PREDICT

PREparing for the Domino effect in Crisis siTuations.

PREPARING FOR THE DOMINO EFFECT IN CRISIS SITUATION

D5.2 PREDICT-FPT Simulation Engine

Due date: 31/05/2016

Submitted date: 26/05/2016

Document ID: PREDICT-20160524-D5-2

Revision: version 1.0

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

FP7-SEC-2013-607697

Document change log				
Revision	Edition date	Author	Modified sections / pages	Comments
Draft v0.1	08/04/2016	ITTI	ALL	Version consists the initial structure of the document and plan of work
Draft v0.2	22/04/2016	ITTI	ALL	First version of each section completed, initial vision of interfaces provided
v0.3	06/05/2016	ITTI	ALL	Final version of each section, API completed.
v0.4	13/05/2016	ITTI	ALL	Major corrections and new paragraphs added based on the comments from internal review.
v0.5	16/05/2016	ITTI	ALL	Document refinements
v0.6	24/05/2016	ITTI	ALL	Adjustments and enhancements after the external review made by Fraunhofer

PREDICT
PREparing for the Domino effect in Crisis siTuations.

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)	

Internal Review			
Done by	Signature	Date	Remarks
Tomasz Springer		11/05/2016	All of the comments were included in the new version of the document.
Xie Jingquan		20/05/2016	All of the most important comments were included in the final version of the document

Security Assessment			
Done by	Signature	Date	Remarks
Dominique Sérafin	Dominique Sérafin	24/05/2016	No security issue

Ethical Assessment			
Done by	Signature	Date	Remarks
Dominique Sérafin	Dominique Sérafin	24/05/2016	No ethical issue

Quality Assessment			
Done by	Signature	Date	Remarks
Yohan Barbarin	Yohan Barbarin	25/05/2016	Ready for submission

DISCLAIMER

TABLE OF CONTENTS

1. Executive summary

Simulation in the PREDICT – FPT is based on the scenarios, which represent changes of the objects' states in time. Scenario is represented in the form of a tree structure, which is based on the node – scenario step correlation. Every scenario step represents a single node in the scenario. Consequently scenario steps are classified into events and interactions, which are thoroughly described in section 5. PREDICT – FPT simulation is the process of interpreting chronologically ordered events and calculating the future turn of events based on the simulation engine and supported by other PREDICT tools (e.g. SBR) and simulation modules (e.g. rule engine).

The concept of the PREDICT – FPT simulation is to follow a stateless approach, which is partially based on the Discrete Event Simulation concept. Thus there is no continuous simulation being processed, as there is a simulation-on-demand (step by step approach) implemented. Hence the interactions (also referred as 'actions') with the user and external tools are required. Following interactions are distinguished in the PREDICT – FPT simulation process: (i) sending and receiving simulation data from/to external tool, (ii) ad–hoc interactions with the user or user's action, and (iii) execution of a simulation module of the PREDICT – FPT.

Users of the PREDICT – FPT are enabled to control the process of a simulation by exploring a scenario paths (based on the tree structure mentioned above), which are built upon the execution of appropriate simulation mechanisms. Furthermore users of the PREDICT – FPT are enabled to acquire through the same GUI data from different data sources such as external to FPT tools and sensors. That includes data interchange with the Scenario Based Reasoning (SBR), MYRIAD (see D.6.2 [1] and D6.3 [2]), PROCeed (see D.5.1 [3]), which consist of interactions with rule engine and other simulation modules.

Crucial component of the scenario steps is called 'event'. Event represents the change of the object (or many objects) in a particular step of the scenario. Following categories of the events can be distinguished: (i) predefined, (ii) event of the user, (iii) decision of the user, (iv) information from the sensor, (v) information from SBR, and (vi) event generated by the rule engine.

PREDICT – FPT simulation engine interfaces are built upon RESTful services (as described in detail in D5.4) [4]. Supplementation and additional description of the methods implemented in the simulation engine is provided in the document as well (see section 6).

2. Introduction

Development of the PREDICT – FPT simulation engine was one of the crucial elements of the project's software development. Simulation engine is the central software component responsible for distributing, synchronising and processing data interchange between different components of the iPDT. Simulation engine controls sending and receiving information from/to external tools, ad – hoc interactions with the user along with execution of a simulation module of the PREDICT – FPT.

PREDICT – FPT is open for future adjustments and improvements as only own software elements and open – source solutions are being used. Furthermore, described services and simulation modules were designed and implement to enable fast integration with any other additional tool or sensor.

2.1 Aim of this deliverable and relations to other PREDICT deliverables

Aim of this deliverable is to describe the process of implementation of the PREDICT – FPT simulation engine based on the characterized in D5.1 [2] software architecture and provided in D5.4 [4] interfaces with external to FPT tools. Implementation process of the PREDICT – FPT simulation engine is based on the outcomes and conclusions from both Incident Evolution Framework Work Package (WP3) and Decision Support Tools (WP6). Thus design and development of the core element of the PREDICT – FPT software follows the guidelines and recommendations from the following tasks and deliverables:

- T3.1 – D3.1 Methodology for the identification and probability assessment of cascading effects [5];
- T3.2 – D3.2 Methods of threat quantification [6];
- T3.3 – D3.3 Methodologies for the operation time model lay-out and specification [7];
- T6.2 – D6.2 Cognitive analysis report to support decision tools specifications [1];
- T6.3 – D6.3 Multi-Criteria Decisions support report & software [2].

