation Open Source Geospatial Foundation (OSGeo). One part of GDAL is the OGR library, which focuses on vector-based features. "This program can be used to convert simple features between file formats performing various operations during the process such as spatial or attribute selections, reducing the set of attributes, setting the output coordinate system or even reprojecting the features during translation" [GDAL]. For instance, the following command converts a GeoJSON feature into ESRI Shapefile format:

```
$ ogr2ogr -f "ESRI Shapefile" feature.shp feature.geojson
  OGRGeoJSON
```

In addition, the program can incorporate spatial features into a relational database and map all feature attributes. For instance, the following instruction creates first a database scheme `ciprtrainer.feature` and then inserts the spatial information into the PostgreSQL database.

```
$ ogr2ogr -f "PostgreSQL" PG:"host='localhost' port='5432'
 dbname='ciprnet' user='foo' password='bar'" "feature.geojson"
 ciprtrainer.feature
```

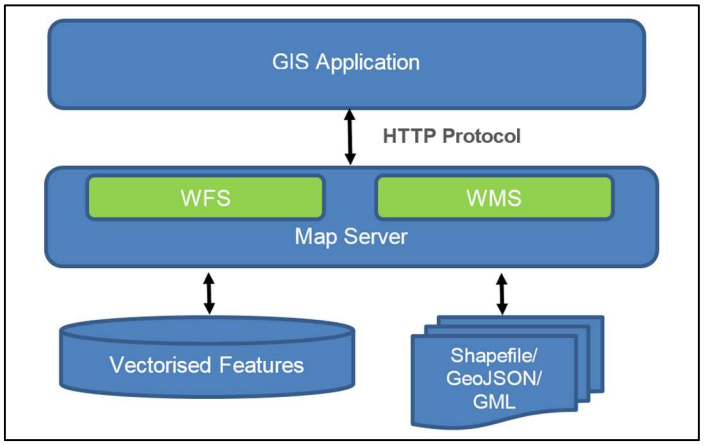


Figure 19: *MapServer* implements the OGC Standards WFS and WMS that provide rasterised and vectorised spatial data for rich GIS applications. Spatial data can be stored in relational databases (e.g. PostgreSQL + Postgis extension) or flat files that can have various formats (e.g. GeoJSON, ESRI Shapefile).

2.7 Training mode and Operation mode

The main focus of CIPRTrainer is for training. However it can also be used in the operation mode in hot phase, depending on the threat evolvement, e.g. flooding has a much slower development than an explosion causing fire emergency. Anyway, these two modes are completely different with different target users.

In general, the major differences between training and operation modes are:

- **Training mode**: the events, i.e. what could happen in training are pre-defined or simulated. The events are injected into the system in a deterministic way by the scenario executor. The simulation results based on the CI models and dependency rules are in most cases also deterministic. The actions in training are executed in a simulated environment. The effects are calculated based on specific-domain simulators like SINCAL for the load flow calculation in electrical networks. More details about domain-specific CI simulator see Section 6.3.
- **Operation mode**: during an emergency hot phase events are from external physical world and in most cases are not really predictable. In real world hot phases actions are executed with real entities like evacuation and no “rollback” is possible, only compensation actions can be done.
- In real world hot phases actions are executed with real entities like evacuation and no “rollback” is possible, only compensation actions can be done.

To summarise, the relation between these two modes for crisis management is as follows: an operational mode may contain multiple training parts for which What-if Analysis can be performed if time permits (for instance during flooding, the whole crisis will have a duration of

several days and if the simulation systems is fast enough, different courses of actions can be tried). If the decision makers or crisis managers are willing to (e.g. the inputs needed for perform such analysis is quite simple to provide and the added value is high).

3 Integration with RESTful Web Services

CIPRTrainer is a complex system with multiple components integrated together. At runtime, all the components must be able to communicate with each other in a pre-defined way. To provide the highest level of flexibility, we decided to use Web-based system integration approach, RESTful Web Service in particular.

3.1 Design Rationale

One of the design rationales of CIPRNet is to keep it simple and flexible. To achieve this we adopted the lightweight RESTful web service as the basis for system interaction and integration. There are lots of discussion [REST08] about the benefits and drawbacks of various system integration paradigms (see Figure 20) including

- Shared databases – different sub-systems write and read the information need to be changed through a central and shared database in a transactional way. Transactions are however very time and resource intensive, i.e. using shared databases to integrate systems is rather inefficient.
- Remote procedural (RPC) calls with heavy and lightweight web services (like SOAP and RESTful) – remote procedurals are traditional ways to access different process on the same machine and further developed to access processes on other machines running in the same local network. Originally it is not designed for machines interconnected through Internet. Therefore to overcome these drawbacks, SOAP standard is proposed to enable an efficient and flexible way of communication through Internet.
- Shared file systems – this fashion of system integration is very similar to the shared databases. Instead of using transactional databases, the rather ad-hoc and flexible file systems are used – either local or remote. The communication complexity is then pushed down to the file system drivers, which should be able handle concurrency and asynchronous access from different sub systems.
- Message Bus– message buses are quite new, which has a root on enterprise information integration. Most of the sub-systems in an enterprise environment need to communicate with each other in a way that the content to be exchanged can be modelled as simple messages. Therefore dedicated message buses are then developed to fit this kind of needs. It provides a reliable and efficient way to integrate local-area enterprise systems. However, for web-based system integration, it is not always a good candidate.

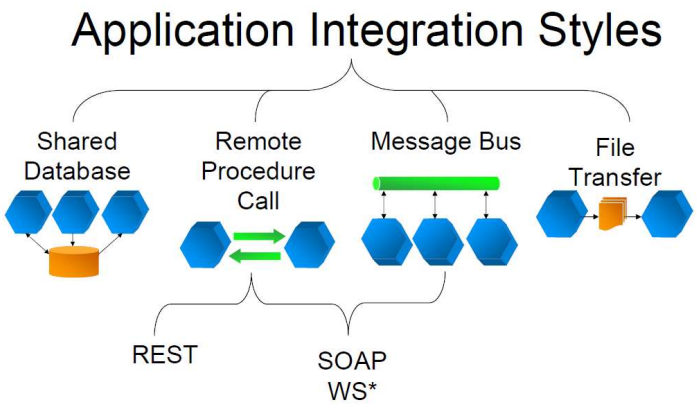


Figure 20: Four typical application integration styles [REST08]

These are basically different ways to support Inter-Process Communication (IPC) in Computer Science. For practical purposes however, there are no one-for-all solutions, where a paradigm fits all use cases. To this end, in the case of CIPRNet, we decided to use the more lightweight RESTful web services to integrate the individual software components for its simplicity, outstanding and almost universal tool support, and well-integration into the HTTP standards. More detailed discussion about the benefits and limitations, see [REST08]

All of the RESTful web services are developed in two phases:

- At the beginning all web services are implemented with mock-up, i.e. hand-crafted static data will be delivered if the service is consumed.
- During the implementation of CIPRNet and its sub systems, part of the mock-ups can be replaced by the real implementations in an iterative way, see Figure 21.

This two-phase design provides a solid basis for the iterative system integration. Moreover, automatic integration testing can be applied to guarantee the correctness of the implemented services with less human efforts. However, special attention needs to be paid for employing Test-Driven Development (TDD) [TDD03] for unit-testing of internal logics of individual tools – it can results in a much larger set of unit testing codes and with the available resources in CIPRNet this could bring more problem than its benefits.

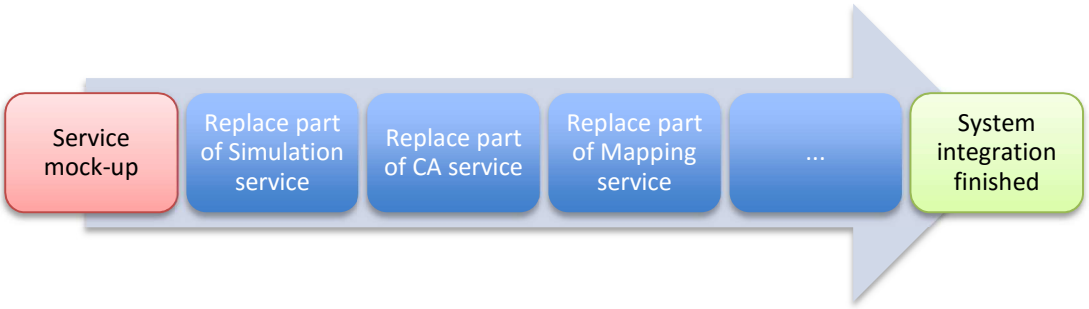


Figure 21: Partial service implementation and iterative integration

For CIPRNet the design principle based on reverse proxy is adopted. There are several major benefits of this approach:

- A consistent and central access of CIPRTrainer to the end-users. They just need to know the URL of the CIPRTrainer and nothing more. Even the documentation and user manuals can be accessed by visiting URL.
- A unified RESTful API specification sitting on top of CIPRNet. Individual internal tools do not need to access other sub systems via a bunch of different URLs by messing up various hostnames and paths. A hierarchical structure of the RESTful services can be provided – all have one and only one access point CIPRNet.
- Centralised logging and monitors
- Hot plug and play
- Scalability and performance considerations, load balancing
- HTTPS termination, ease application development

Based on the experience of previous industry and research projects, some potential issues with reverse proxy based design may arise during the integration work – depending on the scale and complexity of the system to be integrated. Most of these potential issues are technical ones covering various aspects of web development, from security issues to session control, from real-time communication over HTTP to load balancing, etc. A subset of these issues, which will most probably occur during the technical integration phase, are summarised as follows:

- **Security** – crisis management systems should run in a secured tunnel. Point-to-point security measures should be applied to avoid information sniff by third parties within the communication channels. The most established way till now to achieve this level of security is to use HTTPS instead of the plain old HTTP. One issue with HTTPS with reverse proxy is the mixed content vulnerability, i.e. contents turned by the HTTPS protocol contain URLs that use HTTP. For instance a website may be served with HTTPS; the JavaScript libraries it uses however are serviced from a third website, which uses instead HTTP. By default, most modern browsers will block the request and throw some warnings in the browser console. Therefore, the upstream servers need to provide URLs that all use HTTPS version for retrieving resources.
- **Session control** – collaborative system needs to manage user sessions to distinguish different users and their data. With the introduction of the reverse proxy server, session management can be more complicated than without them. For instance, one of the typical pitfalls in session management is to use the path attribute in cookies. With the introduction of a proxy server, the paths of exposed services are normally changed for a consistent view of the whole application. During the development phase the path of a cookie is normally fixed for development testing. However the developers are not aware of the production environment. Therefore it can happen that one application works under development and does not work in production behind of a reverse proxy server. The reason is in most cases the cookies specified with one path are not consistent with the path on the reverse proxy server – they will not be sent back to the reverse proxy server to enable the correct session management in the web application back-ends.
- **Real-time communication** – HTTP and HTTPS are not designed for real-time communication on the Web. They are basically for plain simple request and response over Internet. As the Web itself develops, more applications are emerging and some of them need the capability of real-time duplex communication over Internet. For that purpose, new standards are proposed. One of the most promising and widely used is the WebSocket [WebSocket], which is also included in the HTML5 standard for developing next generation Web applications. WebSocket is based on duplex communication and sits directly on top of TCP, which is the same as HTTP/HTTPS. With the introduction of a reverse proxy server however, problems may arise since most reverse proxy server do not speak WebSocket protocol directly – they need to further communicate with the upstream servers. Therefore it could happen that one HTML5 application with WebSocket enabled works perfect if it is deployed directly to the Internet; however behind a reverse proxy server, it simply does not work anymore.
- **Load balancing** – Modern web applications need to scale, i.e. independent of the users; the system should react timely with a predefined threshold of delay. Using reverse proxy server is the most standard and effective solution to achieve this. Redirecting user requests to a list of CIPRNet upstream servers can be solved with less efforts; however if other issues like session management, monitoring, etc. are also need to be considered, the configuration can be tricky to achieve the desired performance and security criteria.

Special attentions are paid to the security issue of both the development documentation and service endpoints. Access to these URLs needs user authentication, which is not based on session and cookies. Instead, for each HTTP request – including GET, POST, PUT and DELETE, a user and token combination will be checked to ensure that only authorised access is allowed.

3.2 Using HTTP vocabulary and URL

Communication vocabulary is the essential part of interface specification. Traditional methods to define the vocabulary are using a set of functions with understandable names like `sendEvent()` or `getEvent()` to denote the semantics.

In RESTful system design, it can be substantially simplified by using the standard HTTP vocabulary and the concept of "resources". In HTTP protocol, four basic operations are defined[7]:

- GET – it corresponds to the "read" or "get" operation to retrieve the states or attribute values of a certain resource instance
- POST – it deals with the "create" operation to create a non-existing resource instance
- PUT – it is about "updating" attributes of certain resource instances like name, external id, etc.
- DELETE – it is used to remove a certain resource instance, normally based on the unique identifier of that instance

In this way, the normal operations can be simplified by just using these four.

Resources are entities that exist in a system. It can be the model of a person, a book, a critical infrastructure, etc. Virtually it can be anything that the system deals with. URL – the Unified Resource Locator, or the more internationalised URI – the Unified Resource Identifier can be used to represent the resources. Together with the four operations in standard HTTP, it provides a unified way to manage resources.

One example, say "to send an event" from system `A` to system `B`. In the traditional system design, a function or service in non-RESTful style can be defined. It looks like `sendEvent()`. On the opposite side, to get an event from system B can be achieved by using the function `getEvent()`. In RESTful design, first of all a resource `event` can be defined by exposing it as a URL http://myhost/event. In this case, the pre-defined HTTP operations GET and POST can be used to retrieve and send an event from/to the destination system. More concrete, for retrieving event issue the following HTTP request:

```
GET /event HTTP/1.1
HOST: myhost
```

And for sending event the following HTTP request can be used with the appropriate HTTP body information containing the real content of the event.

```
POST /event HTTP/1.1
HOST: myhost
```

In this way, the interface design of the system is much simpler; in principle different system developers only need to know the resource URL since the HTTP vocabulary is standard. This is especially useful for large systems with thousands of operations and resource interacting with each other. For CIPRTrainer, the URLs defined are listed in Table 2.

[7] Other operations like HEAD are also defined, but not so relevant for specifying RESTful Web Services.

Table 2: List of application server and their globally organised ports behind reverse proxy server

ID	App Name	Port	URL
1	MapServer CGI	8000	`/map`
2	Consequence Analysis – NodeJS	3000	`/ca`
3	Flood Simulator – NodeJS	3004	`/fedsim/floodsim`
4	Continuous Integration Server – Jenkins	8080	`/ci`

5	CIPRTrainer Backend – NodeJS
	• Socket.IO
	• What-if Analysis
	• Authentication
	• Resources like WFS
		3005	`/wia`
			`/scenario`
			`/authenticate`
			`/app`
			`/assets`
			`/bower_components`
			`/ciprtrainer`
			`/ctio`
			`/wfs`

6	Complex Event Processing	3007	`/cep`
7	SINCAL Simulator Adaptor	3008	`/fedsim/sincal`
8	Real-time logging facility	3003	`/log`
9	Bugzilla Server	8999	`/bug`
10	OpenTrack Simulator Adaptor	3009	`/fedsim/opentrack`
11	ns-3 Simulator Adaptor	3010	`/fedsim/ns3`
12	Database Adaptor	3011	`/db`

4 SyMo-based Scenario Development

The SyMo (System Modeller) software is developed by Fraunhofer IAIS since 2008. It allows creating detailed system models that consider multiple views like requirements, structure, behaviour, input- and output-data. The software is further able to do analysis and simulation on these models with the objective to find processes that may be optimized. With this information it is possible to create and to analyse complex systems like an organization for example. At Fraunhofer IAIS the software has various purposes and is used in economy related projects as well as in research projects.

For the CIPRNet project SyMo is used as a scenario editor. Main advantage in using SyMo for the creation of the scenarios is that all necessary elements, sequence control, syntax checks and even semantic examination are already implemented and incorporated inside a graphical user interface (GUI).

4.1 SDL – the Scenario Description Language

For the CIPRNet scenarios a template using SDL (Scenario Description Language) was defined that includes all necessary information that is needed by the simulators, complex event processor, scenario executer and the CIPRTrainer frontend.

The CIPRNet SDL file format is formatted in JSON and is structured as follows:

```
Scenario Description Language (SDL)
    |_meta data
        |_name of the scenario
        |_description of the scenario
        |_begin and ending time of the scenario
        |_creator credentials
    |_ simulator config
        |_configuration file for each simulator: Sincal, NS3, OpenTrack
    |_complex event processor config
        |_configuration file for the CEP
    |_critical infrastructure models
        |_domain - "power","telco","railway"
        |_simulator - "sincal","ns-3","opentrack"
    |_dependency rules
        |_functional dependencies between different CI as CEP Rule
    |_decision maker
        |_name
        |_role
    |_resources
        |_type
        |_location
        |_capacity
    |_threats
```

```
        |_domain
        |_simulator
        |_beginning time
        |_location
|_mitigation actions
        |_timepoint
        |_action as text
|_storyline events
        |_time
        |_type (CEP)
        |_payload
              |_id
              |_type (CIPRTrainer)
              |_date
              |_event title
              |_incident description
              |_icon
              |_visible (on map)
              |_location
|_context information
        |_domain
        |_name
        |_description
|_spatial data
        |_background maps
        |_layers
```

It follows a brief description of the different fields of the proposed template:

- *Meta data* consists out of basic information about the scenario. It includes a scenario name, a short description, timeframe and the credentials of the creator;
- *Simulator config* and complex *event processor config* elements include the configuration files for each simulator and the Complex Event Processor (CEP);
- *Critical infrastructure models* define a mapping between the critical infrastructure and their appropriate simulators. In the CIPRNet project the power network is handled by the Sincal simulator, NS-3 simulator does the telecommunication simulation and Opentrack is used for the simulation of the railway infrastructure;
- *Dependency rules* are used to model the interactions between the simulators and threats. They are processed by the complex event processor and have a SQL like syntax;
- *Decision maker* represents the possible actors inside the CIPRTrainer GUI. At the moment there is one Trainer and one Trainee designated for a training session.
- *Resources* are used to describe the major elements inside a scenario that are manipulated by the simulators, threats and the trainee during a training session. They consist

out of a type, location and capacity. For this the scenario map is subdivided into 15 different zones. This allows allocating different groups of action forces or residents into one location without a detailed geo-mapping;
- *Threats* describe external events like cyber-attacks or floods. These are provided as static events that are executed at a specified time-point and location. A threat will trigger a dependency rule so further events e.g. destruction of power lines are executed;
- *Mitigation actions* define possible actions that the trainee may perform during a training session. E.g. moving action forces from one location to another, evacuate residents or inform population about imminent dangers;
- *Storyline* events consists out of a discrete timestamp and a type which is interpreted by the complex event processor and a payload that contains all the semantics of the storyline event like date, geographic location and event description;
- *Context information* allows the definition of conditions from the outside world like weather conditions, traffic situations, pollution, etc. in a crisis region;
- *Spatial data* defines different map layers that may be selected inside the CIPRTrainer GUI.