Furthermore this document is a natural consequence of the conceptual work done and explicitly described within the WP4 (D4.1 [9] and D4.2 [10]).

2.2 Structure of this deliverable

The essence of the document is put in the 'Executive summary' section. Furthermore, in the Section 3 of the document the general concept of the PREDICT – FPT simulation is presented. Detailed description of the stateless simulation approach and introduction to the training mode delivers necessary information for the understanding of the scenario tree concept, which is described in section 4. In the section all the crucial aspects of the scenario tree are presented, including scenario elements, tree exploration aspects, roles of simulation tools, terms of scenario modification and schema of stored data.

Detailed description of the PREDICT – FPT simulation and scenario tree concept enables then to understand the scenario steps, based on events and interactions, which are presented in section 5. Lastly, additional to D5.4 specification of simulation engine interfaces are characterized in section 6.

3. General concept of the PREDICT – FPT simulation

Simulation process in the PREDICT-FPT software is used to foresee possible evolution of the crisis situation and its cascading effects. Simulation is based on the scenario. Scenario represents changes in the objects' states in time. Those objects and their states are part of the simulated reality (e.g. electrical grid, power plant, residential area).

Initializing event (e.g. flooding of the power plant) is called **predefined scenario** and is supposed to be prepared before simulation starts. Predefined scenario is presented in the form of the tree structure (this concept is presented in the section 4). In the structure each node is correlated with a particular scenario step. Scenario steps are divided into **events** describing a situation (mentioned above) and **interactions**, i.e. questions to a user or decision to be taken.

Predefined scenario is a starting point for the simulation triggering further calculations. Once simulation is started additional scenario steps are being generated on the basis and as a consequence of the predefined scenario. That includes generating different, alternative course of events of the particular crisis situation. Extending the predefined scenario by with the additional scenario steps is the responsibility of the PREDICT – FPT simulation engine (described in detail further on).

Thus, **simulation is a process of interpreting and processing chronologically ordered events and calculating the future course of events provided by the simulation engine and supported by other PREDICT tools (e.g. SBR) and simulation modules (e.g. rule engine)**.

The result of the simulation is a new scenario. New scenario can be described, as a predefined scenario extended by new events (first, second and further order impacts), which were generated during the simulation process. Thus, simulation result provides information of a possible cascading effect as a consequence of the given crisis situation. One of the main elements of the foreseeing consequences of the given situation is a stateless approach of the simulation, which is described in the following section.

3.1 Stateless approach

The general concept of the simulation is to follow a stateless approach, partially based on the Discrete-Event Simulation (DES) concept. Following DES, each event occurs at a particular instant in time and marks a change of the state in the system [12]. Hence the simulation is being executed by consequent updates of the scenario content. There is no continuous simulation being processed, which could cause synchronization problems between external tools, user and participating actors (e.g. time laps issues). Stateless approach enables to stop and continue the simulation (i.e. forwarding into the next steps) on demand including already processed data.

Simulation engine processes every user's action taken during the simulation and every message received from other iPDT component or simulation module. Thus, based on the available information

respective scenario elements (i.e. scenario steps) are being updated. There are following user's actions, which can be distinguished from the simulation engine perspective:

- **Sending and receiving simulation data from/to external tool** – there are two external tools, which are implemented with PREDICT – FPT. Those are SBR (WP6) and MYRIAD (WP6). All sent/obtained data is included in the executed scenario.
- **Ad-hoc user's actions** – representing the decisions taken by the user in particular moments of the simulation time, i.e. evacuation of a specific area, usage of resources etc.
- **Executing a simulation module of the PREDICT-FPT** – e.g. triggering the rule engine by delivering proper scenario event. Scenario events can be delivered as a consequence of the simulation, could be predefined before the simulation starts or could be added ad-hoc by the user.

3.2 Introduction to the training mode

Simulation engine of the iPDT was initially considered to address both response and preparedness phase of the crisis. Eventually, based on the research results and end-users requirements, the focus is laid on the cold phase of the crisis, which includes training mode of operation. Hot phase of the crisis could be easily addressed however and the operational mode of operation could be further on developed, but only after the training mode is validated and its usefulness is verified.

Training mode gives the user a wide range of liberty in constructing field of action. Predefined scenario can be built upon both existing and non-existing CIs, as well as with highly probable or absolutely unlikely threats. Furthermore user is enabled to manually interfere with the objects' states, as well as with the characteristics of the operational picture.

Within the training mode users will be able to analyse all the foreseen events and simulate alternative scenarios (based on the user's decisions or probabilities) and get insight into their consequences. Hence training mode provides computational 'what-if' analysis, which enables cascading effects detection.

4. Scenario tree of the simulation engine

Scenario in the PREDICT-FPT is designed as a tree structure. Each 'branch' of the tree structure consists of scenario steps. Scenario steps on each of the tree branch represent an alternative course of action[1]. Tree nodes are called **scenario steps**. If scenario step has more than 1 child, it is called a **interaction**. Otherwise scenario step is called an **event**. For detailed tree representation description see D5.4. Example of such structure is presented below though (see Figure 1).

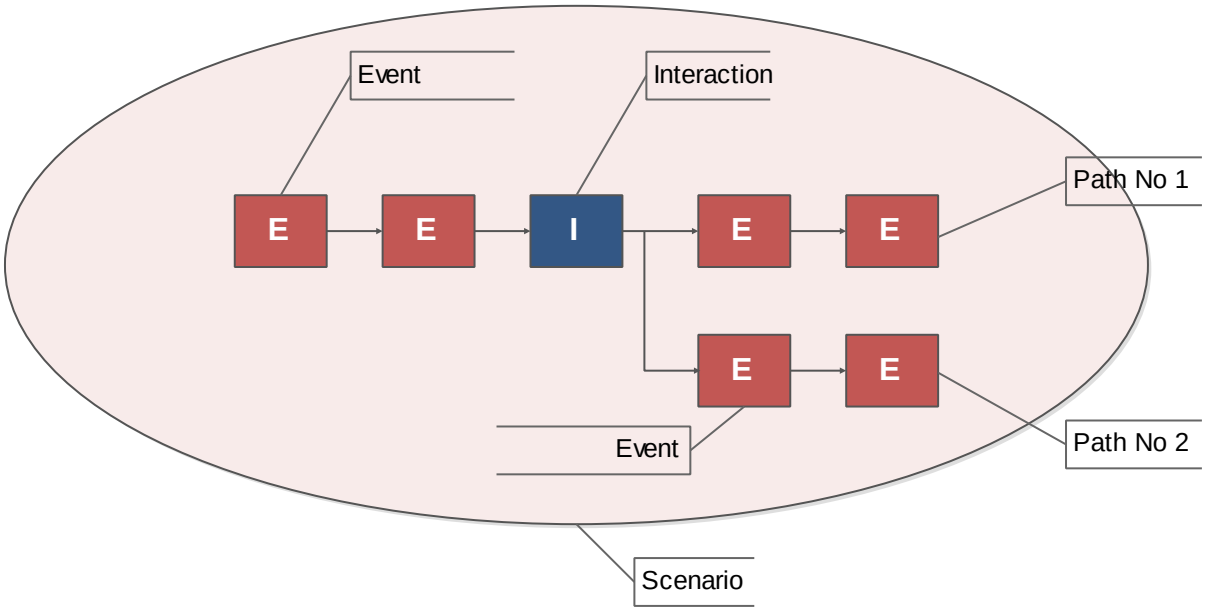

Figure 1 Example of the scenario tree [source: D5.4]

Simulation is triggered by a predefined scenario – a tree of predefined events and interactions (predefined scenario may include more than one path). During the process of the simulation, each time the next scenario step is explored, a predefined scenario may be consistently extended. These extensions can be generated in a result of a communication with particular PREDICT-FPT simulation modules or external data sources.

4.1 Scenario elements

Elements of the scenario can be divided into **direct** and **non-direct part**. Direct elements of the scenario are scenario steps. Non-direct elements are objects, which represent the state of the simulation. State of the simulation differs between various scenario steps. State of the object in

1 Sometimes paths are interchangeably called 'alternative scenarios' in PREDICT documentation

particular event occurrence time represents the current evolution of this object – from the simulation start, through various object's changes caused by events' occurrence.

Scenario's objects can be defined as follows: **Each object** is a tangible, geo-localised physical thing (e.g. building, truck, power plant) or a set of things (electrical grid, water pipes) relevant in a crisis situation**. Objects can group multiple physical objects into an abstract ones represented by a single point, line or polygon on the map**. Basically, they represent something which is submitted for a process of a significant change. Each object is classified into particular category which reflects sectors crucial for cascading effect simulation (e.g. energy, transportation, food and water). Apart from having a unique name, each object possesses a set of attributes, which can be changed during the simulation. These attributes are also exchanged with MYRIAD and SBR as a point of communication.

4.2 Tree exploration aspects

PREDICT user is enabled to control the process of simulation by exploring the scenario tree paths. Hence user decides which path of the tree should be displayed and when next scenario step will be launched (further on referred as 'visited'). Launching a scenario step is the execution of appropriate simulations mechanisms (hereinafter called also 'processing the step') in accordance with the step type and its content. As a result of visiting the particular step, new steps (events of interactions) may be generated, which also can be launched later (i.e. foreseen steps). Every scenario step which represents a single element of currently generated simulation can be called as 'visited' or 'not visited yet' – presented on the Figure 2.