For the creation and validation a template and schema validator was created. The template consists out of the defined values as JSON string with example values set. The schema validator checks the validity of a complete SDL JSON file. It checks the overall structure, the existence of mandatory values and if the correct datatype is set. In the following a part of the SDL JSON schema validator, that validates the first configuration object ("meta" –information) is shown.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "meta": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "name": {
          "type": "string",
          "minLength": 1
        },
        "url": {
          "type": "string",
          "minLength": 1
        },
        "description": {
          "type": "string",
          "minLength": 1
        },
        "creator name": {
          "type": "string",
          "minLength": 1
        },
        "creator email": {
          "type": "string",
          "minLength": 1
```

```
        },
        "creator organisation": {
          "type": "string",
          "minLength": 1
        },
        "begin": {
          "type": "string",
          "minLength": 1
        },
        "end": {
          "type": "string",
          "minLength": 1
        },
        "region": {
          "type": "string",
          "minLength": 1
        }
      },
      "required": [
        "name",
        "url",
        "description",
        "creator name",
        "creator email",
        "creator organisation",
        "begin",
        "end",
        "region"
      ]
    },
```


4.2 SyMo models

In deliverable D6.3 [CIPRNetD63] the SyMo model is described in detail. In this chapter we will recapitulate the main elements that are modelled for the Emmerich Scenario.

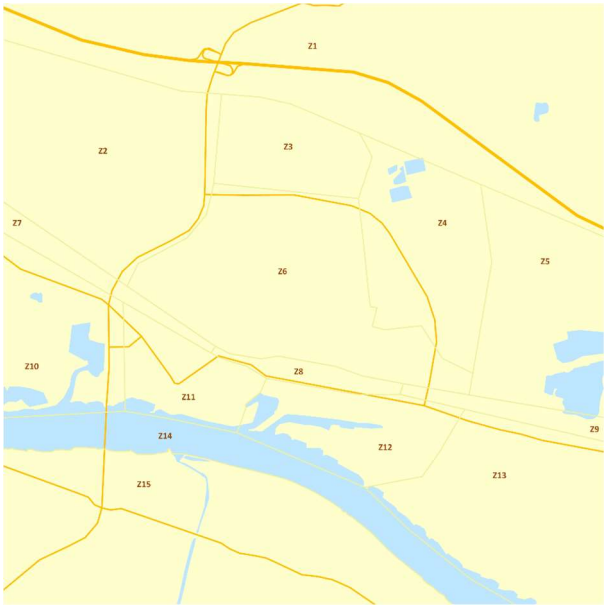


Figure 22: Screen shot of the scenario map with manually defined zones Z1-Z15 inside the map of Emmerich

First of all the map of Emmerich needed to be subdivided into zones, so the resources could be assigned to them. See Figure 22.

Then the main resources, which are actors and environment variables, were defined and created as nodes inside SyMo. The actors are: Residents, evacuees, law forces, action forces and rescue forces. The environment variables are: air pollution, traffic jam and incidents. Then each resource is further specified with a capacity for each location. Figure 23 shows a clipping from the residents and law forces that are mapped to specified location with a fixed capacity.

This schema is used for mapping all resources to specified locations. The graph is then saved as a JSON file and processed with a Lex grammar and Yacc parser [LexYacc].

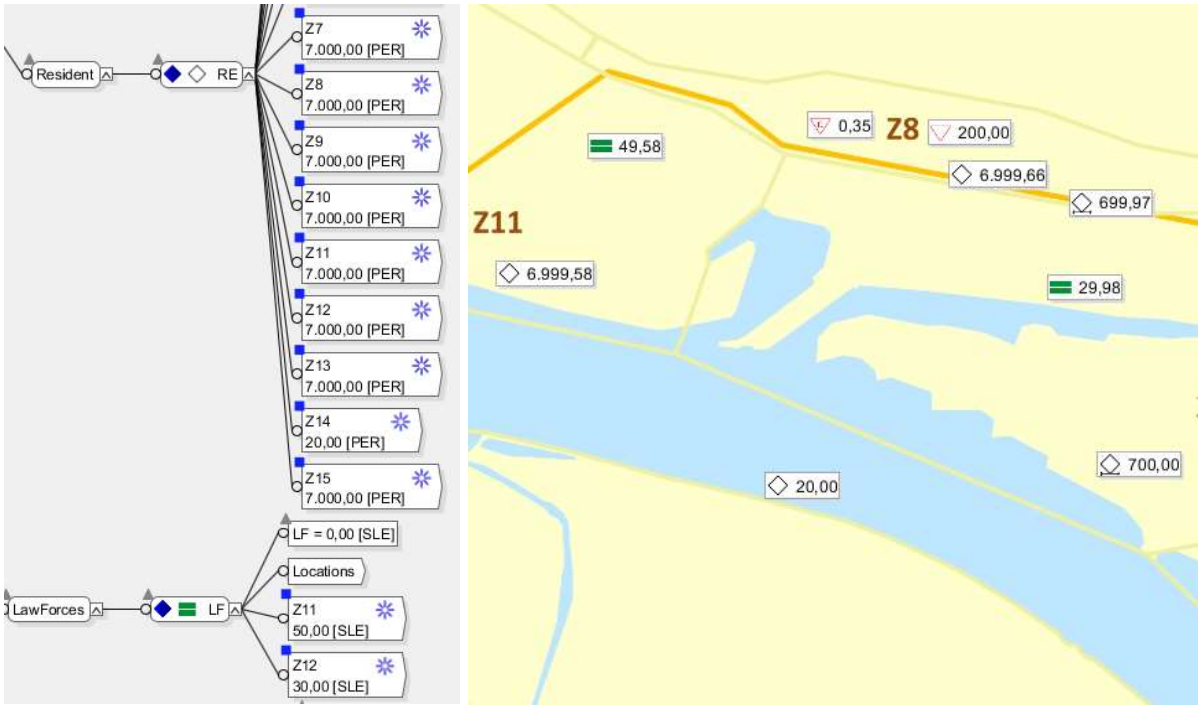


Figure 23: Screen shots of the resources graph and its attributes shown on the map. The lower two leave nodes mean that 50 policemen are initially located in zone Z11 and 30 in zone Z12. The rectangles with the pointed right side link the resources with locations.

4.3 The SyMo to SDL exporter

This chapter explains how the interconnection between SyMo and CIPRTrainer is established. SyMo originally has got a proprietary file format that cannot be parsed easily. Therefor a special export function was written and included in the SyMo software that saves all necessary attributes from the model in JSON format to a file. With this the user is able to create the model inside SyMo and then export it for other applications and further processing.

CIPRTrainer is configured using the SDL file that is also specified in JSON format. To connect both SyMo and CIPRTrainer the SyMo JSON file needed to be further converted into the CIPRTrainer SDL format. For this purpose a LALR parser generator was created. A classical parser consists of two components, a Look-Ahead LR parser and a Backus-Naur Form grammar. The parser is processing the BNF-grammar and can easily convert its content to any different format. For the SyMo to CIPRTrainer exporter it was necessary to parse the SyMo JSON file and convert it into the specified SDL JSON format.

For the creation of the Backus-Naur Form there are various open source lexical analyzer generators available like Lex and Flex. These are necessary to convert the content of a file into predefined tokens, this process is also called tokenization. It is established by mapping regular expressions to one token. The SyMo specific regular expressions are as follows:

```
"TreeText: Start Ressources"            return START_RES;
"TreeText: Ressourcen"                  return RES;
"TreeText: Incidents"                   return CAT_IN;
"TreeText: Environment"                 return CAT_ENV;
"TreeText: Residents"                   return CAT_RI;
"TreeText: Forces"                      return CAT_F;
"TreeText: Other"                       return CAT_OTH;
"TreeText: Effects"                     return CAT_EFF;
"TreeText: Traffic Block"               return SCAT_TB;
"TreeText: Air Pollution"               return SCAT_POL;
"TreeText: Public Information"          return SCAT_PI;
"TreeText: Refugees"                    return SCAT_RU;
"TreeText: Resident"                    return SCAT_RI;
"TreeText: LawForces"                   return SCAT_LF;
"TreeText: Action Forces"               return SCAT_AF;
"TreeText: Rescue Forces"               return SCAT_RF;

[0-9]+[.,][0-9]+        yylval=strdup(yytext); return NUMBER;

"Expr:"                 return EXPR;
Z[0-9]+                 yylval=strdup(yytext); return ZONE;
"IN"                    return IN;
"TB"                    return TB;
"AP"                    return AP;
"PI"                    return PI;
"RU"                    return RU;
"RE"                    return RE;
"LF"                    return LF;
"AF"                    return AF;
"RF"                    return RF;
"t"                     return TIME;
"Sum Polution"          return SUM_POL;
"Sum Loss Residents"    return SUM_LOSS_RI;
"Sum Loss LawForces"    return SUM_LOSS_LF;
\{                       return OBRACE;
\}                       return EBRACE;
"="                     return EQUALS;

\n                      /* ignore newline */
.                       /* ignore all other characters */
```

The left side shows the regular expression and the right side shows the token that is returned to the parser when the regular expression is triggered. The Lex specific header is not shown here. When applying this grammar to the SyMo JSON file the result consists only of tokens.

The second part of the parser generator consists of the LALR parser. For this purpose there are also several open source applications available like Yacc (Yet Another Compiler Compiler). Both tools Lex and Yacc are widely tested and it is common to use both in combination. The programming language for the parser primarily consists of recursive definitions. It starts with the “commands” pattern, this may be

- Empty
- Consist out of one command
- Consist out of multiple commands

Using this recursive technique all possible combination of patterns can be parsed and interpreted. At the end of each recursive definition one element will terminate the recursion process and an output with a specified format can be written to the result file. An example extract from the parser is shown in the following:

```
commands:
        /* empty */
        |
        commands command
        ;

command:
        start_ressources { printf("]\n"); }
        |
        all_tokens
        ;

start_ressources:
  START_RES OBRACE RES OBRACE categorie_expressions EBRACE EBRACE
  ;

categorie_expressions:
        |
        categorie_expressions categorie_expression
        ;

categorie_expression:
        CAT_IN complete_expression
        |
        CAT_ENV OBRACE subcategorie_expressions EBRACE
        |
        […]
        ;

complete_expression:
        OBRACE expressions EBRACE
        ;

expressions:
        |
        expressions expression
        ;

expression:
 EXPR ZONE IN EQUALS NUMBER { printf("{\"type\":\"Incidents\",
 \"location\":\"%s\", \"capacity\":\"%s\" },\n",  $2, $5 ); }
 |
 […]
```

At the bottom all elements have a body, there the recursion is terminated and a JSON string, in accordance with the specification of the SDL format, is written. The result is a JSON Ob-

ject that includes all data from the SyMo model. At the moment only resources are modelled and imported into the SDL file. This is because the resources are the most complex part of the SDL file that cannot easily be created by hand. With SyMo the creation of a multitude of resources and semantic examination of those is done relatively fast.

In the future the parser can easily be extended to process more objects if needed. It is also possible to integrate the parser generator directly into SyMo, so the SDL JSON file can be exported directly from the SyMo GUI. For now there is a command line tool that takes the SyMo JSON file and outputs the SDL JSON file. For this the program automatically calls Lex, Yacc and the SDL schema validator to check the result.

5 Declarative Integration with Complex Event Processing

5.1 Complex Event Processing

Complex Event Processing (CEP) engine is a Java component for analysing, logging and processing data between the simulators, CIPRTrainer and the database server. All the exchanged data are wrapped inside using an event schema and the different components provide adapters to receive and to react to specific events.

An event processing engine typically receives data from an **event producer**. This does the **intermediary processing** on the data like logging, filtering, transformation or complex pattern detection. Finally the data is send to the **event consumer**.

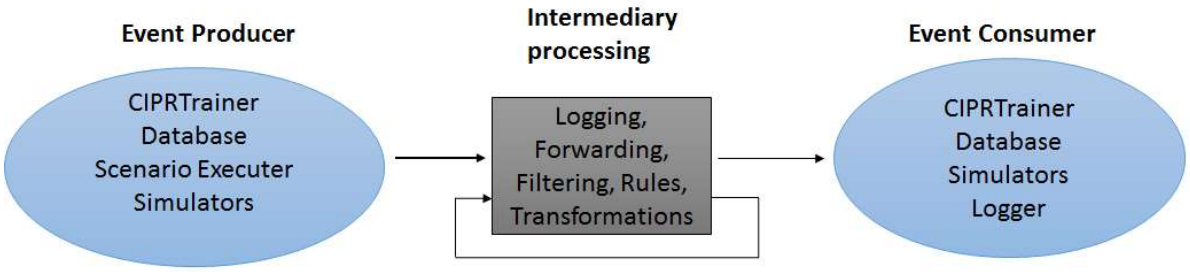


Figure 24: General schema of the complex event processing engine. The intermediary processing contains rules that determine which events are forwarded to which event consumers.

The CEP engine was designed such that its functionality could be easily extended. For this reasons the actual processors, that are responsible for the intermediary processing, are implemented as building blocks. This way the architecture inside the CEP and the corresponding dataflow can be modified and controlled. For example processors can be executed in parallel or sequential. An example of the building blocks architecture is shown in Figure 25.

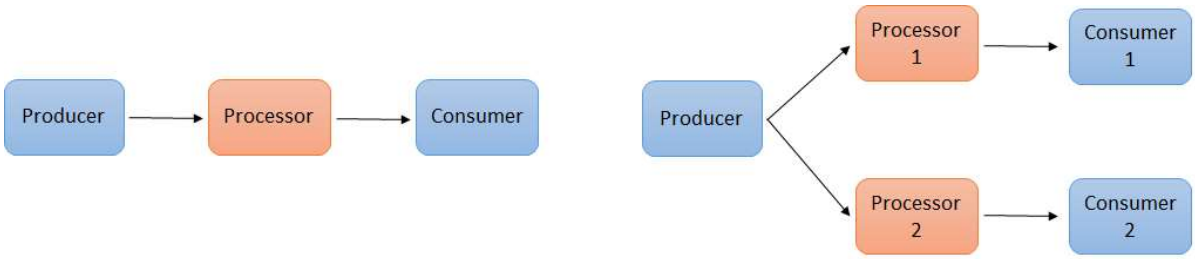


Figure 25: Different architectures of the complex event processing (CEP) engine: On the left side events are passed sequential and forwarded to one consumer. On the right side events are processed in parallel and forwarded to different consumers.

Besides the processors, there are various other components needed to realize a CEP engine. First of all a scheduler is needed for the management of the events and the dataflow inside the engine. The Event data structure must be defined as well as a Queue that can hold multiple events.

An event is defined through a header with standardized attributes and a payload for more advanced and complex information. The header contains an id, type attribute, timestamp and a

location. The payload may be filled with variable attributes that are expected and dependent on the type of the event.

The processor is defined as superclass and beyond it there are different implementations like Log- and Rules-processor. The Logging processor accepts any event and forwards its JSON body over the network and can be viewed inside a web-interface on our main developing server. This way all developers in the team have access to the log when sending events from simulators or the database by just using the browser and calling the server URL.

The Esper-Rules-processor is special in the terms that it includes a declarative programming language for the creation and modification of the rules. Therefore special listeners are registered that act as bridge between the declarative language and the network communication. These are used for talking to other components over the network and each listener represents a method for sending a HTTP specific request method (Get, Post, Put and Delete). This is described in more detail in the following chapter.

Additionally a HTTP adapter for receiving events using the Restful services API is implemented as consumer class. Further there are different classes for time management, using discrete timestamps for the scenario and real timestamps for receiving and logging of events. For testing purposes there also is an implementation of a mock-up for the event producer and event consumer, so the dataflow could be tested with real events and different behaving consumers.

Figure 26 shows the UML class diagram of the described Java classes that show the coarse architecture of the CEP engine. The dataflow of the actual events through the event producer and consumers is shown in Figure 27. The SINCAL, NS-3 and OpenTrack simulators are realized running locally on our main development server and they are capable of sending their state to the CEP engine and also reacting on events. The events that might cause a simulator to react are events from other simulators or scenario specific events. E.g. the SINCAL simulator computes a power breakdown in a specific region, this information is send to the CEP engine and forwarded to the other simulators that depend on a constant power supply, namely NS-3 and OpenTrack. In NS-3 the missing power supply leads to a shutdown of the telecommunication network in the same region where the power breakdown happened.

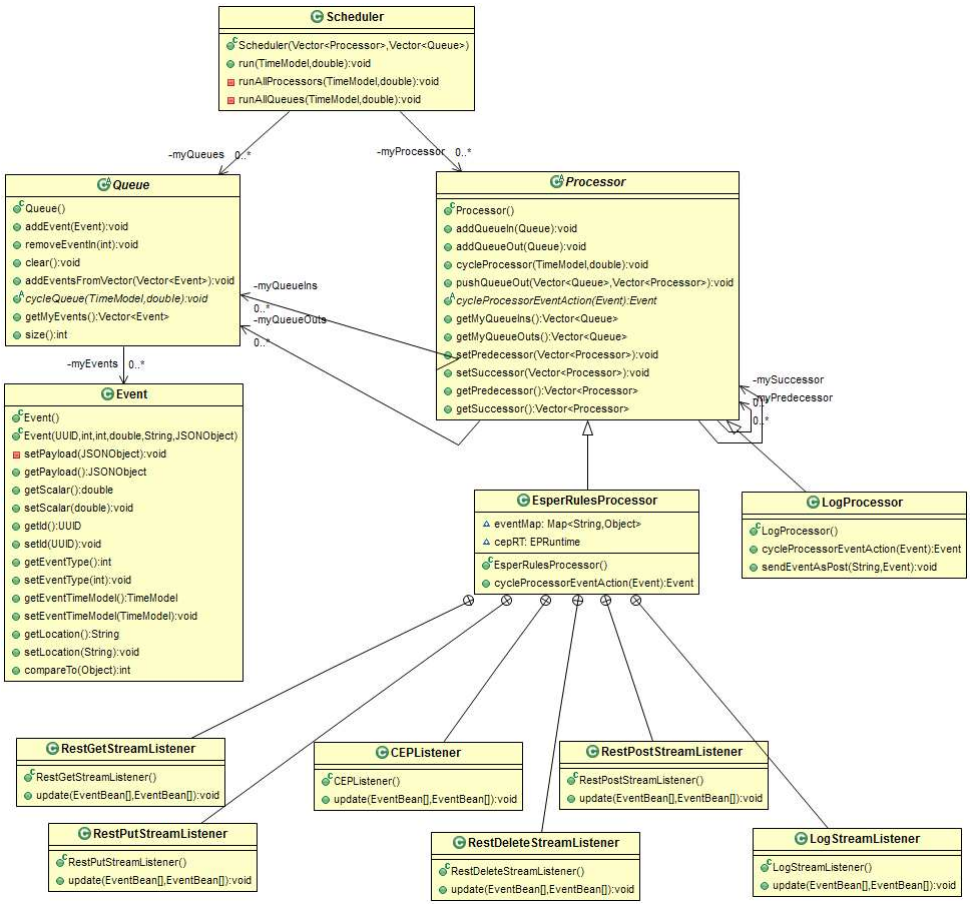


Figure 26: UML class diagram of the Complex Event Processing architecture. Only the most important classes are shown. The system is managed by the Scheduler, this class initiates a Queue for storing the incoming events and forwards them to the processors. The Log processor and EsperRules processor are ordered one after another. Every event is first logged by the Log processor and then processed and evaluated by the EsperRules processor. If a rule is triggered an event is send to the consumers using the network methods that are registered as listeners beneath the EsperRules processor.

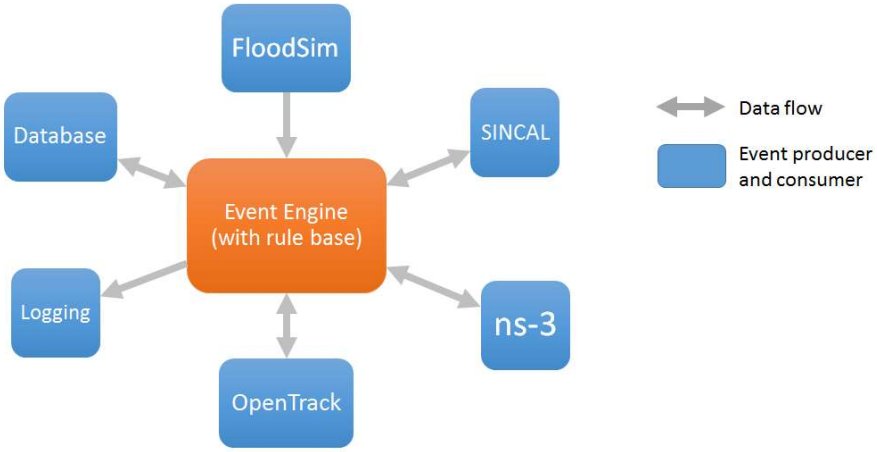


Figure 27: Dataflow of the different components inside the CEP engine. The Sincal, NS-3 and OpenTrack simulators are sending their internal state and any changes that occur during the simulation to the Event Engine. Those changes are evaluated using the declarative defined rules and propagated to the other components if necessary. So those components can send and receive events and are thus event producer and event consumer. The Flood-simulator is only sending and not able to receive and react upon events, therefore it is only an event producer. The database is used for storing scenario and simulator specific variables and the Event engine is reading and writing into it.

5.2 Event Schema

The events are created as JSON objects that are parsed by all component adapters. The proposed JSON format allows CEP engine to understand what kind of message is receiving (in particular from which component) and how the payload is structured. In general a new JSON object is created and sent to the other components if needed. The following table shows the structure of an event. The `id`, `time`, `type` and `location` are mandatory. Payload is adjusted according to the type.

Name	Data Structure	Description
ID	`Java.util.UUID`	Unique id
Time	`TimeModel`	Discrete-time model or Real-time model
Type	`Integer`	Each simulator has predefined type codes: Flood Simulator = 401 SINCAL = 101, 102, 103 NS-3 = 201, 202 OpenTrack = 301, 302 CIPRTrainer = 700, 701, 702, 703 Database = 800 GuiUpdater = 900
Location	`String`	Geolocation as latitude and longitude
Payload	`org.json.simple.JSONObject`	Different attributes encoded as JSON object
Payload .type	`String`	Simulator name or CIPRTrainer specific type
Payload .date	`Java.util.Date`	Real timestamp
Payload .ci_element	`String`	Database column name for the specified critical infrastructure element
Payload .ci_element_type	`String`	Database column name for the specified type of element
Payload .ci_element_id	`String`	Database id of an simulator element
Payload .state	`Enum`	Current status of a simulator element. Options are - Normal - Stressed - Failed - Recovery
Payload .damage	`Double`	Value between 0 and 1 that describes the damage done on a ci_element
Payload .eventTitle	`String`	Title for a CIPRTrainer event
Payload	`String`	Description for a CIPRTrainer event

.incidentDescription		
Payload .location	Double Array	Latitude, longitude and radius that is highlighted on the map
Payload .onmap	Boolean	Definition if the event should be shown on the map with an icon
Payload .icon	String	Icon name that is shown on the map

5.3 Rule Base

This chapter explains how rules are implemented. For this an external Java library was used. The Esper framework [EsperTech] is used for analysing and reacting to events. It is a complete event processing framework. In CIPRNet we are only using one of its modules that allows us to use declarative implementation of rules. So actually it is a programming language that is used beside the Java language.

Rule 1

Prepare and organize received simulator events into categories based on its type

```
insert into SincalEvent
  select payload.get("ci_element") as ci_element,
    payload.get("ci_element_type") as ci_element_type,
    payload.get("ci_element_id") as ci_element_id,
    payload.get("state") as state,
    payload as httpbody
  from Event where type = 101
```

Events with type 101 describe a Sincal-simulator-event. Mostly a simulator event leads to updating the simulators state inside the database

Rule 2

Prepare and forward received events to the simulators