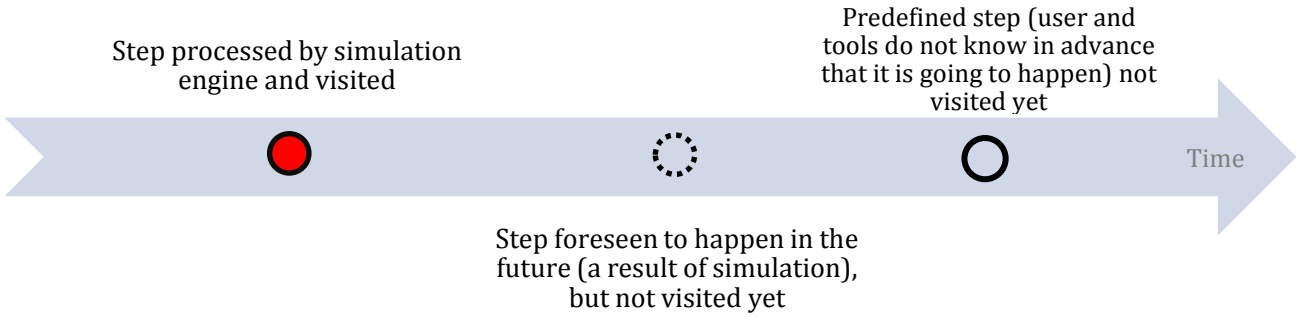

Figure 2 Differences between simulated events

The visited step is coloured in red. Next step, marked with the hash line, is the one which is predicted to happen by the simulation engine. Finally there is a predefined step marked with the solid line, also not visited yet. Predefined step will always happen, no matter what is the current evolution of the simulation. User and other tools taking part in a simulation process expect that foreseen steps will be visited by simulation engine, and are not aware of the existence of predefined steps.

During the exploration of a particular path, user is able to skip to another alternative path and explore it in the same way. Simulation engine remembers which steps were visited on various paths. Hence it is possible to control how was the path was explored (i.e. how many steps were visited).

4.3 Roles of the simulation tools

Data sent by simulation modules included in the PREDICT-FPT and external tools (provided by WP6) – hereinafter called simulation tools – is considered in the same way by the simulation engine. Each simulation tool's proceedings are based on the type of received information about scenario steps and similarly the results produced by a simulation tool are identified in a proper way by the simulation engine to keep the coherence of a scenario.

Main simulation tools, which take part in the simulation process, are:

- **SBR** (Scenario Based Reasoning; developed within WP6 activities) – a tool responsible for generating CI (Critical Infrastructure) cascading effect by performing Bayesian network analysis based on CI dependencies model. SBR is executed by simulation engine when a proper event representing the change of CI state occurs. As a result, SBR provides a set of scenarios, where each scenario includes a list of cascading effect events (representing other CI changes) and is characterised with its likelihood.
- **MYRIAD** (developed within WP6 activities) – a tool responsible for evaluation of simulated scenarios by analysis of contained steps, their relations to concrete sectors (e.g. Economy, Health, Transport, etc.) and importance of these sectors. Then, the result of performed analysis is presented to a user as sufficient information to decide which alternative scenario is optimal for the trained situation. On the basis of this recommendation, user can make decisions. It is also possible to evaluate foreseen steps provided by other simulation tools but not visited yet).
- **Rule Engine** (developed within WP5 activities) – a tool responsible for generating the first, second and further order impacts by processing the events (i.e. treating them as an 'IF' condition) and proposing possible results (i.e. 'THEN' products). Rules can represent simple dependencies between object classes, threats or concrete object instances. Besides, rules can be enriched with probabilistic values or interactions. For instance, 'IF' area X is flooded longer than 24 hours, 'THEN' it is a 70 percent chance of electricity network damage. It can be realised by the rule engine, for instance, by randomisation of a number ranging from 0 to 1 and comparing the results.
- **PROCeed** (developed within WP5 activities) – a tool responsible for representing the simulation results in graphical way and interaction with a user. Additionally, PROCeed gives access privileges to collections of objects, scenarios and rules and finally allows a user to modify them.
- **Other simulation modules** – modules and simulators supplementing the tools listed above, that can be added to PREDICT tools suite in any moment due to the RESTful approach.

4.4 Principles of the scenario modification

Following principles of the scenario modifications were implemented in the PREDICT – FPT software:

- Following the tree structure concept, alternative scenarios technically cannot merge again to a single events path, cannot have cycles and must have single root node (cannot represent a forest).
- If a predefined scenario was used for a simulation purposes, i.e. new events were generated based on the initial story - it is no longer possible to manually modify events which has been already simulated.
- Each scenario extended with the PREDICT tools represents a new version of the story. Scenario A, which is extended with a new simulated event (e.g. generated by SBR) is processed by the simulation engine as a new version of scenario A. Thus the predefined scenario is the first version of the scenario, which iteratively extended. The full history of changes applied to the scenario will not be stored by the simulation engine because it is not required by any of the PREDICT tools.