```
insert into SincalAction
  select payload.get("ci_element") as ci_element,
    payload.get("ci_element_type") as ci_element_type,
    payload.get("ci_element_id") as ci_element_id,
    payload.get("state") as state,
    payload as httpbody,
    concat("http://ciprnet/fedsim/sincal/",
      payload.get("ci_element_id"),
      "/",
      payload.get("state")) as url
  from Event where type = 102
```

Events with type 102 describe a Sincal-action-event that is supposed to be received by the Sincal-simulator. So these events get forwarded to the simulator itself.

Rule 3
Send control command to simulators

`insert into SincalAction`
`  select payload.get("ci_element") as ci_element,`
`  select payload as httpbody,`
`  "http://ciprnet/fedsim/sincal/command" as url`
`from Event where type = 103`

Events with type 103 are for controlling the state of the Sincal-simulator. The simulator may be started or stopped this way. Or even more specific Sincal-commands may be commited.

The Rules 1 to 3 are for controlling the simulators. The `type id` describes the kind of event.

For `Sincal` a type 101 event is called a `SincalEvent` and it means that the event is created and dispatched by the SINCAL-simulator. If during the Sincal simulation a specific element is deactivated, this information needs to be delivered to the CEP so the system may check if there are any dependencies to other simulators and write the new state of the element into the database.

A type 102 event is called a `SincalAction`. This means the event is send to the SINCAL - simulator and requires SINCAL to perform a certain action, like deactivating an element. After deactivating the element a type 101 event is sent from SINCAL, like described before.

A type 103 event allows the remote administration of the SINCAL-simulator. It may be started, stopped or certain steps inside the simulated world may be performed.

For example the SINCAL simulation deactivates a transformer inside the scenario, the state of the transformer element is send to CEP. As a consequence the database is updated and the rules are processed. Rule 6 shows an interdependency rule that is triggered when the transformer “EmCE3” changes its state, due to the lost power a router, namely “Router 17” is also deactivated. For this the NS-3 simulator gets an NS3Action event. The NS-3 simulator will deactivate the router in the simulation engine and then send an NS3Event to the CEP.

Rule 4
Derive database update event

`insert into DatabaseUpdate`
`  select concat("http://ciprnet/db/", ci_element , "/", ci_element_type, "/", ci_element_id, "/state/", state) as url,`
`  httpbody as httpbody`
`from SincalEvent`

The state of the simulated world is inside the database. Meaning resources for the scenario and most simulator objects are all maintained in one database. Doing damage in the simulated world is done by doing a database update and manipulating the appropriate columns. This is achieved through database-events.

Rule 5
Derive GUI update notification event

`Insert into GuiUpdateAction`
`  select "http://ciprnet/wfs/notifier" as url,`
`  payload as httpbody`
`from Event where type = 900`

No parameters are designed right now. In the future however incremental GUI update should be considered for performance reasons. That means the element and the target state of the element can be coded in the event itself as payloads.

Rule 6
Dependency rule from Sincal to NS3

`insert into NS3Action`
` select ci_element as ci_element,`
`  ci_element_type as ci_element_type,`
`  "Router 17" as ci_element_id,`
`  state as state,`
`  httpbody as httpbody,`
`  concat("http://ciprnet/fedsim/ns3/", "Router 17" , "/" , state)`
`  as url`
`from SincalEvent where ci_element_id="EmCE3"`

Example dependency rule that reacts when a Sincal transformer element e.g. "EmCE3" changes its state and triggers an action event to the NS-3 simulator to also change the state of the router element "Router 17".

Rule 7
Network communication REST message

`insert into RestPost`
` select url as url,`
` httpbody as httpbody`
`from GuiUpdateAction`

For the network communication the RestPost event-stream is created. This stream is registered to a Java listener and will send a HTTP Post message to the given URL with httpbody as data.

Rule 8
CIPRTrainer action for moving forces
`insert into CiprTrainerEventForces` `  select payload.get("forces") as forces,`

```
    payload.get("duration") as duration,
    payload.get("activity") as activity,
    payload.get("resource") as resource,
    payload.get("source") as source,
    payload.get("destination") as destination,
    payload as httpbody
  from Event where type = 702
```

Action rule for moving forces from one location to another. This rule is triggered when an action inside the CIPRTrainer GUI is executed by the user.

Rule 9
CIPRTrainer action for informing the population

```
insert into DatabaseUpdate
  select   "http://ciprnet/db/human/sheltered/N3198E4062/number/2000"
as url,
    httpbody as httpbody
  from CiprTrainerEventCMA where activity = "Inform general public"
```

Action rule for informing the population about any dangers e.g. toxic smoke. The information is coded into a URL and the attributes are directly written into the database.

5.4 Event Processing Network

One major feature of the Esper framework is the possibility to define event streams with the declarative programming language. This way a complex dataflow network can be created. This is useful for repeating tasks. Like a simulator changes its internal state and sends the appropriate event to the CEP engine. This action will always lead to subsequent events that need to be created and send by invoking a corresponding rule.

An illustrative sample event-processing network is presented in Figure 28. It contains five event streams from `S1` to `S5`, where S1 is the base stream. All external events are sent to `S1` for logging and further event processing.

An illustrative sample event processing network is presented in Figure 28Four rules are defined to derive the complex events based on the atomic events from external and other derived complex events:

- `R1 : S1 → S2` It inserts a new event to `S2` as long as a new event is inserted into `S1` and the rule condition is evaluated to `true`.
- `R2 : S1 → S3` It inserts a new event to `S3` as long as a new event is inserted into `S1` and the rule condition is evaluated to `true`.
- `R3 : S3 → S4` It inserts a new event to `S4` as long as a new *derived* event is inserted into `S3` and the rule condition is evaluated to `true`. This causes a cascading effect of event processing.
- `R4 : S1×S4 → S5` It inserts a new event to `S5` as long as a new event is inserted into `S1` or `S4` and the rule join condition between `S1` and `S4` is evaluated to `true`. This indicates an event stream join operation.

Listeners with imperative code, like programs written in Java or Python, can be registered on top of separate streams to perform real actions related to outside physical or cyber systems like change the state of a database, update the graphical user interfaces etc. One of the biggest advantages of this kind of design is combining both declarative and imperative world to provide maximum runtime performance and development productivity.

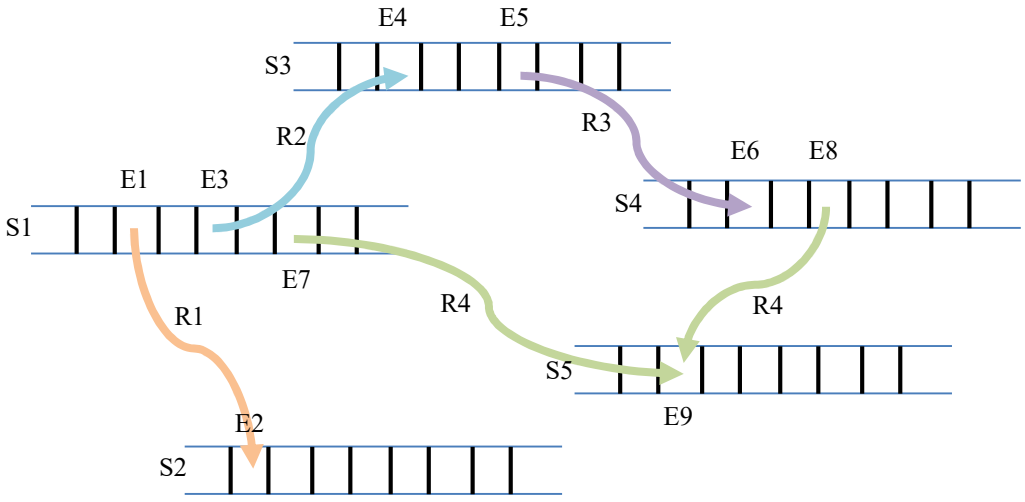


Figure 28: An example event processing network in CIPRTrainer

Figure 29 shows a simplified version of the CIPRNet event stream network. The events and derived events are shown as oval drawing and the components like simulators, database and CIPRTrainer as boxes on the right side. The derived events and thus the dataflow is shown through the arrow lines between two events and the constraint is written above each line.

The provenance is a simple JSON coded event like described in chapter 5.2. The type of the event is leading to a specific derived event. The first derivation of an event mostly leads to a simulator –event, -action or –command. Beside the simulator-events, there are also CIPRTrainer specific events like CiprTrainerAction and GuiUpdateAction.

An example for a simulator-event is a SincalEvent, this has the type 101. This specific event always leads to a database update. This is hardcoded inside the event stream network and is shown through the arrow lines from the SincalEvent stream to the DatabaseUpdate stream. This way the changes made inside the Sincal simulator are written into the database and thus propagated to the CIPRNet network. The other components are informed that the database has changed and can act accordingly.

A simulator-action, e.g. a SincalAction with type 102 is used when the Sincal simulator should be triggered to change its internal state and to initiate a new simulation. For this one of the network streams RestPut, that sends a HTTP Put request, is used. This way the Sincal simulator is receiving the event, reacts upon it and then again sends an event as confirmation to the network. This may lead to further events when other simulators are involved.

The CiprTrainerAction event is used when any changes inside the CIPRNet network occurred that need to be processed and lead to a state change inside CIPRTrainer. For reasons of performance a second event type GuiUpdateAction is used to actually trigger a GUI update inside CIPRTrainer.

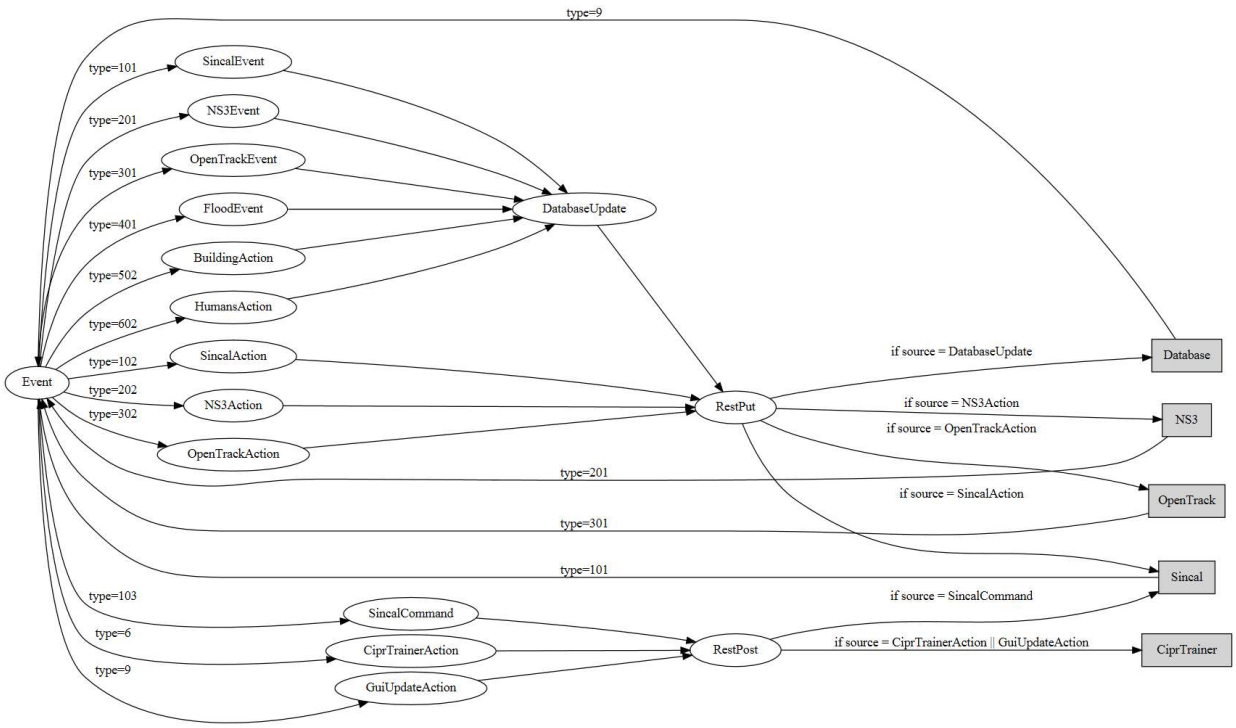


Figure 29: The CIPRNet event stream network.

The network communication streams are implemented in Java and are directly referenced to the event streams RestPost, RestPut, RestGet and RestDelete. The graph shows only two of those streams. The RestPut stream is called from the DatabaseUpdate stream and from the simulator-action streams. This causes the RestPut stream to send a HTTP Put message to the database and to the Sincal-simulator respectively.

6 Federated Simulation – Models, Simulators and Adaptors

The federated simulation environment consists of CI models, dedicated domain-specific simulators including both CI simulators and threat simulators, and the simulator adaptors that enable the communication with other CIPRTrainer components. This section gives more technical details about the implementation.

6.1 Dependency Model

Dependencies are pervasive in networks of critical infrastructures. This is the root for serious cascading effects, e.g. the unstable power supply will cause the failure of telecommunication base stations. On a coarse level, the dependencies exist on the infrastructure level like different sectors (Figure 30); on the fine-grained level, they exist on different elements within certain infrastructure instances.

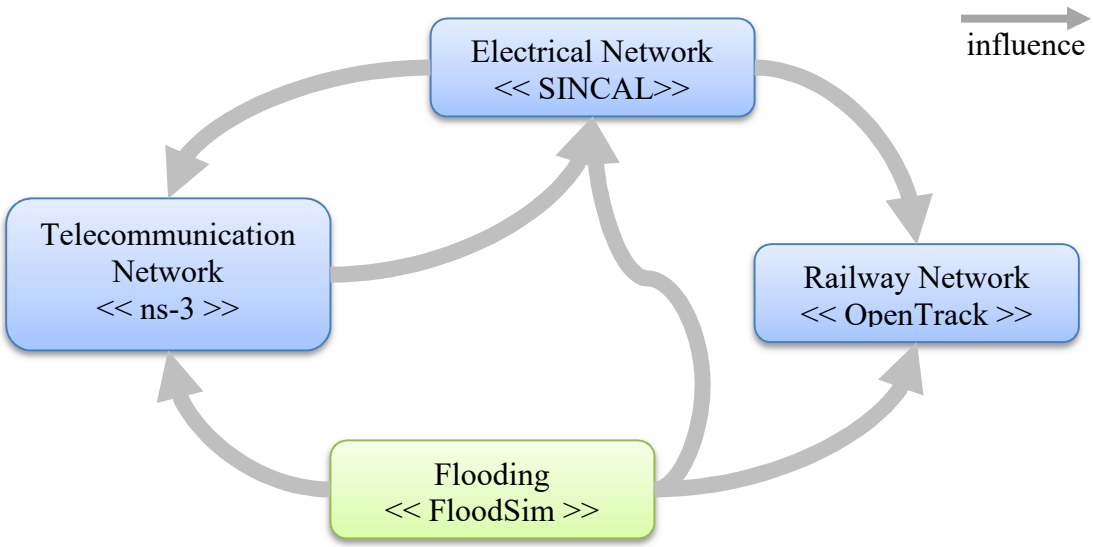


Figure 30: The dependency graph of simulation domains (corresponding to CI sub-sectors)

Detailed CI element dependency is encoded as a more complex graph, which is similar to the directed graph illustrated in Figure 30. As presented, it is significantly more complex than the sector dependency graph.

The dependency graph is modelled as declarative rules in the scenario description file (see Section 4.1 and Section 5.3). They can be retrieved at runtime by the rule engine to determine the dependencies. On the other side, the SyMo scenario editor can also export the dependency model offline for runtime use. A visualisation of the rule base in provided in Figure 29.

6.2 Refined Simulation Models

Within this reporting period, the already available simulation models of the involved critical infrastructures, which are described within deliverable D6.3, were improved significantly regarding the complexity, the degree of detail and the realism.

The model of the electrical power network was extended by the plausible reproduction of an electrical distribution network for the city of Emmerich. As, in contrast to the electrical transmission network, no data about the state of real electrical distribution within the city is

available to the public, a fictive, but yet plausible model of the distribution network was created. The structure of this model is based on a typical distribution network of a small city, the constraints and particularities given by the transmission network and the topographic structure of the city were taken into account. Figure 31 shows a part of the model. The distribution network is modelled up to the 20 kV medium voltage distribution network layer, with cabinet feeders as the endpoints of the network. Each of the cabinet feeders transforms the 20 kV electrical voltage to 400 V low voltage, which is delivered to the single houses. As the low voltage network is not covered by the model, each cabinet feeder provides power supply to a specific small area within the city and therefore usually supplies several houses with power. In dedicated zones, i.e. industrial areas or the Emmerich harbour area, the model comprises single cabinet feeders, which provide higher output voltages.

The distribution network is modelled with PSS®SINCAL (see section 6.4.1). Each cabinet is represented as a *load* element, which can be understood as the cabinet and the connected electrical consumers.

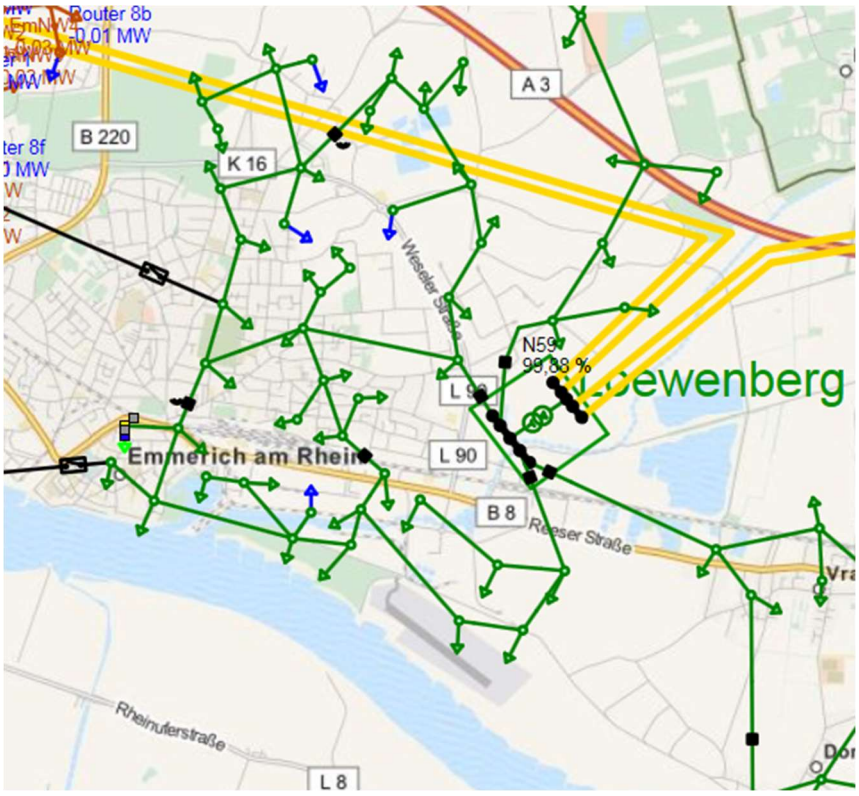


Figure 31: The model of the electrical distribution network. The green lines represent the 20 kV underground cables, the green triangles depict the cabinet feeders. The connection of the distribution network to the transmission network via the "Station Loewenberg" transformer substation can be seen.

Besides the model of the electrical power network, the model of the telecommunication network was subject to a minor revision, which aimed to add more detail to the model by including more endpoint routers, and to improve the plausibility of the model by adjusting the several routers positions and the links between them.

6.3 Threat Simulators

The flood simulator `FloodSim` is based on SOBEK as indicated in D6.3 [D6.3], which further uses the Saint-Venant equations with the so-called staggered grid numerical scheme

[Stelling03]. The simulation itself is however quite slow and is not suitable for online real-time flood simulation. Therefore the simulation is done before-hand and cached in a PostGIS database.

One of the distinctive features provided by SOBEK is its deterministic simulation. It provides an excellent prerequisite to cache the simulation results in a high performance database and other layers in a cache hierarchy (see Figure 32).

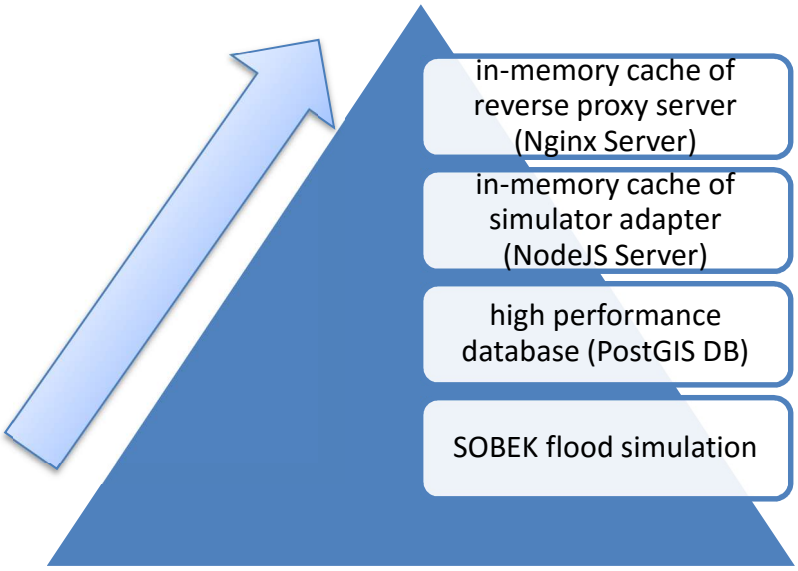


Figure 32: Cache hierarchy of the flood simulation results

The UML sequence diagram below explains the workflow of how `FloodSim` is integrated into CIPRTrainer.

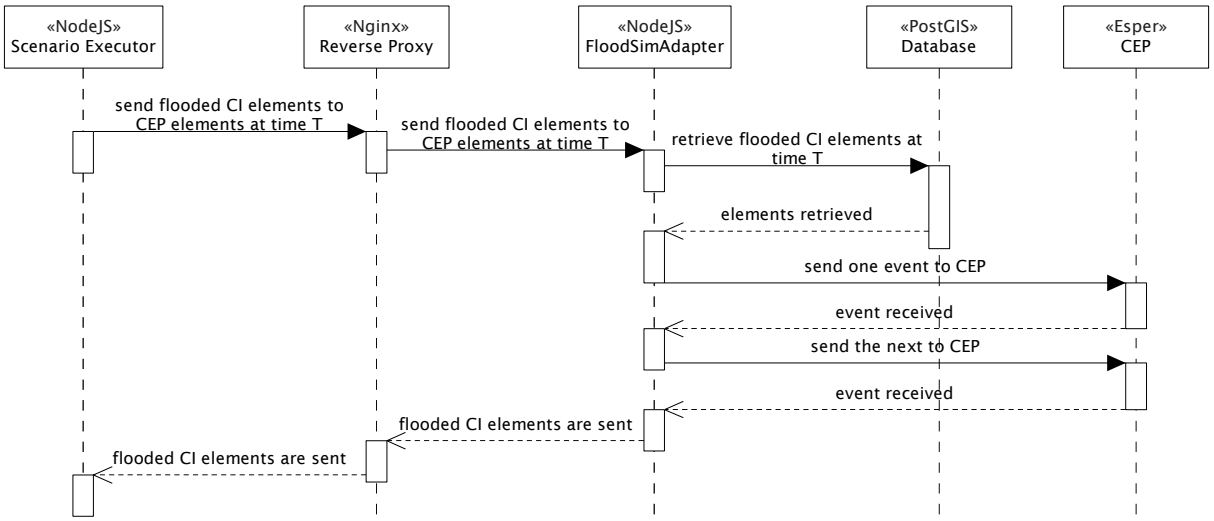


Figure 33: UML sequence diagram illustrating the workflow of retrieving flood simulation results

One caveat: only the flooded CI elements are retrieved by correlating the flood simulation results, i.e. water level, with the coordination of the CI elements stored in the same database.

The reason to do that is to fully make use the declarative and indexing feature of relational spatial database systems – PostGIS in our case. It is much more convenient and efficient to do this correlation operation inside of the database system than with other high-level program-

ming languages like Java. The correlated results can also be pre-calculated and materialised in a relational table `ci_under_water` under the schema `floodsim`:

```
insert into floodsim.ci_under_water (id_ci, water_depth, step) as
with s as (
  select e.id id_ci, ST_Value(rast, ST_Transform(e.geom,28992))
 v, f.rid
  from cielement.ci_element_point e, lizard.rees f
)
select id_ci, v, rid
from s
where v is not null
order by v desc
;
```

The core operation in the query above is the PostGIS function `ST_Value()` that accepts two parameters:

- The flood simulation results as a raster data type in the database
- A given location in the same spatial reference system (SRS)

As the example above illustrated above, the raster is coded with the SRID 28992. Therefore, therefore the coordination of the to-be compared location should be transformed into this spatial reference system at runtime or query time. After that, this function returns the pixel value of that location – with possible interpolation since the raster is discrete and the location can be between two raster points. An example of the interpolation is illustrated in Figure 34. The area represented by the polygon containing the black cross point is less flooded than the one with the green cross point. The polygon with the orange cross point is mostly flooded. In this example, the water level of the black cross point can be retrieved directly from the raster data while the green and the orange one need to be interpolated by considering the neighbouring raster points.

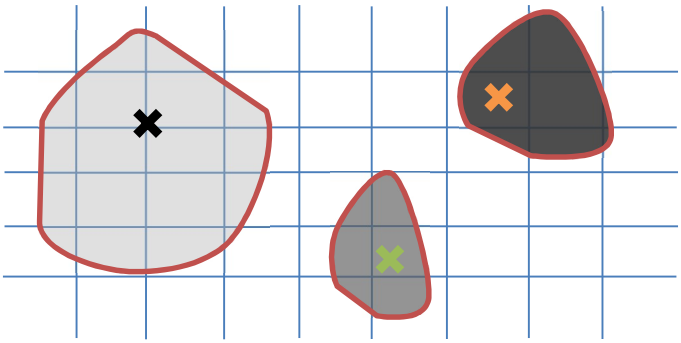


Figure 34: Interpolation of location values of raster in PostGIS

6.3.1 Event Generation

Threat simulators like flood simulators are mostly read-only system. That means, they generate events as event producers and do not received events as event consumers. The generated events are sent to the event processing system via a RESTful Web Service call.

6.4 CI Simulators

Within the CIPRNet project, three different types of infrastructures are taken into account: the electrical infrastructure including the transport and distribution networks, the telecommunication infrastructure, and the railway infrastructure. For each of these different infrastructure types, dedicated simulators exist, which depict the specific behaviour of the corresponding infrastructure. In the CIPRNet project, the software *PSS®SINCAL* is used to simulate the behaviour of the electrical infrastructure. For the simulation of the telecommunication infrastructure, the software package *ns-3* is brought to use, and for the simulation of the railway infrastructure, the software *OpenTrack* was chosen. In the following subsections, each simulator will be explained in detail.

6.4.1 PSS®SINCAL

The commercial software PSS®SINCAL by Siemens PTI is designed for the planning and analysis of electrical networks. It provides a graphical user interface to specify the logical and spatial structure of the network.

Several aspects of an electrical network can be analysed with the software, within the CIPRNet project. The most important one is the calculation of the *load flow* throughout the network. The PSS®SINCAL's load flow calculation solver uses numerical methods like Newton-Raphson to calculate the flow of electric power through the transmission and distribution network.

The basic simulation elements in PSS®SINCAL are *Nodes or Busbars*, *Node Elements* and *Branch Elements*. A network's structure is described by its nodes and branches. The branches connect two nodes to each other. A branch (or branch element) goes from the starting node to the end node. Node elements are connected to the nodes. For a load flow simulation, the flow of power and various other values are calculated for each node in the network.

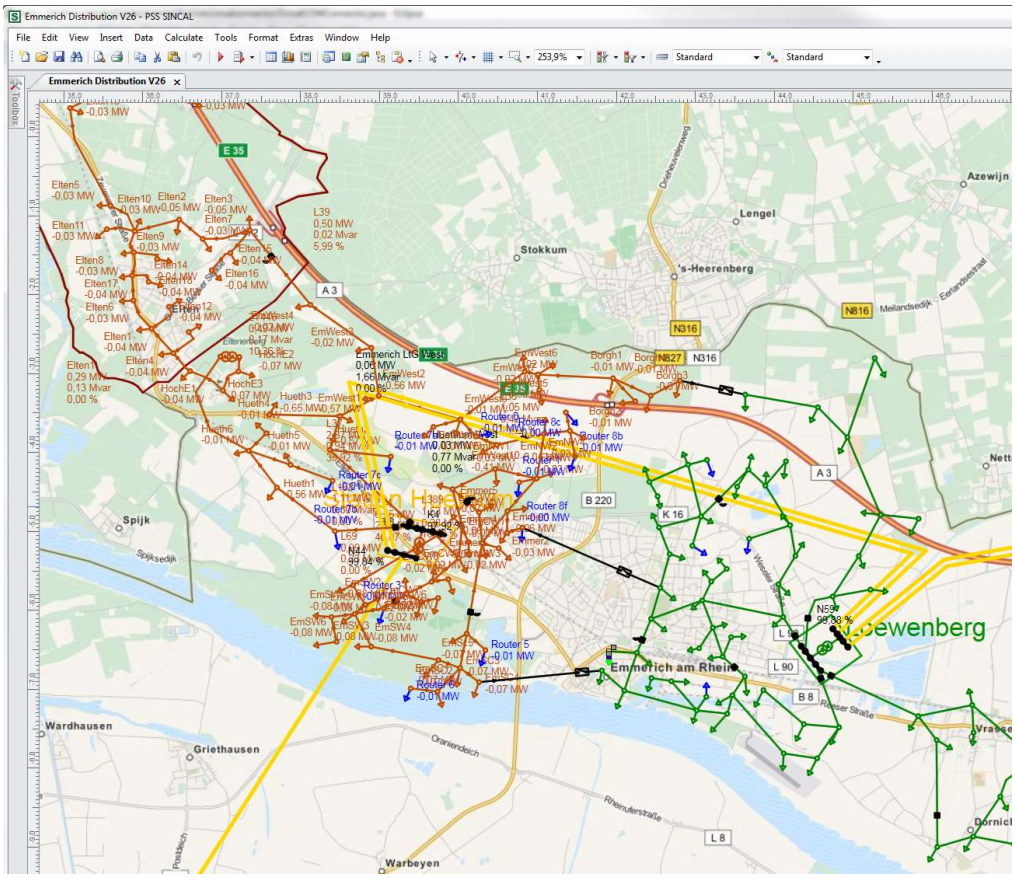


Figure 35: Screenshot of the PSS®SINCAL software. The graphical user interface shows the elements of the distribution- and transport network and their interconnections. The view can be enriched with geographical maps to support the modelling of real-world conditions.

6.4.2 ns-3

The Network Simulation version 3 tool (ns-3) is a discrete-event network simulator, provided free under a GNU GPLv2 license. Its purpose is to provide an "open simulation environment for networking research", including IP-based networks and non-IP based communication networks. The ns-3 software package is built using C++ and Python with scripting capability. The simulator is build up by several modules containing models for real-world network devices and protocols. Compared to PSS®SINCAL or OpenTrack, ns-3 is more a software framework which enables the user to implement simulation solutions which are tailored for a specific use case. An ns-3 model basically consists of the elements

- Nodes
- Applications
- Channels
- Net-devices
- Topology Helpers

These elements form basic abstractions of the real world conditions. A *Node* is in this connection a basic computing device which is part of a network. It can be seen as a computer which is capable to be equipped with functionality (applications, network devices, protocol-stacks etc.). An *Application* is the abstraction of a computer-program which generates network activity. The abstraction for the data transport media (i.e. cable, WiFi, etc.) in ns-3 is called *Channel*. Nodes are connected to these communication channels, this means channels form links

between nodes. The networking devices consisting of both hardware and software drivers are represented in ns-3 through a *Net Device*. These Net Devices are "installed" on a specific Node of the simulated network and assigned to a channel. To connect a node to multiple channels, a Net Device has to be installed on the node for each channel.

A simulation model in ns-3 has to be created programmatically, i.e. a C++ program has to be written and compiled, within the code, object instances of the ns-3 elements have to be created and connected to each other. As the setup of a large network could be tedious, the ns-3 library provides so called *Topology Helper* classes, which make these process more convenient. The simulation is compiled and started through the build system *waf*.

6.4.3 OpenTrack

The railway simulation software OpenTrack is a commercial software system for the planning and simulation of railway networks developed by the Institute for Transport Planning and Systems[8]. OpenTrack simulates all relevant operational processes on a railway network, for example train movements, the signalling system, the dispatching of train services, as well as delays, failures and incidents. It is mainly used for the analysing the capacity of railway lines and stations, the construction and analysis of timetables and the analysis of signalling systems. The software is used by various railway infrastructure operators like Deutsche Bahn or SBB (Swiss national railway cooperation).

The input entities of an OpenTrack simulation is data about the rolling stock, i.e. weight, maximum velocity, acceleration etc. of the simulated trains, the topology of the railway infrastructure, i.e. railway tracks, signalling, railway stations etc., and timetable information for the simulated trains. The output of the simulation is a set of various diagrams, for example way / time diagrams for each train, track occupations and many other data.

The actual model of the railway infrastructure is generated by through the graphical user interface of the software. The user can input the track layout in a schematic way, as well as the other input entities. The railway network is represented by a so called *double vertex graph*. In a double vertex graph, the graph's vertices (i.e. the nodes of the graph) do not appear alone, but always together with a second vertex. Thus, contrary to classical (single vertex) graphs, double vertex graphs can provide information at each vertex about the edge the vertex has been reached through. Figure 36 shows a simple example of a track layout represented as double vertex graph. Valid paths through the double vertex graph have the form $v_i v'_i$, $v_{i+1} v'_{i+1}, \ldots v_{i+k} v'_{i+k}$, or in the reverse form $v'_i v_i$, $v'_{i+1} v_{i+1}, \ldots v'_{i+k} v_{i+k}$. This ensures that simulated train's behavior at network branches (i.e. railway switches) is always plausible. Within the example given in Figure 36, the path $v'_1 v_1$, $v'_2 v_5$ would therefore be illegal.

[8] http://www.ivt.ethz.ch/index_EN

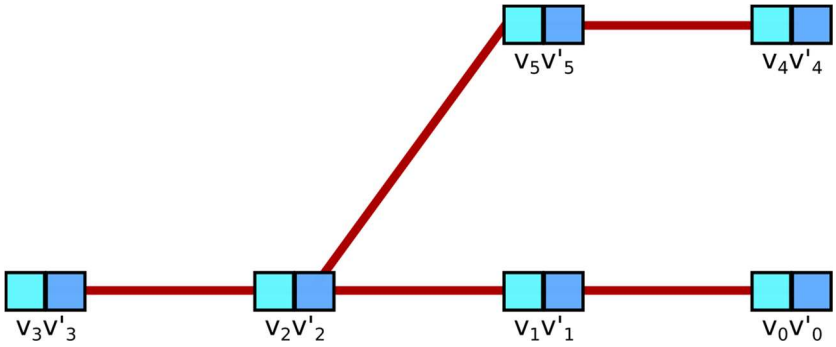


Figure 36: A simple railway track layout with one switch represented as double vertex graph.

Various attributes can be assigned to vertices, for example, vertices can be marked as *station vertices*, indicating a single track of a railway station.

The simulation is controlled through the graphical user interface of OpenTrack, the simulation can be started, stopped and run to a specific simulation time. Besides the manual simulation control using the GUI, OpenTrack also offers access to its functionality through the OpenTrack API, which is described in section 6.5.4.

6.5 Simulator Adaptors

The Simulator Adaptors implemented within WP 6 are one of the core-elements of the federated simulation environment which is used by the CIPRTrainer software system. The simulation adaptors provide a unified access to the specific CI-simulators used by the CIPRTrainer and implement functionality for the control of the simulators and for the retrieval of simulation data. As each simulator usually provides its own specific access mechanism, a dedicated simulation adapter has to be implemented for each CI-simulator used within the federated simulation environment.

6.5.1 Architecture and Operation of the Adaptors

The intercommunication between the specific simulators is implemented through intercommunication between the according simulator adaptors. This communication between the adaptors is based on the event system described in section 5 and follows the REST software-architecture principle. A simulator adaptor in its generalized form implements a REST endpoint with HTTP as the application layer protocol. The simulator adaptor defines a set of commands which can be called from outside via the REST interface. These commands mainly include calls to start, stop or step the simulation, and calls to set or retrieve the simulated operational state of a specific CI-element. Each CI-element simulated by a specific simulator is identified with a unique name; the operational state comprises the values *normal state*, *failed state*, *stressed state* and *recovery state*. The mapping of a specific simulator's state variables to one of these four common operational states is subject to the concrete implementation of the simulator adaptor and depends eventually on CI-simulator used.

A graphical overview of the general architecture of a simulation adaptor is depicted within Figure 37. Each simulation adaptor implements a common set of sub-modules, which serve as the connection to the CEP engine. A HTTP server is used to implement the REST-based interface to the event system; an event is submitted to the simulator adaptor by sending a HTTP request to the simulator adaptor's internal HTTP server, using a dedicated URL, depending on the command being passed to the simulator through the simulation adaptor. The *start*, *step* and *stop* commands are submitted through a HTTP POST request, the specific command is

encoded using a JSON-based message object within the body of the HTTP request. The *set state* command is issued by sending a HTTP request using the PUT method; the *get state* command is send to the adaptor using the GET method. In both cases, *get state* and *set state*, the identifier of the according CI-element is encoded in the URL of the request. State changes in the simulated CI-elements are propagated to the outside using the event-system, i.e. events are sent to the CEP engine using its REST-based interface. To realize this, each simulator adaptor implements a HTTP connection to the REST endpoint of the CEP engine. The specific information of a CI element state event is encoded as JSON object and send in the body of the HTTP request. This information comprises, amongst others, data fields containing the simulated CI-element's name, type, the operational state and the time the reported state change has occurred.

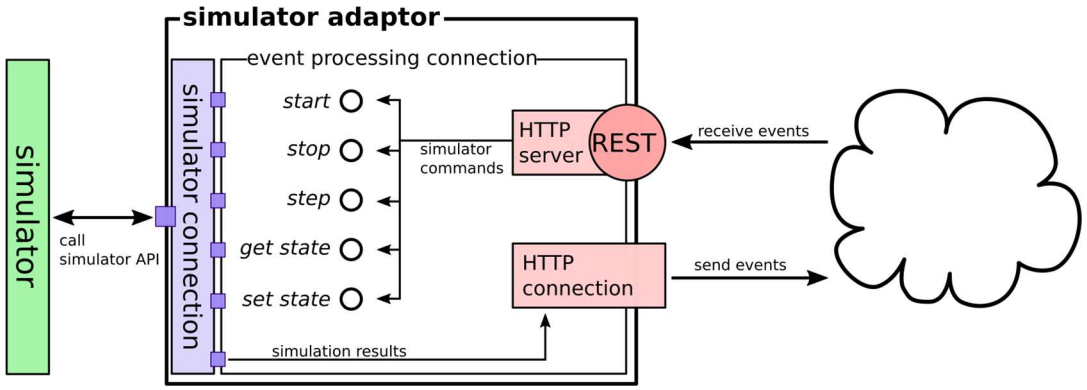


Figure 37: Graphical overview of the simulation adaptor. Events are sent and received through the network via HTTP. A REST-based interface to the simulator commands is implemented through a HTTP server. CI element state changes as the outcome of a simulation are sent to the CEP engine using HTTP requests. The simulator connection sub-module realizes the concrete connection to a specific simulator.

Besides the common set of functionalities to implement the connection to the CEP-engine, each simulator adaptor comprises a specialized connection module to the concrete simulator. As each CI simulator provides its own interface, this module has to be implemented for each simulator used within the federated simulation environment, while the common sub-modules can be reused for each new simulation adaptor. In Figure 38, an overview of the three CI-simulators used in the CIPRTrainer software system, *PSS Sincal*, *NS3* and *OpenTrack*, as well as their corresponding simulation adaptors and the simulator-specific API mechanisms are shown. The configuration of each specific simulation adaptor and its interface to the connected CI-simulator is described within the following subsections.

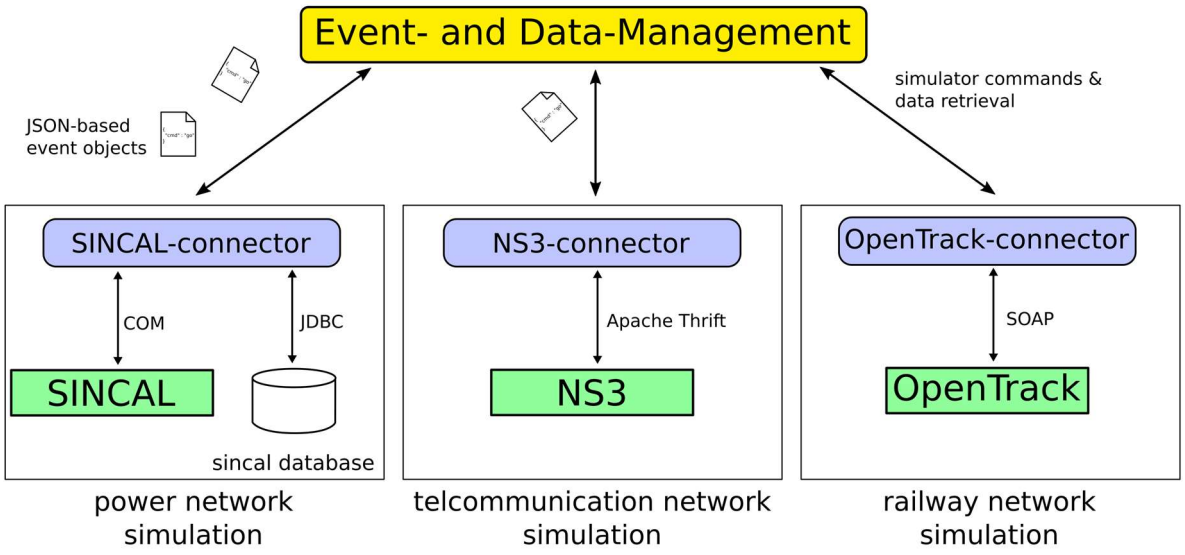


Figure 38: The simulator adaptors (violet colour) implemented for the CIPRTrainer with their connection to the specific simulators and to the CEP system (green color).

6.5.2 Sincal Adaptor

The sincal adaptor implements the connection of the PSS®SINCAL simulator for electrical infrastructures (c.f. section 6.4) to the federated simulation environment. This simulation software provides an interface using the Microsoft *Component Object Model (COM)*. This interface allows the usage of the SINCAL simulation functionality from "outside" in an automated way, i.e. the simulation functions can be called from another process. Within the current state of the implementation of the sincal adaptor, the COM interface is mainly used to start a load flow simulation on a given electrical infrastructure model. The interoperation with the SINCAL COM interface is implemented through the external library *com4j*[9], which provides tools to generate Java definitions of the COM type libraries provided by SINCAL. These Java definitions are then used by the sincal adaptor to access the according functionality.

Besides the COM-based interface, the PSS®SINCAL simulator makes use of a dedicated database management system where all the model data, as well as simulation results are stored. Within the CIPRNet project, the SINCAL simulation software is configured to *Microsoft Access* as primary database. The open source JDBC *UCanAccess*[10] driver is used to read data from the SINCAL database, and to write data to it. The SINCAL database is used to retrieve the operational state of each element of the electrical infrastructure by reading the corresponding data fields from the database, as well as to set the operational state of a specific CI-element by writing to the database, changing the relevant data fields of the CI-element.

6.5.3 ns-3 Adaptor

The ns-3 adaptor implements the connection of the ns-3 simulator (cf. section 6.4.2) to the federated simulation environment. It implements the general architecture for the simulation

[9] https://com4j.java.net/

[10] http://ucanaccess.sourceforge.net/site.html

adaptors (cf. section 6.5.1) by providing implementations of the interfaces defined by this architecture. Compared to the CI-simulators PSS®SINCAL and OpenTrack, the ns-3 software system does not provide an API to access the simulator's functionality from outside. As ns-3 can be more seen as a flexible software library and concrete simulations (or the simulation models, respectively) have to be implemented and compiled to a working program, the realization of an API is subject to the user. This dedicated API functionality was implemented using the *Apache Thrift*[11] interface definition language and communication protocol.

Thrift is a software library and set of code-generation tools developed at Facebook which provides functionality to enable inter-process communication via RPC independent of the used programming language. Datatypes and service interfaces are defined in a single language-neutral file, the necessary code to build RPC clients and servers is generated automatically using Thrift's toolchain. The RPC functionality provided by Thrift is implemented using a network based client-server approach, where various protocols and transport mechanisms are provided by the library.

Within the federated simulation environment, the Thrift server is implemented within the actual ns-3 application, i.e. within the simulation of the telecommunication infrastructure. The Thrift client is implemented within the ns-3 adaptor. The ns-3 adaptor uses a binary protocol for the exchange of data via RPC, which has advantages in the processing speed through the receiver. On the transport layer, a Socket based communication mechanism is used. This distributed architecture allows the separation of the ns-3 adapter from the ns-3 simulation, i.e. ns-3 can be run on a different machine than the adapter and, for example, the federation environment, yielding in a scalable system architecture. Furthermore, with this approach, a multi-user setup can be implemented by running different instances of the simulation, with each simulation instance listening on a different TCP port.

6.5.4 OpenTrack Adaptor

The OpenTrack server communicates through SOAP (Simple Object Access Protocol, see Figure 39). Messages are encoded in the HTTP body using XML-based web service description language (WSDL). For instance, this SOAP message tells OpenTrack to start the simulation:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope xmlns:SOAP-
 ENV=http://schemas.xmlsoap.org/soap/envelope/>
  <SAOP-ENV:Body>
    <startSimulation>
  </SOAP-ENV: Body>
</SOAP-ENV:Envelope>
```

Messages can be exchanged asynchronously. It supports two basic pattern of operation:

- One-way: The server takes a request by the client
- Notification: The servers responds to the client

[11] https://thrift.apache.org/

The example message above illustrates a One-way message. Notification messages have the same information structure and can look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope xmlns:SOAP-
 ENV=http://schemas.xmlsoap.org/soap/envelope/>
  <SAOP-ENV:Body>
    <trainDeparture trainID="IC1031" stationID="BA" time="23534">
  </SOAP-ENV: Body>
</SOAP-ENV:Envelope>
```

For instance, this message tells the OpenTrack connector that the train IC1031 departed successfully from station BA at simulation time 23534. For each train, the OpenTrack server notifies the adaptor regularly about its current position, time, delay and route offset.

Possible OpenTrack notifications are:

```
simStarted(time)
simStopped(time)
ping(time)
trainPositionReport(trainID, routeID, routeOffset, time, delay,
 speed, acceleration)
trainArrival(trainID, stationID)
trainDeparture(trainID, stationID, time, delay)
trainPass(trainID, stationID, time, delay)
routeReserved(trainID, routeID, time)
routeReleased(trainID, routeID, time)
routePartReleased(trainID, routeID, partID, time)
```

To simulate an accident in the Emmerich scenario the OpenTrack adaptor has to interpret this accident in the OpenTrack formalism. It is not possible to simulate a derailment *directly*. Therefore, an adequate interpretation has to be chosen that corresponds to the accident. For instance, a constant occupation of a route element or a power outage for that specific route could simulate the derailment since no trains are able to use or pass it.

The connector implements the five methods (start, stop, step, getState and setState, see Figure 37Figure 37) and wraps incoming requests into a SOAP request since the OpenTrack server is based on SOAP. The content of a SOAP notification will be extracted and wrapped into REST request such that the simulators and the event processing engine can process the data.

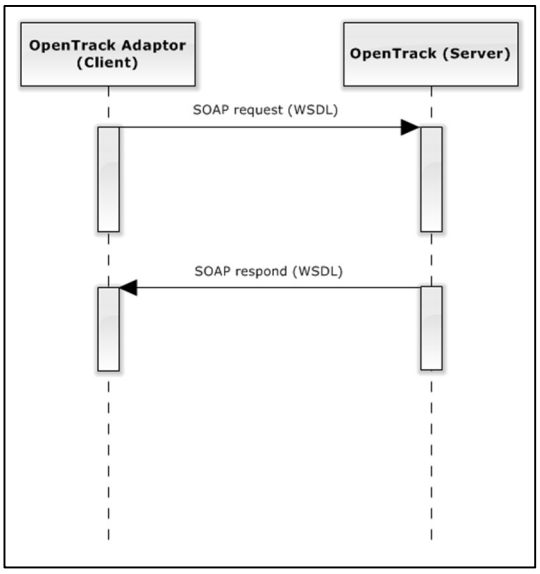


Figure 39: OpenTrack server and adaptor communicating through SOAP.

6.6 TMM – Time Management Module

The time synchronisation is illustrated in Figure 40. The flood simulator `FloodSim` is the triggering simulator that generates events every ΔT time. In the current configuration, ΔT is set to 15 minutes. The term "15 minutes" means simulation time, i.e. the real time or wall time needs to calculate the 15 minutes simulation can vary from that. In fact, it is controlled by the scenario executor (see Section 2.5) in a central way. For instance, scenario executor can say run as fast as possible, i.e. no delay will be introduced between different simulation steps.

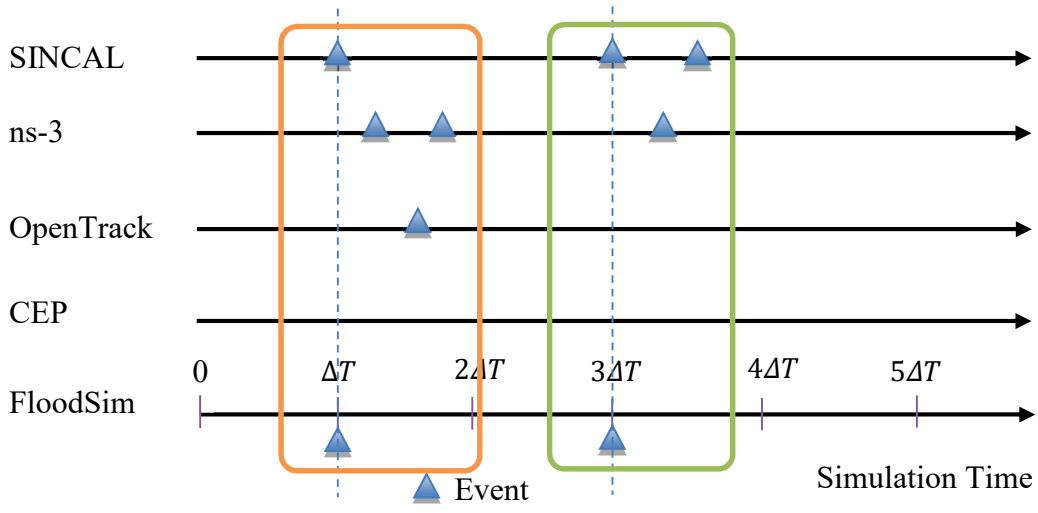


Figure 40: Illustration of the simulator time synchronisation problem

The orange block in Figure 40 shows the one-way dependency from the SINCAL model to the ns-3 and OpenTrack models (see Figure 30). That means the model change in SINCAL results in the model update in ns-3 and OpenTrack. The green block in Figure 40 illustrates the inter-dependency relation between SINCAL model and ns-3 model. That means changes in SINCAL model results in the update of ns-3 model, which further triggers the model update in SINCAL. A concrete example with detailed sequences is illustrated in Figure 41. All these should be synchronised with each other.

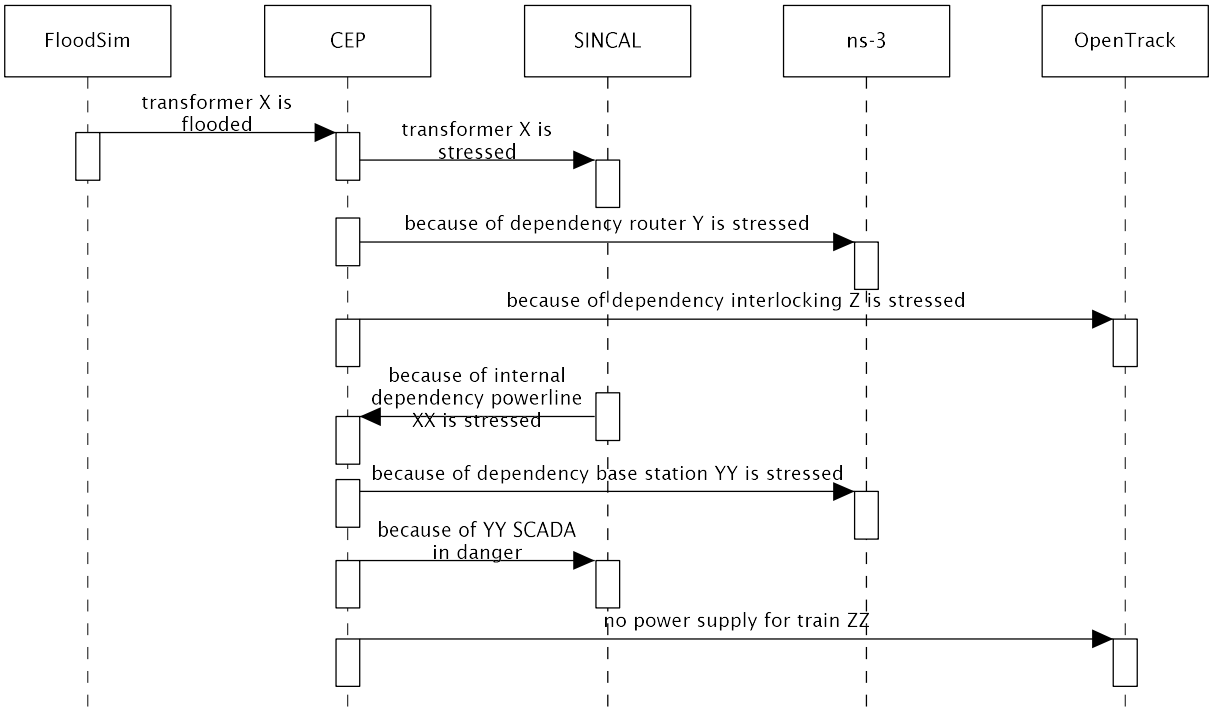


Figure 41: Sequences of dependency and inter-dependencies execution with CEP

6.7 CI State Database for the WFS Service

In order to expose the simulation model to the CIPRTrainer, a facade database is developed that aggregates all the involved CI models. The WFS/WMS services extract the relevant information at runtime and push it to the CIPRTrainer front-end. The information includes:

- Geospatial information of CI elements like the coordination of a transformer, the polygon of a railway main station or the polyline of a railway track.
- The state information of CI elements like normal, stressed, failed and recovery [CIState].
- Meta-information like the name, a short description of the CI.

The database design is illustrated in Figure 42 as an Entity-Relation diagram in Chen syntax. The design follows strictly the database normalisation form [NORM1,NORM2] to remove redundancy information storage. Redundant information is provided as various database views (without physically materialise it to the physical storage like hard disks) to ease the access from outside. In general there are several entities listed below:

1. The entity `CI_State` denotes the possible states of a CI or CI element. Basically in the implemented database table, the `State` column contains the four states defined in [CIState], i.e. `normal`, `stressed`, `failed` and `recovery`.
2. The entity `CI_Type` contains the domains of CI like electrical network, telecommunication network and railway network.
3. The entity `CI_Element_Type` is about concrete CI elements like a transformer in the electrical network or a router in a telecommunication network. It is different than the `CI_Type` entity. A `CI_Type` can contain multiple types of `CI_Element_Type`. For instance, in electrical network – as CI, there exist transformers, sub-stations, power poles, etc.

4. The entity `CI_Element_Point` models the real instance of the CI elements. It includes the name of the CI element, a short description, the element types and most importantly the state information and the geo-location. Current design of the database distinguishes CI elements with geometry type `POINT`, `POLYLINE` and `POLYGON`. The reason for this kind of different handling lays in the efficient modelling capability provided by PostGIS, which is used in the database system to handle geospatial objects. In order to efficiently query and store different kinds of spatial object, the types must be provided during schema generation phase. In Figure 42, only the CI element with geometry type POINT is illustrated. The SQL code generate this table is provided as follows:

```
create table cielement.ci_element_point (
  id integer default nextval('cielement.seq_ci_element'),
  name varchar(128) not null,
    -- different elements may have the same name
  description varchar(1024),
  ci_element_type_id integer,
  ci_state_id integer default 1, -- default normal
  geom geometry(point, 4326),

  primary key (id),
  foreign key (ci_element_type_id) references
 cielement.ci_element_type (id),
  foreign key (ci_state_id) references cielement.ci_state (id)
);
```

The line highlighted with orange colour shows the geometry type is POINT. This kind of specification enables a more efficient processing of spatial objects in PostGIS.

5. The entity `CI_Element_State` is a weak entity and is implemented in the database as a database view. It combines all the information and provides it to other components like the WFS services. This view is defined as follows:

All entities have an attribute `ID` to uniquely identify any instance of that entity.

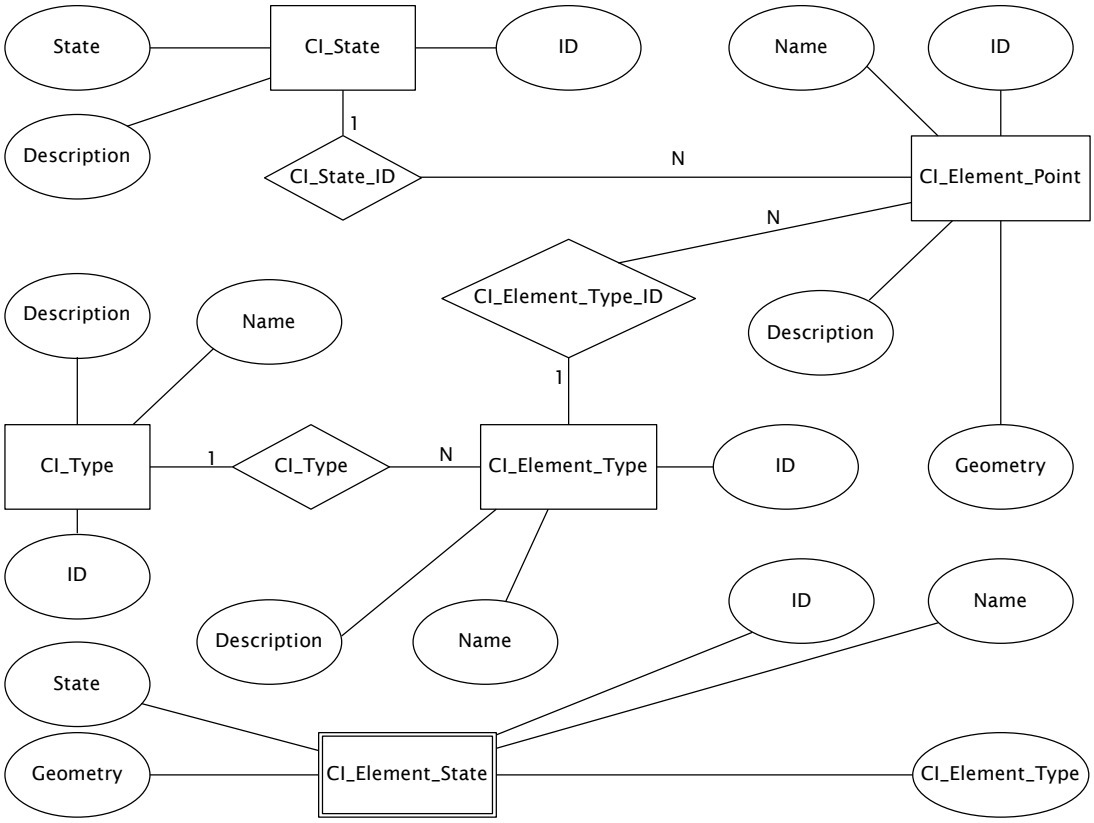


Figure 42: Entity-Relation (ER) Diagram of the CI element state database

6.7.1 Example of the CI state database tables

In this section, an example database of CI states with the materialised tables is presented.

Table 3: CI State table

ID	State	Description
1	Normal	A CI element works as expected
2	Stressed	An element works but does not provide sufficient QoS
3	Failed[12]	The affected element does not work any more
4	Recovery	The element starts to work after a failure

Table 4: CI Type table

ID	Name of type	Description
1	Power	The electrical network
2	Telecommunication	The telecommunication network
3	Railway	The railway network

[12] In the original paper, it is called “crisis”

Table 5: CI Element Type table

ID	Name of type	Description	CI type
1	Transformer	Both two and three winding transformers	1
2	Powerline		1
3	Substation		1
4	Router		2
5	Base station		2
6	Main station		2
7	Track	The railway track	3

Table 6: CI Element Point table

ID	Name	Description	Type	State	Geometry
1	Unterwerk Merhoog Transformer	Transforms 110 kV railway transmission power to 15 kV railway traction power	1	1	(51.7488354597647, 6.49879836738078)
2	Station Loewenberg	Transmission network substation in Emmerich	3	1	(51.8335914046541, 6.27877500310703)
3	Station Mittelrhein Transformer 1	380 kV to 110 kV Transmission network	1	1	(51.6466668099744, 6.68069475148655)
4	Station Mittelrhein Transformer 2	380 kV to 110 kV Transmission network	1	1	(51.6466668099744, 6.68069475148655)
5	Huethum-Ost	110 kV Subtransmission Overhead Line from 2 to 4	2	1	
6	Router 2	It's a router	4	1	(51.8307034, 6.2422109)
7	Router 6	Yet another router	4	1	(51.8320054, 6.237606)
8	Router 5	This is an awesome router	4	1	(51.8325755, 6.2403887)

The database view `ci_element_state` is not a materialised view. It is defined as follows:

```
create view cielement.ci_element_state as
  select ce.id, ce.name, cet.ci_type, cs.state, ce.geom
  from
    cielement.ci_element ce
    inner join cielement.ci_element_type cet
      on (ce.ci_element_type_id = cet.id)
    inner join cielement.ci_state cs
      on (ce.ci_state_id = cs.id)
;
```

Basically, it combines the information from different basis tables and provides a more human-friendly information to other system components like the WFS service that can be displayed in the front-end directly.

7 What-if Analysis in CIPRTrainer

One of the novel capabilities provided by CIPRTrainer is What-if Analysis for crisis management training. It is based on the fast rollback of various simulated worlds, which is in general not possible for real-world situations. This section provides an insight about the technical implementation of rollback and the corresponding impact and consequence analysis.

7.1 Federated Simulation-based What-if Analysis

What-if Analysis is based on the rollback support of different software components including the simulators (both domain-specific CI simulators and threat simulators), visualisation module, time management module and spatial objects.

To facilitate the rollback in CI simulators, we decided to use Git – an open source version control system. It supports lightweight branching and extensions can be added into the core Git systems. The basic idea is to dump the model in simulator memory and store it as a file. After that the file will be put into the Git repository and depending on the branches in CIPRTrainer, a new branch in Git will be created if necessary by using the `branch` command.