4.5 Stored data schema

Storing data of a scenario in a stateless manner requires defining an effective database schema for querying and saving information. The main challenge is to represent changes in objects' states in the next steps without data duplication. Simulation engine uses a RDBMS (Relational Database Management System) which is an efficient way of storing well-defined structures of data. A simplified database schema has been presented on the Figure 3.

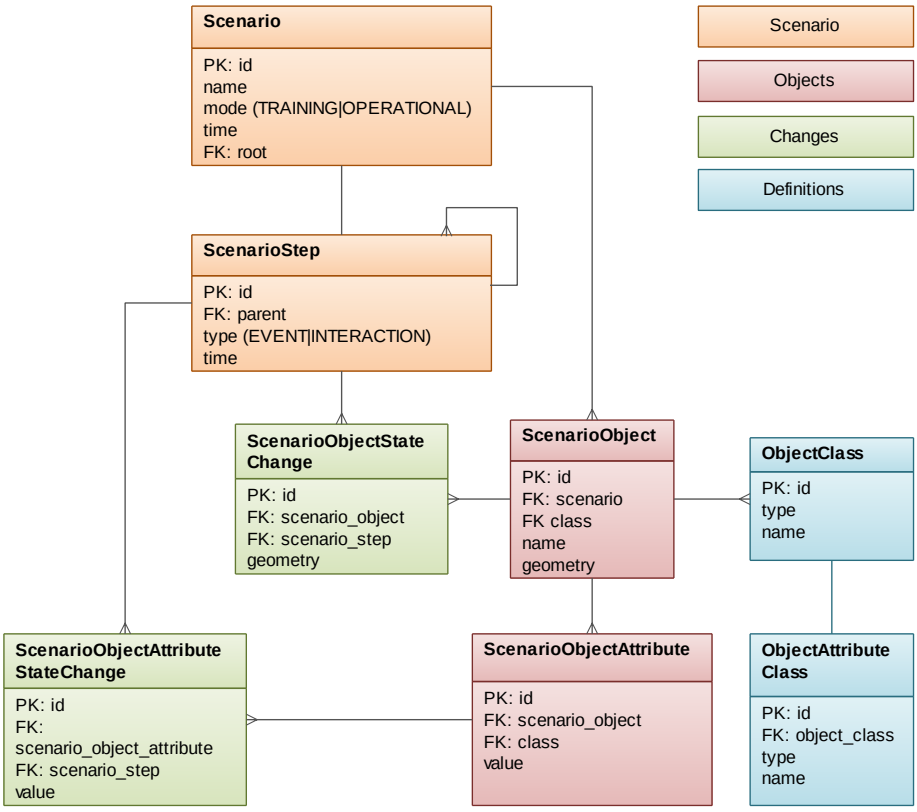

Figure 3 Simplified schema of simulated data

Simulation engine recognizes 2 kinds of state changes represented by entities:

- `scenario_object_state_change` – indicate change of an object's geometry, e.g. moving car, advancing flood;
- `scenario_object_attribute_state_change` – indicate change of an object's properties, e.g. car speed, flood water level.

Each state change is related to a specific step representing a single event. For each step simulation engine allows retrieving a complete picture of the simulated environment including all objects states and attributes changes using discussed database schema and dedicated functions responsible for calculating objects' properties.

The client of the simulation engine is required to specify a concrete step of the scenario when querying for objects or GIS data, otherwise it will receive an initial state of the world.

5. Scenario steps

This section defines scenario steps from the simulation engine point of view, i.e. presents their role, characteristics and possible usage.

According to the description in the section 3, **scenario step represents a single node in a scenario tree**. Each scenario step is characterised with an occurrence time (simulation time), unique identifier, and type. Each scenario step includes information of child nodes which represent the following steps. The number of listed nodes is ranging from 0 (for tree leaves) to theoretically unlimited number – but practically it is not more than 3-4.

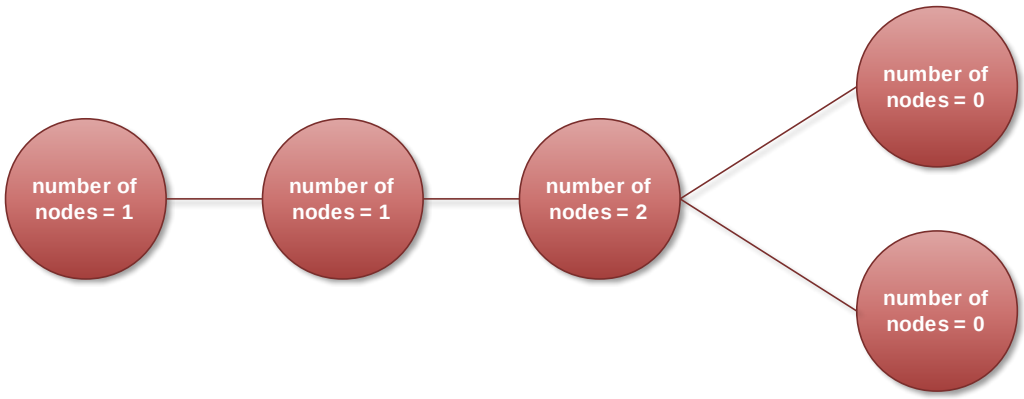

Figure 4 Number of nodes of scenario steps

The main types of scenario steps are called events and interactions. Their detailed characteristics are presented in the following subsections.

5.1 Events

Event represents the change of object's state (or many objects) in the particular step of a scenario. It could be an evolution of threat characteristics, new state of object, crisis management action performed by the user or normal situational routine. Each event has a source which has to be assigned to one of the following categories:

- **PREDEFINED** – event included in the predefined scenario, which represents a crisis situation's event that is confirmed to happen. This event source (i.e. PREDEFINED) is dedicated mostly to threats, object routines or unexpected triggers of the simulation.
- **USER** – new events (i.e. launching rescue action) that can be added by a user to the ongoing simulation after the last event processed by the simulation engine on particular path. It can be, for instance, decision of evacuation of specific area made ad-hoc by a user.
- **DECISION** – decision made by a user due to an interaction initialised by the simulation engine (e.g. interaction defined in predefined scenario). DECISION source type is something different than USER source – the first one can appear only after interaction steps (next to the tree branch, where scenario tree splits into two or more paths).

- **SENSOR** – new event provided by some external sensors (e.g. detection of the toxic cloud). Events added by sensors are not a result of any calculation, but the representation of the potentially realistic world, where many events is being registered by external systems and sources. SENSOR event type can be registered both by a system and a user (e.g. on the basis of observations). It does not represent decision, in contrast to the USER source.
- **SBR** – a new event based on the results generated by SBR, i.e. information of changes of concrete events in specific moments of time. SBR often returns alternative chains of events to, that are added then to scenario by simulation engine.
- **RULE_ENGINE** – a new event generated by the rule engine based on the detected state of simulation and predefined rules.

The table below (Table 1) presents the set of the basic questions regarding the differences between events sources and answers corresponding with various event sources.

Table 1 Questions and answers regarding event sources

Question:	PREDEFINED (PRE)	USER (USE)	DECISION (DEC)	SENSOR (SEN)	SBR (SBR)	RULE_ENGINE (RUL)
Does a user or other tools expects it (is it visible for it)?	No	Yes	No	No	Yes	Yes
Can it be foreseen?	No	No	No	No	Yes	Yes
Can it be added by a user?	No	Yes	Yes	Yes	No	No
Can it be provided by other tool or system?	No	No	No	Yes	Yes	Yes
Is it included by other tools in calculations?	Yes, only if visited	Yes	Yes	Yes, only if visited	Yes	Yes

A simple example of events' sources is presented in the figure below. The first row represents a time of the simulation. Red marker represents the occurrence time of the recently visited step. Events which appear after the red time-marker are planned to be visited. It is worth to know that a SENSOR event is the only one, which is not being expected both by the user / other tools and a simulation engine.

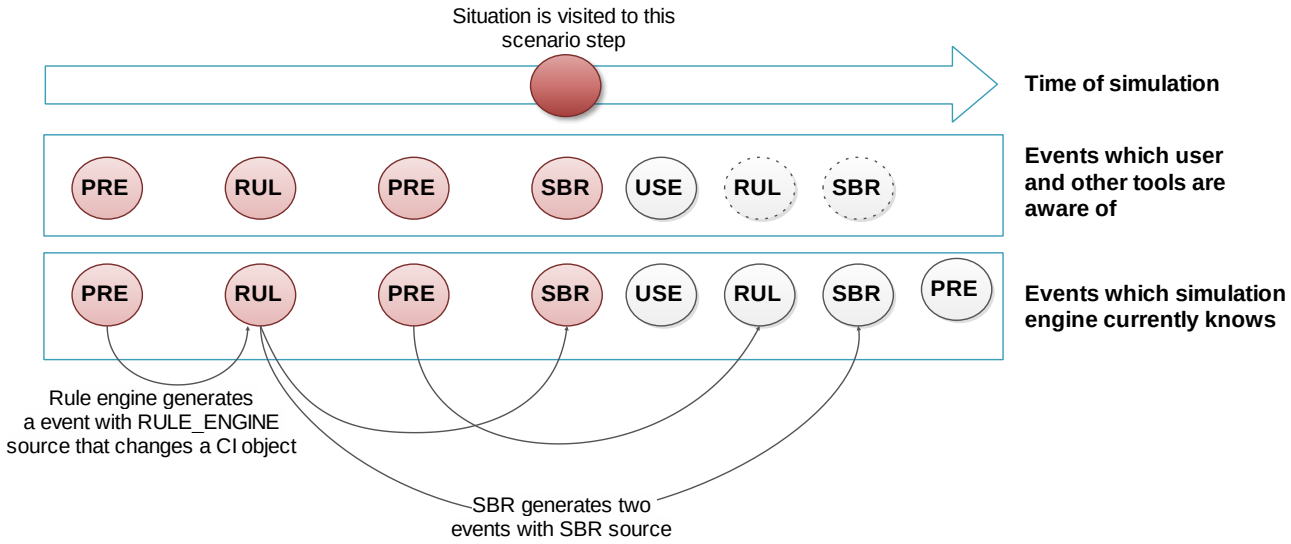

Figure 5 Example of events sources classification

5.2 Interactions

The second type of scenario step is interaction. Interactions represent questions for the user - included in the predefined scenario - which are located before each scenario tree branch. Interactions are the only scenario steps which are followed by two nodes. Interactions are a single-choice type and may include any question, based on the scenario designer decision. With each answer a single path and a corresponding DECISION event is related.