```
git branch <new_branch>
git checkout <new_branch> # switch to the new branch
```

If rollback operation is required by the users, the history model will be retrieved from the repository by issuing the following command:

```
git checkout <existing_branch> # switch to an older branch
```

To facilitate the rollback in other components like time management, visualisation and spatial information, the spatial-temporal features in the PostgreSQL database management system is used. The specific `range` type is used to provide a high-level representation of temporal objects. A set of temporal tables are defined (as illustrated in Figure 43). This database schema is part of the complete CIPRNet database. It is used to enhance the management of temporal objects inside of the database without pulling the data into application servers. With this approach, plenty of information can be retrieved directly inside of the DBMS with declarative queries, e.g. the following view exposes the action tree for the selected training session:

```
create view fedsim.action_tree as
select t.id tid, b.id bid, a.id aid, a.simtime,
  v.id vid, v.parameter_name pname, v.parameter_value pvalue
from fedsim.training t left join fedsim.branch b on (t.id =
 b.training_id)
  left join fedsim.action a on (b.id = a.branch_id)
  left join fedsim.action_value v on (a.id = v.action_id)
order by v.id
;
comment on view fedsim.action_tree is
'action tree as json for each training'
;
```

Figure 43: Temporal tables for the rollback support in CIPRTrainer

7.2 Impact and Consequence Analysis Module (CAM)

The overall goal for the CAM is to provide the CIPRTrainer users with the capability to understand the broader consequences of their action and inactions during and at the end of a training session.

7.2.1 CAM Mission statement

Consequence analysis goes beyond impacts, as it clarifies the meaning of impacts and the inoperability of critical and non-critical infrastructure for the population and businesses. A complete and detailed consequence analysis of everything is not possible and not desirable. The CAM should focus on the CIPRTrainer user and the information he needs to learn and perform better than before. The CAM is not intended to be a finished product readily usable for wide variety of conditions. Just like CIPRTrainer as a whole, it serves as a working demonstrator for the specified scenarios. But in general it should be possible to adapt the CAM to different scenarios. Therefore it needs to be conceptualised and implemented in a way that allows later modification.

7.2.2 CAM Conceptual framework

This section clarifies the underlying concepts of the CAM.

7.2.2.1 Purpose of the CAM

Aim of CAM is to provide the CIPRTrainer user with an evaluation of the impacts during and at the end of a training session. The evaluation results are stored on the CIPRTrainer database. The CIPRTrainer front-end handles the visualization of these results (see section 2.3 for details).

7.2.2.2 General approach

For the CAM we distinguish between **impact** and **consequence** (see D6.1):

- **Impact** is the direct outcome of an event, for example the destruction of a private house or the reduction/loss of function of an infrastructure.
- An impact has **consequences**, for example the rebuild cost of a private house or the economic losses due to the reduction/loss of the infrastructure function for infrastructure stakeholders.

Impact can be differentiated in **direct** and **indirect** impacts:

- Direct impacts are the direct damages to CI elements or other assets.
- Indirect impacts are the cascading effects.

Consequences can also be differentiated **direct** and **indirect:**

- Direct consequences are directly related to the impact, for reconstruction cost of a flooded building.
- Indirect consequences are indirectly related to the impact, for example the number of homeless persons.

A further differentiation is possible in **CI related** and **other** consequences:

- **CI related** consequences are related to the impacts of an incident on CI, for example the recovery costs for a CI operator, the GDP loss produced by a power outage or the number of households without electricity.
- **Other** consequences are related to the impacts on everything else, for example the reconstruction cost of flooded houses.

Consequence analysis (CA) therefore comprises the estimation and assessment of these types of consequences of impacts. The impacts are estimated in two components of the CIPRTrainer: The federate simulation for flood and CI and the impact module of the CAM for other impacts. In Figure 44: Concept of impact and consequence the approach is shown.

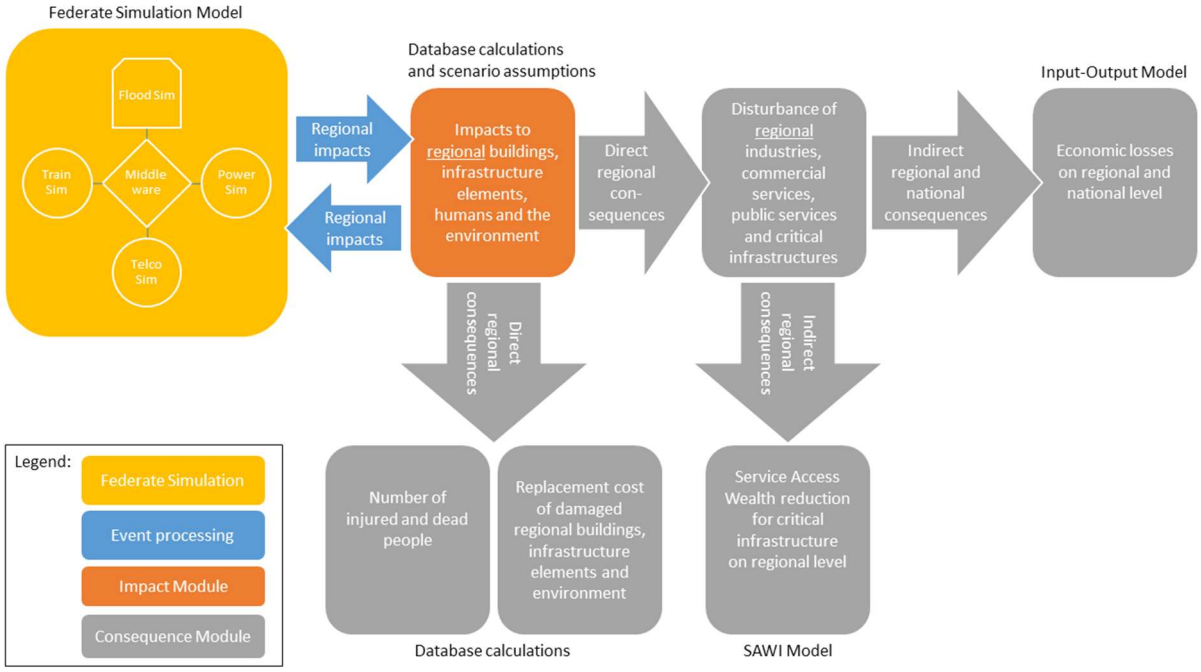


Figure 44: Concept of impact and consequence

The yellow box symbolise the federate simulation with the four different simulation models for Flood, Telecommunication CI, Power CI and Railway CI. All these models are able to estimate cascading effects within their domain. The overarching cascading effects on the regional level are handled through the event processor symbolised as blue double arrow. The impact module is responsible for the calculation of impacts on the regional level, e.g. flood damage to buildings. It uses damage functions for different type of hazards and elements, e.g. flood depth functions for buildings.

The calculated impacts can lead to direct regional consequences in terms of number of injured and dead people, costs for damaged buildings, infrastructure elements and environment. These consequences are calculated by the CAM. Additionally there are indirect regional and national consequences resulting from the direct consequences: the disturbance of regional industries, commercial services, public services and the service of (critical) infrastructures. To assess these indirect consequences different measures are possible. A discussion of different possible options is in the section 7.2.2.7. For the CAM we have chosen input-output modelling to analyse the regional and national effects on the economy which is elaborated in section 7.2.2.7.2.

7.2.2.3 Geographical dimension of the analysis

The CAM takes the geographical dimension of the impacts and consequences into account. For the CAM it is important where an impact has happened and where the consequences oc-

cur, which is not necessary the same area (cascading effects and indirect consequences). In the CAM we use two-dimensional grids to locate people and the built-up area (buildings, infrastructure and environmental areas). These grids are INSPIRE compliant.[13] For Germany we use a 1km*1km grid (see Figure 45), for the Netherlands a 500m*500m grid (see section 7.2.3.2 for details).

Figure 45 Area of Emmerich with German 1km*1km grid and power nodes and lines (map: OpenStreetMap)

For the use of these grids in the CAM one important assumption was necessary: If a hazard has an impact on a grid cell, the whole grid cell is affected, e.g. if the cell is only partly flooded the assumption is that the whole cell is flooded. This assumption is necessary as we have no information about where in the cell the resident or buildings are located. This can lead to an overestimation of the consequences. With more detailed data it would be possible to relax this assumption.

7.2.2.4 Concept for direct impacts

The technical federate simulators are only able to provide impacts and cascading effects for their domain (electricity, telecommunication and train traffic) and the flood simulator does not produce any impacts by itself, it only calculates the geographical extent of the flood with attributes for height and rise rate. So we needed a separate impact module for the flood impacts and all other impacts of the different scenarios. It has to be noted that some impacts are not calculated; instead they are part of the storyline and therefore predefined. An example is

[13] The INSPIRE directive (Infrastructure for Spatial Information in the European Community) aims to create a European Union (EU) spatial data infrastructure. This will enable the sharing of environmental spatial information among public sector organisations and better facilitate public access to spatial information across Europe. http://inspire.ec.europa.eu/

the train derailment in the Emmerich Scenario, where the amount of damage to humans and buildings form the train crash is defined in the storyline.

7.2.2.4.1 Concept for direct impacts on humans

Impacts on humans can lead to injuries and death. For operationalisation mortality functions can be used. We based our approach on a general framework for loss of life estimation from Jonkman et al. 2010 [JONK2010]. Basis principle is to look at the exposed individuals to a certain hazard. If the people are informed they can shelter (i.e. keep the door and windows closed when a chemical cloud is coming, going upstairs in a flood etc.) If they cannot shelter nor self-evacuate, they are ex posed to the threat. Emergency forces can evacuate them if present (depends on trainee decision), otherwise they are exposed until the end of the threat. The effects of the impact on the exposed people are calculated with hazard specific mortality functions. The more intense an impact is (e.g. high flood depth and rise speed of water during a flood; time of day, during the night or during rush hour) the more casualties are to be expected.

The inherent mobility of humans brings some conceptual issues. Usually residential data is used to assess impact on humans. But this leads to an overestimation of impacts on residential areas in the daytime, as normally a big part of the residents is at work (or school, university) or pursue other activities (shopping mall). There are different solutions discussed in the literature. More static approaches use a simplified binary distinction between daytime and night time distribution (see [FREI2012] and [Leung2010]). Others try to develop models of dynamic behaviour of residents, which is a more difficult task (see [Polese2014]). The CRISMA project has developed a model based on the assumption, that it is possible to redistribute population data according to time and local parameters. To do this, data about the dynamic change of population locations is necessary. The model distinguishes between workplace, commuting and home. Areas that are primary business or commercial sectors can be used as workplace location and residential areas as home location. Transport networks are used as commuting locations. Important for the redistribution is the how many people are redistributed to a specific area. For residential areas population density can give guidance. For transport networks traffic information system could be useful to identify heavy traffic connections. Workplaces are much harder to estimate. To distribute the workforce detailed workplace data is needed.

Regarding CIPRTrainer scenarios the data available are not sufficient for the dynamic modelling of the population. Then, a simplified approach is used considering only residential data. Anyway, a dynamic population model would be desirable.

7.2.2.4.2 Concept for direct impacts on buildings, infrastructure elements and environment

The impacts on buildings, infrastructure elements and environment are conceptualised in a similar way to impacts on human. First these objects need to be physically exposed to a hazard, e.g. a house must be in the flooded area. Second the object must be vulnerable to the hazard. The damage depends of the intensity of the hazard (e.g. flood depth) and the degree of sensitivity of the object to the specific threat (e.g. the main material of the building: wood vs. brick). Some damage functions for specific threats and specific objects are available in the literature. For flooding we could draw upon the ‘Standard Method 2004 Damage and Casualties Caused by Flooding’ from the ‘Ministerie van Verkeer en Waterstaat’ [Kok2004] of the Netherlands and the book ‘Hochwasserschäden’ [Thieken2010] for Germany. But not for all types of objects and hazards are damage functions readily available. In these cases we have made assumptions on the basis on available damage functions. In Figure 46 is an example of a flood damage function for low-rise dwellings from [Kok2004].

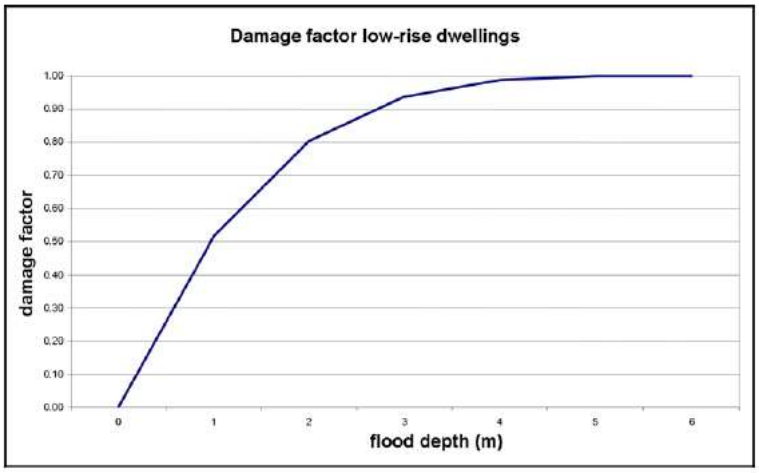


Figure 46: Example of a flood damage function for low-rise dwellings (source [Kok2004])]

7.2.2.5 Concept for indirect impacts

Indirect impacts are the cascading effects through dependencies in a critical infrastructure and interdependencies between critical infrastructures. This is done by the federate simulators and the event processor, see sections 5 and 6. Information of status changes on infrastructure elements are changed in the CAM database by the database adapter see section 2.4.

7.2.2.6 Concept for direct consequences

7.2.2.6.1 Direct consequences on humans

For the CA we decided to only count the number of injuries and deaths. As an economic evaluation of life is impossible and would be a highly ethical issue, we refrained from doing so.

7.2.2.6.2 Direct consequences on building, infrastructure elements and environment

To assess the direct consequences on a specific building, infrastructure element or environmental area, we need a metric to express the value of the damage. We decided to use reconstruction cost for this purpose, because information about the potential reconstruction cost of residential, commercial, industrial and public buildings are derivable from official data on build cost in Germany and the Netherlands (see section 7.2.3.3). For infrastructure elements and the environment a variety of data sources could be found (see section 7.2.3.3). To calculate the actual reconstruction cost for a specific object a damage factor is needed. This is conceptualised as a value between 0 and 1, with 0 no damage and 1 total destruction. This damage factor is determined by the impact module (e.g. flood-depth-functions). The actual reconstruction cost of a specific element is defined as a function of the damage factor.

7.2.2.7 Concept for indirect consequences

7.2.2.7.1 Indirect consequences of power outage on regional businesses and households

For the estimation of power outage cost empirical studies are a common method. There are studies which use historical data to derive cost estimates. For example published Lawton et al. [Lawton2003] the results of an analysis of studies on power outages costs in the USA in the 1980th and 1990th. Other authors collect new data through empirical studies with power customers, for example [LaComm2004]. For Germany there are studies available from the "Hamburgisches WeltWirtschaftsinstitut" (HWWI) [Piaszeck2013] and from the "Institute of Energy Economics" at the University of Cologne [Growitsch2013].

For the CAM we build upon the work of HWWI and conceptualise the consequences of power outage as the estimation of the 'value of lost loads' per hour (VOLL) [Piaszeck2013, p. 6-

9]. For businesses HWWI used an indicator that shows how much output is produced by one kWh for a specific sector:

$$VOLL_{business} = \frac{GrossValueAdded}{ElectricityConsumption}$$

With the knowledge about how much electricity is consumed in an hour and the assumption of zero substitutability of electricity, we they estimate the outage cost per hour.

For households it is more difficult to conceptualise. HWWI used welfare gain from electricity-dependent leisure activities as approximation. For the quantification of welfare gain for one hour of leisure the used average net wages per hour as a proxy. The microeconomic optimality condition is that at the margin the benefits of one hour of leisure is equal to the opportunity costs in terms of foregone labour income. As a further assumption they propose that 50 percent of all leisure activities are electricity-dependent.

$$ValueOfLeisure = 0.5 * AverageLeisureHours * NetWagesPerHour$$

$$VOLL_{households} = \frac{ValueOfLeisure}{ElectricityConsumption}$$

The VOLL values for the different districts Germany for 2010 are available in the study [Piaszeck2013 , p. 17 and 19]. We used the values given for the district ‘Kleve’.

7.2.2.7.2 Indirect economic consequences on a regional and national level

There are basically two major streams in economic theory for the consequence analysis of disastrous events: input-output models (IO) and calculable general equilibrium models (CGE). Both modelling approaches address the interaction of the different economic sectors. They differ however in which manner these sectors interact and how the sectors react to external shocks ([Halle2014], p. 43-44; [Okuyama2007], p. 116-118). IO-models focus on the interrelations of production, where a sector needs inputs from other sectors to produce goods. In the basic IO-model prices don’t play any role. On the other hand focus CGE-models on the effects price variations to the supply and demand in the different sectors ([Halle2014], p. 44). The basic IO-model is demand driven. A disaster can therefore only be modelled as reduction of the final demand. This causes a decrease in the production of final goods and subsequent in all dependent sectors who supply intermediate or raw goods for this final goods. A loss of production of a supplier firm due to a disaster has correspondingly to be modelled ‘in reverse’. In newer IO-models this restriction has been relaxed. In contrast to IO-models there are no explicit flows of goods in a CGE model. The economic system is always perfect balanced due to the price mechanism of the markets. A reduction in production capital due to a disaster leads to a decrease of supply and a subsequently to a price increase. This leads in turn to a demand reduction and to a new equilibrium. Moreover in CGE-models production factors can be substituted in short term. This induces a fast adaption of firms to mitigate the disaster effects. CGE models are thus more optimistic than IO-models, where production technologies are fixed in the short term. One limitation of both approaches is the high aggregation level. Sectors are the main “economic actors”. If one sector suffers from a disaster, all businesses aggregated in this sector suffer the same consequences, regardless of spatial location. There are no distinct production functions and no explicit supply chains modelled in the sector. ([Halle2014], p.44-46; [Okuyama2007], p. 116-118; [Rose1995]).

For the assessment of indirect economic consequences we decided to use input-output (IO) modelling. We preferred to have explicit flows of goods that could be interrupted, which is far easier to understand for a trainee than the abstract price mechanisms in the CGE models. Additionally, for training purposes the more pessimistic results of the IOM lead to more distinct short term evaluations of different courses of action.

In the following a brief overview on the IO model basics and a simple example is given.

IO model basics

The IO model was invented by Wassily Leontief. He used product flow data to construct transaction matrices, which enables the analysis of the interrelations of sectors in an economic system ([Leontief1953]; [Leontief1956]).

IO-models focus on the interrelations of production, where an economic sector needs inputs from other economic sectors to produce goods. With the help of the method of IO-modelling the effects of business interruption due can be estimated. Impacts on buildings and infrastructure can lead to a reduced output of certain economic sectors. This leads to a decrease in the production of final goods and subsequent a reduction in all dependent sectors. The dependencies are operationalised as coefficients. Input-output tables are available for Germany from the German statistical office and for the Netherlands from the Central Bureau for Statistics (CBS). The input coefficients are directly available for Germany, for the Netherlands they have to be calculated from table data (see example for technology matrix below).

A simple example

In the following the method of IO-modelling will be explained with the help of a simple example. Table 7 shows a simple input-output table of a two 2 sector economy. Central element is the transaction matrix which is highlighted in grey colour. Every column indicates how much intermediate input from Sector 1 and Sector 2 is needed to produce the total output of the sector. Sector 1 needs intermediate inputs with the value of 30 of Sector 1 and intermediate inputs with the value of 150 from Sector 2. Additionally primary inputs (labour, capital etc.) with the value of 120 are needed to produce sector 1 goods with a value of 300.

Table 7: Example two sector economy

Input-output table	**Buying sectors**			
Selling sectors	**(Column1) Sector 1**	**(Column 2) Sector 2**	**Final demand**	**Total utilisation**
Intermediate inputs				
(Row 1) Sector 1	30	220	50	300
(Row 2) Sector 2	150	100	250	500
Primary inputs	120	180		
Total production	300	500		GDP = 800

These interrelations can be mapped as technology matrix A where a_{ij} are the input coefficients:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} \end{pmatrix}$$

The input coefficients represent the percentage of the cost of input factor i of the total cost of sector j.

For the example two sector economy the total output vector X and the intermediate input values matrix Z and the final demand vector C are:

$$X = \begin{pmatrix} 300 \\ 500 \end{pmatrix}$$

$$Z = \begin{pmatrix} 30 & 220 \\ 150 & 100 \end{pmatrix}$$

$$C = \begin{pmatrix} 50 \\ 250 \end{pmatrix}$$

The technology matrix A is:

$$A = \frac{z_{ij}}{x_j} = \begin{pmatrix} 0.10 & 0.44 \\ 0.50 & 0.20 \end{pmatrix}$$