Interactions are processed in the following way:

1. Simulation engine detects interaction which is 'placed' on the scenario path.
2. Simulation engine interprets the paths following the interaction.
3. Question correlated with the detected interaction is sent to the PROCeed to be displayed through GUI.
4. Based on the user's answer, the proper scenario step is being launched and simulated (i.e. visited by the simulation engine).
5. Following the interaction 'regular' simulation process is continued.

Interactions could be also generated automatically by rule engine. Then the question and possible answers is based on the rule definition. This aspect however is still under development and will be described in detail in the D5.3.

6. Specification of simulation engine interfaces

PREDICT-FPT RESTful interfaces were presented in D5.4 and are systematically updated within Swagger documentation API.

Nevertheless, this document needs a supplementation presenting methods important from the simulation perspective. The essential methods are introduced within the further subsections. The following elements for a method description are considered:

- **Name** of the interface presented in form `METHOD /path`, where `path` corresponds to data model entity and the `METHOD` refers to one of the CRUD methods and used color scheme.
  - POST – Create / add new entries (green scheme);
  - GET - Retrieve one / retrieve list of entries (blue scheme);
  - PUT - Update / change state of existing entries (orange scheme);
  - DELETE - Delete entries (red scheme).
- **Summary** briefly describing the interface.
- **Description** (optional) describing the interface in more detailed way.
- **Parameters** of the interface, characterised by the name, location, description, 'required/optional' clause and the parameter schema.
- **Responses** to the request, defined by the code (compliant with HTTP status codes), description and the response schema.

6.1 Providing information about objects' states changes

Changed objects service enables to check the objects, which were changed during the simulation in particular simulation time. On the Figure 6 `/changed-objects` method is requested with `firstStepId` parameter = 3 and `lastStepId` parameter = 8. As a response three `objectsId` are returned: {12, 6, 2}.

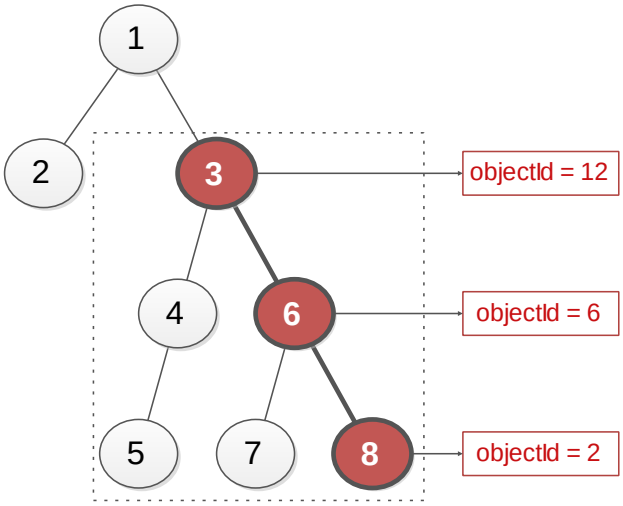

Figure 6 /changed-objects method example

Method is described in detail in the Table 2.

Table 2 GET /v1/scenarios/{scenarioId}/changed-objects

GET /v1/scenarios/{scenarioId}/changed-objects				
Summary				
List ids of changed objects between two steps (inclusive) on the same tree path.				
Description				
Example 1: list changes that occurred in step "3" of scenario "1": /v1/scenarios/**1**/changed-objects?firstStepId=**3**&lastStepId=**3** **Example 2:** list changes that occured in all steps on the tree path between steps "3" and "5" of scenario "1": /v1/scenarios/**1**/changed-objects?firstStepId=**3**&lastStepId=**5**				
Parameters				
Name	Located in	Description	Required	Schema
scenarioId	path	Unique identifier of a scenario.	Yes	integer(int64)
firstStepId	query	Unique identifier of the first scenario step (inclusive).	Yes	integer(int64)
lastStepId	query	Unique identifier of the last scenario step (inclusive).	Yes	integer(int64)
Responses				
Code	Description			Schema
200	OK			{ }

default	Unexpected Error			Error {
				code: integer *
				message: string *
				}

6.2 Providing information about objects' states in relation to particular scenario step

`/objects` method provides parameters of each object valid for the requested `scenarioStep`. This enables to reproduce the common operational picture of the scenario in the desired moment of the simulation. Details of the method are presented in the Table 3.

Table 3 GET /v1/scenarios/{scenarioId}/objects

GET /v1/scenarios/{scenarioId}/objects				
Summary				
Retrieves a list of all scenario objects at a given step (after previous steps on a tree path of alternative situations occur).				
Parameters				
Name	Located in	Description	Required	Schema
scenarioId	path	Unique identifier of a scenario.	Yes	`integer(int64)`
stepId	query	Unique identifier of a scenario step. If not provided the initial state of an item will be assumed.	No	`integer(int64)`
size	query	number of items to be returned	No	`integer(int64)`
page	query	page to be returned, 0-indexed	No	`integer(int64)`
sort	query	sort items according to the specified expression(s), e.g. "name,desc"	No	`string((-?\w+)(,(-?\w+))*)`
Responses				
Code	Description			Schema
200	OK			{ total: integer (int64) *Total number of scenario objects.* data:[ScenarioObject { }] }
default	Unexpected Error			Error { code: integer * message: string * }

6.3 Updating objects' states in relation to particular scenario step

The method under `/rule-engine/run` is an example of the PREDICT - FPT simulation module method executed by the simulation engine. Simulation engine runs rule engine (WP5 module) to generate possible impact based on the particular `scenarioStep` (it is always an event). The result of the method is a subtree which is then injected into the scenario tree.

Table 4 POST /v1/rule-engine/run

POST /v1/rule-engine/run				
Summary				
Evaluates state of the world at a given step of the scenario.				
Description				
Based on rules predefined by the experts generates events (next steps in the scenario) leading to changes in objects' attributes. Returns subtree of the scenario including events generated by the system, with root of the tree equal to the given step (stepId).				
Parameters				
Name	Located in	Description	Required	Schema
scenarioId	query	Unique identifier of a scenario.	Yes	`integer(int64)`
stepId	query	Unique identifier of a scenario step. If not provided the initial state of an item will be assumed.	No	`integer(int64)`

Responses

Code	Description	Schema
201	Created	ScenarioStep { *Represents a single node in a tree of alternative situations. Parent of this node might be derived by the client from the subtree of the scenario to which the node belongs.* id: ScenarioStepId integer type: string *Identifies type of the step.* • ***EVENT*** *– defines change of an object's state.* • ***INTERACTION*** *– requires the user to choose one from many options.* **Enum:** parent: ScenarioStepId integer (int64) *Unique identifier of a scenario step.* nodes: [undefined] time: string (date-time) *Date and time when the step occurs in a simulation (TRAINING mode) or when an event occurred in reality as reported*

		by the sensor (OPERATIONAL mode). This time must be after the "startAt" time of the Scenario to which the step belongs, and after the "occuredAt" time of the parent node (if exists. }

default	Unexpected Error	Error {
		code: integer *
		message: string *
		}

6.4 Searching for events' classes

User's decisions taken or other interactions (as described in section 5) are one of the specified events sources. E.g. the decision made is correlated with a particular rescue action to be taken into account during the simulation. The consequences of the given rescue action are strictly dependent and limited to the respective object's class and current object's state. E.g. the car wouldn't stop, if it wasn't moving. The proper search method, enabling user to find valid action for the given object and state of simulation, is described in the table below.

Table 5 GET /v1/events/search

GET /v1/events/search				
Summary				
Retrieves a list of all event classes that can change given attribute and object classes.				
Parameters				
Name	Located in	Description	Required	Schema
size	query	number of items to be returned	No	integer(int32)
page	query	page to be returned, 0-indexed	No	integer(int64)
sort	query	sort items according to the specified expression(s), e.g. "name,desc"	No	string ((-?\w+)(,(-?\w+))*)
objectClassId	query	Unique identifier of an object class.	Yes	integer(int64)
objectAttributeClassId	query	Unique identifier of an object's attribute class.	Yes	integer(int64)
Responses				
Code	Description			Schema
200	OK			{ total: integer (int64) Total number of event classes. data: [EventClass { }]

		}

default	Unexpected Error	Error {
		code: integer *
		message: string *
		}

7. References

[1] D6.2 Cognitive analysis report to support decision tools specifications

[2] D6.3 Multi-Criteria Decisions support report & software

[3] D5.1 PREDICT – FPT Design Document

[4] D5.4 PREDICT – FPT External Interfaces Specification

[5] D3.1 Methodology for the identification and probability assessment of cascading effects

[6] D3.1 Methodology for the identification and probability assessment of cascading effects

[7] D3.2 Methods of threat quantification

[8] D3.3 Methodologies for the operation time model lay-out and specification

[9] D4.1 System design document

[10] D4.2 System realisation document

[11] Stewart Robinson (2004), Simulation – the Practice of Model Development and Use, Wiley.

8. List of acronyms

CI – Critical Infrastructure

DES – Discrete – Event Simulation Concept

FPT – Foresight and Prediction Tool

iPDT – Integrated PREDICT Tool Suite

SBR – Scenario Based Reasoning Tool

List of Tables

List of Figures

PREDICT PROJECT PARTNERS