The calculation of A for the example economy is also shown in Table 8.

Table 8: Input coefficients in the two sector example economy

Input Coefficients	Sector 1	Sector 2	Demand	Total Utilisation
Sector 1	30/300 = 0.1	220/500 = 0.44	50	300
Sector 2	150/300 = 0.5	100/500 = 0.20	250	500
Primary Inputs	120/300 = 0.4	180/500 = 0.36		
Total Production	1.0	1.0		GDP = 800

These concepts are usually formulated as:

$$X = AX + C$$

If this equation is rewritten with an identity matrix I it enables the assessment of the direct and indirect effects of changes in the final demand vector through the inverse coefficients:

$$C = X(I - A)$$

This is called the 'Leontief inverse':

$$X = (I - A)^{-1}C$$

For the two sector example economy the inverse coefficients are:

$$(I - A)^{-1} = \begin{pmatrix} 1.60 & 0.88 \\ 1.00 & 1.80 \end{pmatrix}$$

The required total production values for the specified demand are:

$$X = (I - A)^{-1}C = \begin{pmatrix} 1.60 & 0.88 \\ 1.00 & 1.80 \end{pmatrix}\begin{pmatrix} 50 \\ 250 \end{pmatrix} = \begin{pmatrix} 300 \\ 500 \end{pmatrix}$$

The inverse coefficients enable the analysis of demand variations. The decrease of the demand in the example two sector economy (due to a production interruption because of a large disaster) by 30 units for sector 1 and 50 units for sector 2 has the following effect on the total production X:

$$X = \begin{pmatrix} 1.60 & 0.88 \\ 1.00 & 1.80 \end{pmatrix}\begin{pmatrix} 20 \\ 200 \end{pmatrix} = \begin{pmatrix} 208 \\ 380 \end{pmatrix}$$

With the new production values and the fixed coefficients the new input values can be calculated in the input-output table. The decrease of the demand leads to a total decrease of the GDP value from 800 to 588 (see Table 9).

Table 9: Example of a demand decrease

Calculation of the input values	Sector 1	Sector 2	Demand	Total Utilisation
Sector 1	0.1 * 208 = 20.8	0.44 * 380 = 167.2	20	208
Sector 2	0.5 * 208 = 104.0	0,20 * 380 = 76	200	380
Primary Inputs	0.4 * 208 = 83.2	0.36 * 380 = 136.8		
Total Production	208	380		GDP = 588

Caveats

The basic input-output model bases on the Leontief production function, which has several properties:

- Linear production function
- One good per sector
- Fixed input coefficients
- No input substitutions

These assumptions limit the explanatory power of the basic model, especially for longer time periods. However in the short term directly after a disaster the assumptions can be an acceptable approximation. In the course of time many variations and extensions of the basic model were proposed in the literature (e.g. inoperability IOM ([Santos2004], [Lian2006]), supply driven IOM ([Smith2010]), but we decided to start with the basic model as it is the least data hungry and easiest to understand for the CIPRTrainer user. In the future an enhanced IOM could improve the explanatory power if needed.

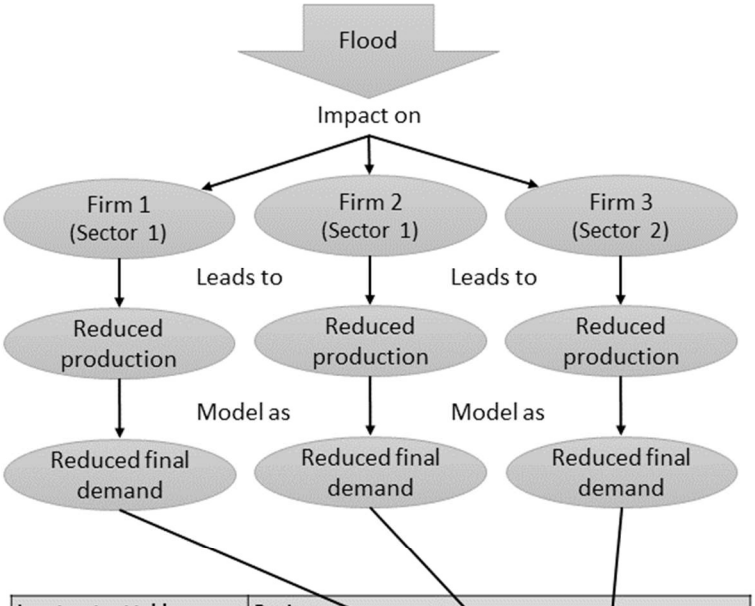
Concept for using IOM in the CAM

The method of IOM is used to calculate the indirect effects of the disturbance of economic sectors in the CAM. In Figure 47 the logic is shown by an example. A flood leads to impacts on local business firms of two different sectors. This impacts lead to a reduced production. In IOM we have to model the reduced production as reduced final demand. Due to the interdependencies in the input-output-model the effect on the intermediate production and the total production of all sectors can be calculated. The results depend on the seriousness of the impact and which sectors are affected.

Obstacles due to lack of data

One obstacle for the use of IOM in the CIPRTrainer is the lack of regional input-output data. There a proposals in the literature how to regionalise national data (see [Flegg2013], [Kowal2012], [Kowal2013], [West1994]), but using one of these methods is a very complex and time consuming task, hence not feasible in the timeframe of the CIPRNet project. We decided to take a much simpler approach, even if this leads to a less realistic estimation of the economic consequences. Our main goal with the CAM is to give the trainee feedback about his actions; a less precise estimation of indirect economic consequences is therefore acceptable. In the future these simplifications can be replaced by a more precise method. Our approach was to assume that the regional input-output structure is the same as on the national level. The absolute values for the different sectors are proportional to the GDP share of the region. For the district ‘Kreis Kleve’ the GDP share was 1.3% of the national GDP in the year 2011. The IOM uses this ‘regionalised’ IO-table data.

Another obstacle is the lack of output data on business level, i.e. how much a business is producing in one year. We had to refer to a combination of value-added data on the district level and data on the number of firms with a specific NACE-code in the district to generate a dataset on value-added per firm with a specific NACE-code.



Input-output table	Buying sectors			
Selling sectors	Sector 1	Sector 2	Final demand	Total utilisation
Intermediate inputs Sector 1	30 → 20.8	220 → 167.2	50 → 20	300 → 208
Sector 2	150 → 104.2	100 → 76	250 → 200	500 → 380
Primary inputs	120 → 83.2	180 → 136.8		
Total production	300 → 208	500 → 380		GDP 800 → 588

Figure 47: Concept for using IOM

7.2.2.7.3 Indirect consequences as loss of service access

As an alternative to an economic assessment of the consequences our project partner ENEA has proposed an indicator for Service Access Wealth (SAWI). This indicator deviates from the strict monetary appraisal of infrastructure services to a 'wealth' related approach, whereby wealth is used in the sense of wellbeing. A loss of service decreases the wellbeing of citizens.

It is planned to use parts of these concept also in the CIPRTrainer. For concept details see D7.4 chapter 3.

7.2.3 CAM data and data base concept

For the CAM we had to obtain a large amount of data from various sources. In this section we begin with the data and the data sources we had to acquire, followed by the database concept for the storage of the data.

7.2.3.1 Data compatibility

The use of data from various sources leads often to data compatibility issues. This is especially the case if the data is from different years. As the most recent census data for Germany is for the year 2011, we decided to choose 2011 as reference year for all other data. Because of the cross-border approach of the CIPRTrainer we need data from different EU Member states. To get compatible spatial data we based our approach on data that is compatible with the INSPIRE directive of the EU, which aims at the creation of an inventory of standardised spatial

data in the Member states.[14] But it was not always possible to get data from 2011, so in some cases we had to use data from other years.

7.2.3.2 Spatial data

Spatial data is available in very different level of detail, from 'world data' of the UN organisation over national data to the very small scale of single elements. For the CA we had to find a common data level and decided to use the inspire gird data as a base. Other data with different resolution should be allocated to these grids, e.g. business data that was only available at street level has been allocated to the appropriate grid cells due to nearest neighbour calculations.

7.2.3.2.1 Grid data for Germany

We use data from the national statistics authority of Germany 'Statistisches Bundesamt'. An online platform with public data called DESTATIS is public available. In 2011 German a national census has been held. Some data was also collected on 100m*100m and 1km*1km grids.[15]

Germany has restrictive data protection laws. Microdata has to be altered in such a way that no statistical inference on a single person or household is possible. Due to the fine resolution of the 100m*100m grids a lot data has to be altered to conform to the laws. This leads to an imprecision of the grid data. To reduce this effect we decided to use 1km*1km data for Germany.

We used:

- Census data from 2011 for residents and households in 1km*1km grid (EPSG 3035)
- Census data from 2011 for residential buildings in 1km*1km grid (EPSG 3035)

7.2.3.2.2 Grid data for the Netherlands

For the Netherlands we use data from the national statistics authority of the Netherlands 'Centraal Bureau voor de Statistiek' (CBS). The grid data for the Netherlands is available in 100m*100m and 500m*500m grids. We decided to use the 500m*500m grid as it is nearer on the 1km*1km of Germany.

We used:

- Census data from 2011 for residents and households in 1km*1km grid (EPSG 28992)
- Census data from 2011 for residential buildings in 1km*1km grid (EPSG 28992)

7.2.3.2.3 Other spatial data

We had access to a derived spatial business dataset on street level. The data set was derived from different databases from commercial data providers:

- Deutsche Post Daten
- NAVTEQ (HERE)
- Microm consumer marketing

From the derived dataset we extracted the number of firms with specific NACE-Code on street level and allocated these firms to the grid.

---

[14] The INSPIRE directive (Infrastructure for Spatial Information in the European Community) aims to create a European Union (EU) spatial data infrastructure. This will enable the sharing of environmental spatial information among public sector organisations and better facilitate public access to spatial information across Europe. http://inspire.ec.europa.eu/

[15] https://www.zensus2011.de

For data und land-use we use CORINE land cover data from 2006 (see [Corine2006]), which are available free of charge.

Infrastructure data with geographic coordinates could be derived from OpenStreetMap (OSM) It has to be mentioned that this data is far from complete as it depends on OSM users to insert the data sets. Some regions are more detailed than other regions. But for the purpose of CIPRTrainer it was sufficiently complete for the district of Emmerich.

7.2.3.3 Non-spatial data

In addition to spatial data we needed non-spatial data. This comprises:

- cost data
  - reconstruction cost for buildings
  - reconstruction cost for infrastructure elements
  - reconstruction cost for environment
  - cost of emergency force actions
  - cost on value of lost loads due to a power outage
- macro-economic data
  - national input-output tables
  - input coefficients
  - data on number of firms on district level
  - data on value-added on district level

7.2.3.4 Database concept

The CAM database concept is based on the INSPIRE compliant grid datasets from Germany and Netherlands. Every grid cell contains data on the geolocation it covers and some meta-information. As the data-tier of CIPRTrainer system constitutes a relational database we developed a relational database schema, which is shown in Figure 48. The figure shows an ER-diagram with a Chen notation. On the left side of the diagram is the `grid_cell` entity with its attributes (i.e. columns of the table). All other entities on the right side of the diagram are related to the `grid_cell` entity, i.e. a grid cell contains `humans`, `residential` and `business` buildings, `infrastructure` elements and one or more `environments`. This is expressed as 1-to-n relation. All these related entities have attributes itself. Some of them have an additionally n-to-1 relation to one other entity. There is also an entity for emergency force actions, which is unconnected to the other entities.

The details of these entities are shown in the table on the following pages. The tables have a similar structure:

- Name of the attribute
- Data type, e.g. integer or text
- Column `Fix/Var` gives information about the possibility to alter the value during a training
  - `Fix`: The value is fixed and cannot be altered during a training
  - `Var`: The value is variable and can be altered during a training
- Explanation
- Value range

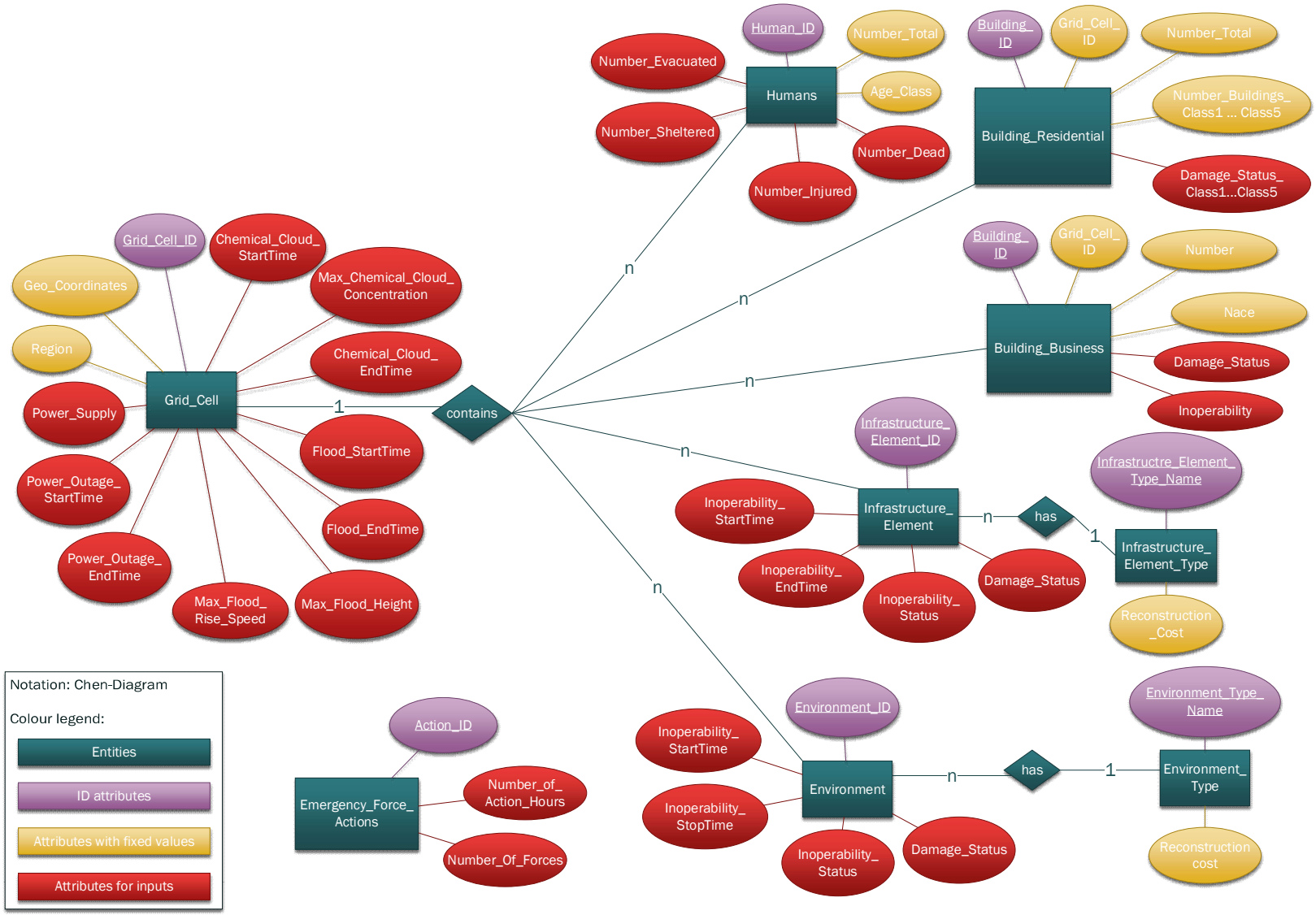


Figure 48: CA database diagram

Table 10: grid_cell

grid_cell				
Name	**Data type**	**Fix/Var**	**Explanation**	**Value range**
power_supply	boolean	var	Power supply	yes/no
region	text	fix	Name of the region	-
gitter_id_1km	text	fix	Official INSPIRE grid cell id	-
grid_cell_id	serial	fix	Internal data base id	-
max_flood_height	numeric(4,2)	var	Maximum height of the flood	-
power_outage_starttime	timestamp	var	Start time of the power outage	-
power_outage_endtime	timestamp	var	End time of the power outage	-
chemical_cloud_starttime	timestamp	var	Start time of the chemical cloud	-
chemical_cloud_endtime	timestamp	var	End time of the chemical cloud	-
flood_starttime	timestamp	var	Start time of the flood	-
flood_endtime	timestamp	var	End time of the flood	-
max_flood_rise_rate	numeric(4,2)	var	Maximum rise rate of the water	-
x_mp_1km	text	fix	Centre point in ETRS89-LAEA Europe (EPSG: 3035).	-
y_mp_1km	text	fix	Centre point in ETRS89-LAEA Europe (EPSG: 3035).	-
max_chemical_cloud_concentration	numeric(3,2)	var	Maximim concentration during the chemical cloud	0 to 1
geom	geometry	fix	Centre point of the grid cell (EPSG: 4326)	-

Table 11: human

human				
Name	**Data type**	**Fix/Var**	**Comment**	**Value range**
number_total	integer	fix	Number of humans	-
number_evacuated	integer	var	Number of evacuated humans	-
number_sheltered	integer	var	Number of sheltered humans	-
number_dead	integer	var	Number of dead humans	-
number_injured	integer	var	Number of injured humans	-
human_id	serial	fix	Internal data base id	-
grid_cell_id	serial	fix	The grid cell the humans reside	-
gitter_id_1km	text	fix	Official INSPIRE grid cell id	-

Table 12: building_business

building_business				
Name	**Data type**	**Fix/Var**	**Comment**	**Value range**
building_id	serial	fix	Internal data base id	-
grid_cell_id	serial	fix	Internal data base id	-
nace	text	fix	NACE code of the firm who owns the building	-
damage_status	numeric(3,2)	var	Damage status of the building 0 means unharmed 1 means complete destruction	0 to 1
inoperability	numeric(3,2)	var	Inoperability of the production 0 means full production 1 means no production	0 to 1
gitter_id_1km	text	fix	Official INSPIRE grid cell id	-

Table 13: building_residential

building_residential				
Name	**Data type**	**Fix/Var**	**Comment**	**Value range**
buildings_id	serial	fix	Internal data base id	-
grid_cell_id	serial	fix	Internal data base id	-
gitter_id_1km	text	fix	Official INSPIRE grid cell id	-
number_buildings_class1	integer	fix	Number of buildings with 1 housing unit	-
number_buildings_class2	integer	fix	Number of buildings with 2 housing units	-
number_buildings_class3	integer	fix	Number of buildings with 3-6 housing units	-
number_buildings_class4	integer	fix	Number of buildings with 7-12 housing units	-
number_buildings_class5	integer	fix	Number of buildings with >= 13 housing units	-
damage_status_class1	numeric(5,4)	var	Damage status of buildings with 1 housing unit	0 to 1
damage_status_class2	numeric(5,4)	var	Damage status of buildings with 2 housing units	0 to 1
damage_status_class3	numeric(5,4)	var	Damage status of buildings with 3-6 housing units	0 to 1
damage_status_class4	numeric(5,4)	var	Damage status of buildings with 7-12 housing units	0 to 1
damage_status_class5	numeric(5,4)	var	Damage status of buildings with >= 13 housing units	0 to 1
number_total	integer	fix	Total number of buildings	-

Table 14: infrastructure_element

infrastructure_element				
Name	**Data type**	**Fix/Var**	**Comment**	**Value range**
infrastructure_element_id	serial	fix	Internal data base id	

damage_status	numeric(3,2)	var	1 means comlete destruction	0 to 1
inoperability	numeric(3,2)	var	0 means full function	0 to 1
infrastructure_element_type	text	fix	Type of the infrastructure	
grid_cell_id	serial	fix	Internal data base id	
gitter_id_1km	text	fix	Official INSPIRE grid cell id	

Table 15: environment

environment				
Name	**Data type**	**Fix/Var**	**Comment**	**Value range**
environment_id	serial	fix	Internal data base id	-
damage_status	numeric(3,2)	var	Damage status of the element 0 means unharmed 1 means complete destruction	0 to 1
inoperability	numeric(3,2)	var	Inoperability of the element 0 means full function 1 means complete failure	0 to 1
environment_type	text	fix	Type of the environment	-
grid_cell_id	serial	fix	Internal data base id	-
gitter_id_1km	text	fix	Official INSPIRE grid cell id	-

Table 16: emergency_force_actions

emergency_force_actions				
Name	**Data type**	**Fix/Var**	**Comment**	**Value range**
action_id	serial	fix	Internal data base id	-
number_of_forces_total	integer	fix	Number of forces	-
number_of_action_hours	integer	var	Number of whole hours of action	-

7.2.4 CAM implementation

In the following section the state of implementation of the CAM is outlined. First a quick overview over the implemented functionalities is given, followed by a more in-depth description of the actual implementation in the data base.

7.2.4.1 Summary of the state of implementation by March 2016 and further tasks

The database conceptual schema (see section 7.2.3.4) was implemented in the CIPRTrainer database, i.e. all tables and the relations between the tables were created.

The collected data (see section 7.2.3) for the derailment scenario were transformed and inserted in to the appropriate tables in the data base schema.

The following functionalities are currently implemented:

- Impact and consequences of a chemical could on residents
- Impact and consequences of flooding on residents
- Calculation of reconstruction cost for damaged residential and business buildings
- Calculation of reconstruction cost for damaged infrastructure elements
- Calculation of first responder action cost
- Calculation of direct power outage cost for residents

The following functionalities will be implemented in the next months:

- Impact of flooding on residential and business buildings, infrastructure elements and environment
- Calculation of costs for damaged environments
- Indirect consequences on regional and national level with input-output modelling

Additionally some of the already implemented impact functions will be improved for better accuracy.

For the implementation of the flooding scenario a lot of additional data has to be collected, transformed and inserted in the database, especially for the Netherlands.

7.2.4.2 Implementation details

In this section some implementation details are outlined.

7.2.4.2.1 Direct consequences

Number of injured and dead humans (total and per grid cell):

Depends on the exposure of the humans and the severity of different threats (e.g. for flooding the severity depends on water depth and water rise speed) or is predefined in the scenario.

Number of damaged residential and business buildings (total and per grid cell):

Depends on the exposure of the buildings and the damage functions for different building types or is predefined in the scenario.

Reconstruction cost for residential and business buildings (total and per grid cell):

The equation for the reconstruction cost of buildings is:

$$reconstruction\_cost = damage\_factor * max\_reconstruction\_cost$$

The `max_reconstruction_cost` for Germany are derived from German property value standard procedures „Sachwertrichtlinie SW-RL 2012".

`Damage_factor` is a value from 0 to 1, with 0 = no damage and 1 = total destruction.

This is a linear function. In future version this could be made more realistic by using non-linear functions for the different types of buildings.

Reconstruction cost for train infrastructure (total):

The equation for the reconstruction cost of infrastructure elements is:

$$reconstruction\_cost = damage\_factor * max\_reconstruction\_cost$$

The `max_reconstruction_cost` is derived from a non-public railway construction study for Germany.

`Damage_factor` is a value from 0 to 1, with 0 = no damage and 1 = total destruction.

This is a linear function. In future version this could be made more realistic by using non-linear functions for the different types of infrastructure element types.

7.2.4.2.2 Indirect consequences

Cost of first responder actions (total):

The equitation for the cost of first responder actions is:

$$cost\_of\_emergency\_actions = number\_of\_personell * hours\_of\_action * cost\_per\_hour$$

The `cost_per_hour` is derived from the standard rate for firefighters in Emmerich am Rhein.[16]

Evaluation of power outages for households:

The equation for the value of lost loads for households due to power outage (per grid cell) is:

$$voll\_households = number\_of\_affected\_housholds * average\_voll\_per\_hour\_district\_kleve$$

The `average_voll_per_hour_district_kleve` is derived from studies about German power usage [Piaszeck2013].

The equation for the total cost of power outage for households (total and per grid cell) is:

$$total\_cost\_power\_outage = voll\_households * duration\_of\_power\_outage$$

7.2.4.3 In-database analytics

For the CAM a separate database schema is used in the CIPRTrainer database (see 2.4 for the CIPRTrainer database).

The calculations are implemented as SQL-queries and functions direct in the database. This enables the full use of the in-build data analytic optimisations of the database, and leads to faster calculations and less traffic to the other CIPRTrainer components.

The basic concept is to create a view for every specific consequence in the consequence analysis. The views in the CAM serve as storage for queries and as result tables. They can be easily accessed by the CIPRTrainer front-end to visualise the CAM results for the user. The views update automatically (running the queries) when they are accessed. An example for the number of damaged business buildings is the following query:

```
CREATE OR REPLACE VIEW
 ca.result_buildings_business_number_damaged_total AS
 SELECT sum(building_business.number) AS sum
   FROM ca.grid_cell
     JOIN ca.building_business ON grid_cell.grid_cell_id =
   building_business.grid_cell_id
     JOIN ca.nace ON building_business.nace::text =
 nace.nace::text
  WHERE building_business.damage_status > 0::numeric;
```

Additionally data base functions were written for specialised tasks. These task are the calculation with mortality functions for chemical clouds and flooding, and the cost of first responder actions. An important function is the RESET function, which will reset the columns with variable data fields in the database tables to the standard values at the beginning of a training session.

The changes in the CAM data base through user actions or the scenario storyline are done by the data base adapter (see 6.7). If the Trainee chooses to see the CA (by pushing the CA button in the GUI) the CIPRTrainer front-end will access the specified views and visualise the

[16] "Tarif zur Feuerwehrsatzung der Stadt Emmerich am Rhein", https://www.emmerich.de/

results of the CA (see 2.3). These concepts are depicted in the Figure 49. The red arrow path shows the interaction in case of a storyline event, the blue shows the execution of a mitigation action and the green the request of a CA.

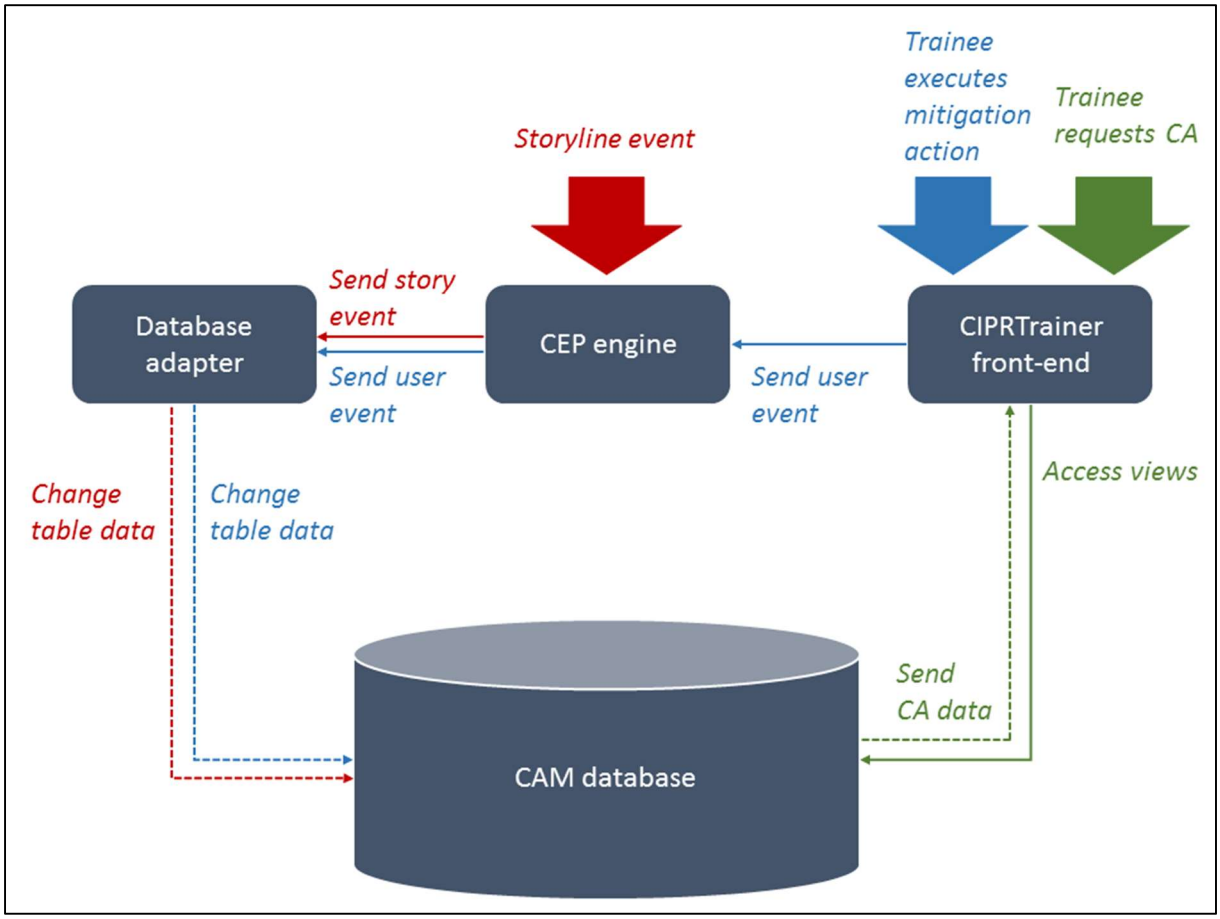


Figure 49: CAM database interaction with other components

7.2.5 Possible future modifications

The CAM is, like the CIPRTrainer at a whole, a prototype to show the possibilities of such a system. In the future this prototype could be enhanced by:

- Other Hazards (Storms, earthquakes, flood surge)
- Dynamic population model
- Enhanced input-output model
- Availability of more detailed data, especially for businesses and buildings

8 Conclusion

This deliverable D6.4 provides a comprehensive description of the implementation details of CIPRTrainer, an innovative application that provides capabilities for What-if Analysis and for exploring different courses of action. CIPRTrainer focuses on crisis management training. The conceptual architecture for CIPRTrainer has been described in deliverable [CIPRNetD61]. The basis for the scenario modelling activities was the deliverable [CIPRNetD62]. Relevant models of Critical Infrastructures (CI) were elaborated in the deliverable D6.3 [CIPRNetD63]. In addition, several refined CI models were supplied in this document (see Section 6), as announced in the previous deliverable D6.3.

The technical implementation of CIPRTrainer consists of several core components:

- ***Scenario management.*** It includes the creation, conversion and execution of CI-centric crisis management scenarios with an emphasis on dependency modelling of CI. Scenarios are created within SyMo in an offline fashion. The results can be exported as SDL, which is the Scenario Description Language developed for modelling crisis management scenarios. Scenario Executor, which is part of the scenario management components, can import and execute SDL.
- ***Declarative dependency handling with Complex Event Processing.*** Cascading effects caused by sophisticated dependencies between CI are deduced by executing the declarative rules encoded in EPL – Event Processing Language. The open source event-processing engine Esper has been adopted to interpret the rules. It has been seamlessly integrated into the CIPRTrainer backend.
- ***Federated simulation system with models, simulators and adaptors.*** It provides the technical basis for performing rollback and What-if Analysis, since in real-world context rollback of what has already happened is not possible. The simulation backend consists of three domain-specific CI simulators – SINCAL, ns-3 and OpenTrack – plus one threat simulator, the flooding simulator. For each simulator, a CIPRTrainer adaptor has been developed, enabling the RESTful communication with other components.
- ***Consequence Analysis.*** Crisis management involves performing various kinds of user actions like initiate an evacuation under crisis situations, send a unit of first responders (police, fire fighters, medical emergencies, etc.) to certain location, etc. The impacts and consequences of performing these actions are provided by this module. Technically, it is implemented inside of the database management systems to maximise the system performance.
- ***HTML5 Web front-end for interaction with system users.*** Advanced Web technologies are adopted to provide a pervasive user experience. This includes responsive design that enables an optimal Look&Feel with different browser configuration and mobile devices. HTTP Push technology minimise the delay of event visualisation by avoiding constantly queries the CIPRTrainer backend. In addition, an internationalised user interface makes CIPRTrainer useful for cross-border scenarios.

All these components are loosely coupled with RESTful Web services that provide a high-level of scalability – in terms of both development productivity and system running performance.

Three limitations of the current system still exist, namely

- the incomplete rollback capability,
- the visualisation of the Consequence Analysis and
- the fact that SDL can only be checked on a syntactic level with JSON Schema.

In the near future, the rollback functionality will be further developed and finalised. Together with the Consequence Analysis module, a comprehensive What-if Analysis capability promised in DoW will be fully implemented, such that it can be demonstrated at the forthcoming CIPRNet course in July 2016. Advanced visualisation of Consequence Analysis module is currently under development, with the same target deadline as rollback.

The limitation to syntactic checks will remain for the time being. Semantic checks are an option for future work. Future extensions of the federated simulation system would require the development of adaptors if new simulators would need to be added. This is a design feature of the way we implemented federated simulation (cf. also [DIESIS]). Developing new adaptors requires a deep understanding of the RESTful interfaces.

To maximise system performance, several functionalities will be moved into database management system as extensions to avoid the overhead of network transmission of data, similar to what has already been done in the Consequence Analysis module. Validating SDL including rules and other scenario elements on a semantic level is a challenging task and formal ontology with dedicated Description Logic reasoners can facilitate this task. Impacts and consequences are in fact a function of time, i.e. they change as time evolves. This issue could be addressed as well in future versions of CIPRTrainer. Re-usability of several components in CIPRTrainer is a long-term goal, especially re-using the crisis management scenarios encoded in SDL, as a kind of European scenario database, and the federated simulation environment, as the proposed EISAC, for other similar research projects and even production environments.

9 References

[CIPRNetD61] EU FP7 CIPRNet, Fraunhofer, Deliverable D6.1 – Conceptual design of a federated and distributed cross-sector and threat simulator, 2014.

[CIPRNetD62] EU FP7 CIPRNet, CEA, Deliverable D6.2 – Application Scenario, 2014.

[CIPRNetD63] EU FP7 CIPRNet, Fraunhofer, Deliverable D6.3 – Federate CI Models, 2015.

[CIPRNetD64] EU FP7 CIPRNet, Fraunhofer, Deliverable D6.4 – Implementation and integration of the federated and distributed cross-sector and threat simulator (forthcoming 2016)

[CIPRNetD723] EU FP7 CIPRNet, ENEA, Deliverable D7.2+D7.3 – Database of the DSS with consequence analysis and Assessment and selection of vulnerability and risk estimators, 2015.

[CIPRNetD74] EU FP7 CIPRNet, ENEA, Deliverable D7.4 – Implementation of the DSS with consequence analysis, 2015.

[CIState] Albert Nieuwenhuijs, Eric Luiijf, and Marieke Klaver. Modeling dependencies in critical infrastructures. In Critical Infrastructure Protection II, pages 205–213. Springer, 2008.

[Corine2006] CORINE Land Cover (CLC2006); Federal Environment Agency, DLR-DFD 2009

[DIESIS] EU FP7 Project DIESIS, GA no 212450, www.diesis-project.eu

[DIESIS2] EU FP7 Project DIESIS, Fraunhofer, Deliverable D4.2a "Proof of concept", 2010.

[DoW] Annex I – Description of Work (Annex to the Grant Agreement of CIPRNet)

[EUCensus11] European Commission (EUROSTAT): The Census Hub: easy and flexible access to European census data, ISBN 978-92-79-37803-4, DOI 10.2785/52653, 2014. http://ec.europa.eu/eurostat/de/web/population-and-housing-census/census-data/2011-census

[EWI13] Growitsch, C.; Malischek, R.; Nick, S.; Wetzel, H. (2013): The cost of power interruptions in Germany. an assement in the light of the Energiewende. EWI. Köln (EWI Working Paper, 13/07).

[EsperTech] EsperTech Inc.; Complex Event Processing Framework [http://www.espertech.com] (last access on 24/03/2016)

[Flegg2013] Flegg, Anthony T.; Tohmo, Timo (2013): Estimating regional input coefficients and multipliers. The use of the FLQ is not a gamble. University of the West of England (Economics Working Papers Series, 1302).

[FREI2012] Freire, S.; Aubrecht, C. (2012): Integrating population dynamics into mapping human exposure to seismic hazard. In: Natural Hazards & Earth Systems Sciences 12 (11), S. 3533–3543.

[GA] European Commission, represented by REA: Grant Agreement FP7-312450-CIPRNet

[GDAL] URL: http://gdal.org/1.11/ogr2ogr.html (last access on 03/22/2016).

[Growitsch2013] Growitsch, C.; Malischek, R.; Nick, S.; Wetzel, H. (2013): The cost of power interruptions in Germany. an assement in the light of the Energiewende. EWI. Köln (EWI Working Paper, 13/07).

[Halle2008] Hallegatte, Stéphane (2008): An adaptive regional input-output model and its application to the assessment of the economic cost of Katrina. In: Risk Analysis 28 (3), S. 779–799. DOI: 10.1111/j.1539-6924.2008.01046.x.

[Halle2014] Hallegatte, Stéphane (2014): Natural disasters and climate change. An economic perspective. Chapter 2: Springer Berlin Heidelberg.

[JAN2016] David Scott Janzen. "An Empirical Evaluation of the Impact of Test-Driven Development on Software Quality". Dissertation. University of Kansas, 2006.

[JASM] URL: http://jasmine.github.io/2.4/introduction.html (last access on 03/22/2016).

[JONK2010] Jonkman, S. N.; Lentz, A.; Vrijling, J. K. (2010): A general approach for the estimation of loss of life due to natural and technological disasters. In: Reliability Engineering & System Safety 95 (11), S. 1123–1133. DOI: 10.1016/j.ress.2010.06.019.

[Kok2004] Kok, M.; Huizinga, H.; Vrouwenvelder, A.; Barendregt, A. (2005): Standard Method 2004 Damage and Casualties Caused by Flooding. DWW-2005-009. Ministerie van Verkeer en Waterstaat.

[Kowal2012] Kowalewksi, Julia (2012): Regionalization of national input-output tables. Empirical evidence on the use of the FLQ formula. HWWI (HWWI Research, 126).

[Kowal2013] Kowalewksi, Julia (2013): Regionalization of National Input–Output Tables: Empirical Evidence on the Use of the FLQ Formula. In: Regional Studies 49 (2), S. 240–250. DOI: 10.1080/00343404.2013.766318.

[LaComm2004] LaCommare, Kristina Hamachi; Eto, Joseph H. (2004): Understanding the Cost of Power Interruptions to U.S. Electricity Consumers. Ernest Orlando Lawrence Berkeley National Laboratory.

[Lawton2003] Lawton, L.; Eto, Joseph H.; Katz, A.; Sullivan, M. (2003): Characteristics and trends in a national study of consumer outages costs. 16th Annual Western Conference. Center for Research in Regulated Industries. Online verfügbar unter www.crri.rutgers.edu.

[Leontief1953] Leontief, Wassily (1953): Domestic production and foreign trade. The American capital position re-examined. In: Proceedings of the American Philosophical Society 97 (4), S. 332–349.

[Leontief1956] Leontief, Wassily (1956): Factor proportions and the structure of American trade. Further theoretical and empirical analysis. In: The Review of Economics and Statistics 38 (4), S. 386–407.

[Leung2010] Leung, S.; Martin, D.; Cockings, S. (2010): Linking UK public geospatial data to build 24/7 space-time specific population surface mod-els. GIScience 2010: Sixth international conference on Geographic Information Science. Hg. v. University of Zürich. Zurich.

[LexYacc] Levine, John; Mason, Tony; Brown, Doug (1992): Lex & yacc

[Lian2006] Lian, Chenyang; Haimes, Yacov Y. (2006): Managing the risk of terrorism to interdependent infrastructure systems through the dynamic inoperability input–output model. In: Syst. Engin. 9 (3), S. 241–258. DOI: 10.1002/sys.20051.

[Lizard15] Lizard, 2015: Lizard flood scenario database, see http://flooding.lizard.net

[Mierlo05] Mierlo, M.C.L.M. van, Gudden, J.J. , Overmars, J.M.S. (2005): Delft-FLS Modellen Krefeld – Kampen en Wesel – Gorcum. Rapport Q3859, WL / Delft Hydraulics.

[MUNLV06] MUNLV-NRW et al, (2006): Risicoanalyse für die grenzüber-schreitenden Deichringe am Niederrhein / Risico analyse grensoverschrijdende dijkringen Niederrhein : Phase 1 : Ent-wicklung einer gemeinsamen Untersuchungsmethode /
fase 1 : ontwikkeling ven een gemeenschappelijke method, Aachen u.a.

[NODE] https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-14-04 (last access on 03/22/2016)

[NORM1] Edgar F Codd. A relational model of data for large shared data banks. Communications of the ACM, 13(6):377–387, 1970.

[NORM2] Edgar F Codd. Further normalization of the data base relational model. Data base systems, pages 33–64, 1972.

[NS-3HWWI15] Piaszeck, S.; Wenzel, L.; Wolf, A. (2013): Regional diversity in the costs of electricity outages. Results for German counties. HWWI (HWWI Research, 142).

[NS-3] ns-3 home page http://www.nsnam.org (last access on 16/07/2015)

[Okuyama2007] Okuyama, Yasuhide (2007): Economic Modeling for Disaster Impact Analysis: Past, Present, and Future. In: Economic Systems Research 19 (2), S. 115–124. DOI: 10.1080/09535310701328435.

[OGC] URL: http://www.opengeospatial.org/ogc (visited on 06/07/2015).

[OpenTrack] The OpenTrack Home Page http://www.opentrack.ch (last access on 16/07/2015)

[Petermann10] Petermann, Thomas; Bradke, Harald; Lüllmann, Arne; Poetzsch, Maik; Riehm, Ulrich (2010): Gefährdung und Verletzbarkeit moderner Gesellschaften – am Beispiel eines großräumigen Ausfalls der Stromversorgung. Endbericht zum TA-Projekt. Hg. v. TAB (141).

[Piaszeck2013] Piaszeck, S.; Wenzel, L.; Wolf, A. (2013): Regional diversity in the costs of electricity outages. Results for German counties. HWWI (HWWI Research, 142).

[Polese2014] Polese, M. et al. (2014): CRISMA. Version 2 of Dynamic vulnerability functions, Systemic vulnerability, and Social vulnerability. D4.32., CRISMA-Project.

[REST08] Pautasso, Cesare, Olaf Zimmermann, and Frank Leymann. "Restful web services vs. big'web services: making the right architectural decision." Proceedings of the 17th international conference on World Wide Web. ACM, 2008.

[Rose1995] Rose, A. (1995): Input-Output Economics and Computable General Equilibrium Models. In: Structural Change and Economic Dynamics 6, S. 295–304.

[Santos2004] Santos, Joost R.; Haimes, Yacov Y. (2004): Modeling the demand reduction input-output (I-O) inoperability due to terrorism of interconnected infrastructures. In: Risk Analysis 24 (6), S. 1437–1451. DOI: 10.1111/j.0272-4332.2004.00540.x.

[SINCAL] The SIEMENS PSS® SINCAL Home Page http://www.sincal.de (last access on 16/07/2015)

[Smith2010] Smith, Braeton J.; Vugrinm Eric D.; Loose, Verne W.; Warren, Drake E.; Vargas, Vanessa N. (2010): An input-output procedure for calculating economy-wide economic impacts in supply chains using homeland security consequence analysis tools. Proposed for presentation at the North American Regional Science Council 57th Annual North American Meetings of the Regional Science held November 10-13, 2010. Sandia National Laboratories.

[Sojeva15] Betim Sojeva: Using Web GIS for Designing Added-Value Training Systems for Crisis Managers, Master's Thesis, University Koblenz-Landau, Germany, 2015.

[Stelling03] Stelling, G. S. , Duinmeijer, S. P. A. : A staggered conservative scheme for every Froude number in rapidly varied shallow water flows. International Journal Numerical Methods In Fluids, Vol. 43, pp. 1329–1354, 2003.

[Thieken2010] Thieken, Annegret H. (Hg.) (2010): Hochwasserschäden. Erfassung, Abschätzung und Vermeidung. München: Oekom.

[Tomasz2015] Brian Tomaszewski. Geographic information systems (GIS) for disaster management. Boca Raton, FL: CRC Press, 2015 (cit. on pp. 7–9).

[TDD03] Beck, Kent. Test-driven development: by example. Addison-Wesley Professional, 2003.

[Usov10] A. Usov, C. Beyel, E. Rome, U. Beyer, E. Castorini, P. Palazzari, A. Tofani (2010): The DIESIS Approach to Semantically Interoperable Federated Critical Infrastructure Simulation. *Second International Conference on Advances in System Simulation (SIMUL 10)*, Nice, France, August 22–27, 2010.

[WebSocket] https://www.websocket.org

[West1994] West, C. T.; Lenze, D. G. (1994): Modeling the Regional Impact of Natural Disaster and Recovery: A General Framework and an Application to Hurricane Andrew. In: International Regional Science Review 17 (2), S. 121–150. DOI: 10.1177/016001769401700201.

[WFS] URL: docs.opengeospatial.org/is/09-025r2/09-025r2.html (visited on 03/22/2016).

[WMS] URL: http://www.opengeospatial.org/standards/wms (visited on 03/22/2016).

[WL01] WL|Delft Hydraulics (2001): Manual Delft-FLS, version 2.55. Rapport R3288/R3224.

[Xie15a] Xie, J., Theocharidou, M., Barbarin, Y., Rome, E.: Knowledge-driven scenario development for critical infrastructure protection. In: Critical Information Infrastructures Security - 10th International Workshop, CRITIS 2015, Berlin, Germany, October 5-7, 2015. p. to appear. Lecture Notes in Computer Science, Springer, 2015.