A software framework for the generation of dynamic vulnerability maps for risk assessment

P. Arena[1], L. Patanè[1], S. Caruso[1], M. Anastasi[1], A. Cannata[2]
[1]*DIEES- University of Catania, Catania, Italy.*
[2]*Euroconsult s.a.s., Ragusa, Italy.*

Abstract

In this paper a software framework able to generate dynamic vulnerability maps for risk assessment is proposed. A series of models for different risk scenarios have been considered and described together with their implementation in the developed software framework. Finally the visualization of the acquired data and simulation results can be used as a support for real-time emergency planning.
Keywords: prediction models, disaster management

1 Introduction

One of the main goals of environmental risk analysis consists of foreseeing and estimating the environmental impact due to critical events happening within an industrial plant. Often, static vulnerability maps are used for preventive impact analysis; sometimes, even more sophisticated dynamic maps are used for predicting the consequences of industrial accidents.

Using an appropriate software tool enclosing both of these potentialities, and having the capability to let the user choose and finely tune the prediction models is a key element to insure a prompt response from authorities and to give rescue teams the ability to react as quickly as possible.

An ideal integrated framework for disaster management should include the flexibility and expandability of accurate software for dynamic modelling, but also the intuitiveness and simplicity of a business instrument based on a user-friendly interactivity. Our risk prevention and analysis platform was developed with this aim in mind from the outset. This instrument features a geographic localisation platform based on photographic maps and offers a complete set of tools for the analysis of static and dynamic data.

Our platform, named Juan Chedan (Just Another Chemical Analyzer), consists of a main framework that offers data gathering, synchronisation and presentation facilities, and a set of plug-in offering multiple computational algorithms based on several prevision models. The platform is expandable by the addition of new plug-in that can be developed using a very streamlined SDK. Thanks to the use of plug-in, the functionalities of the main framework are factored and are available to be used transparently and independently from the data semantics, for computations related to accident models as diversified as gas dispersions, explosions and fire accidents.

This structure may help to improve the productivity, and thus the response speed, of emergency responders, by offering a single unique, easy to learn and streamlined environment able to run multiple prevision models using a common user interface.

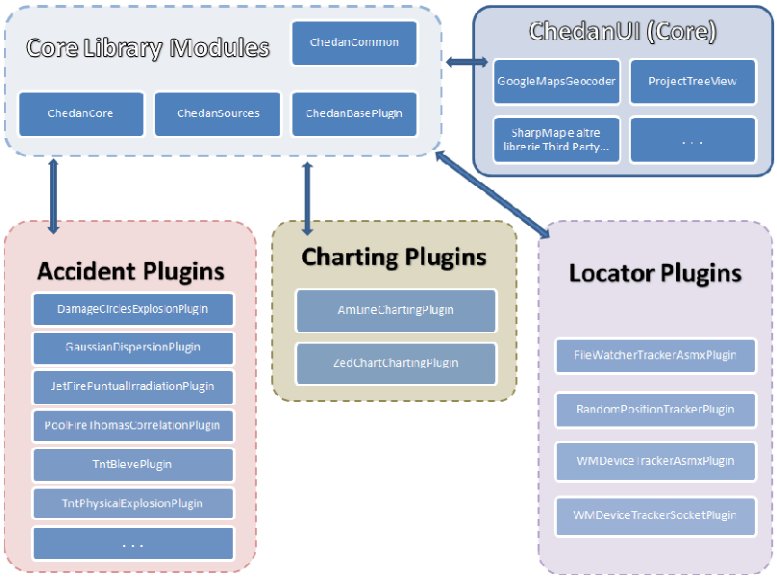


Figure 1: Juan Chedan architecture and list of developed plugins.

2 Software architecture

The aim of our work was to simulate as closely as possible the consequences of an industrial incident; with this aim, we have studied a number of mathematical models existing in literature, and sometimes we have rearranged and fixed some of the less detailed models in order to achieve more consistent results. Then we tried to find a compromise between the reliability of a model and its user friendliness, choosing in the end a set of optimal models and trying to find the best way to implement them. Each model was implemented as a plugin for the Juan Chedan platform, following the Juan Chedan SDK

guidelines. Thanks to this choice, it's possible to leverage the Juan Chedan visualization engine and data input framework. Fig. 1 shows the interactions between the Juan Chedan modules and the different plugins implemented. Further informations about the integrated framework structure can be found in [1].

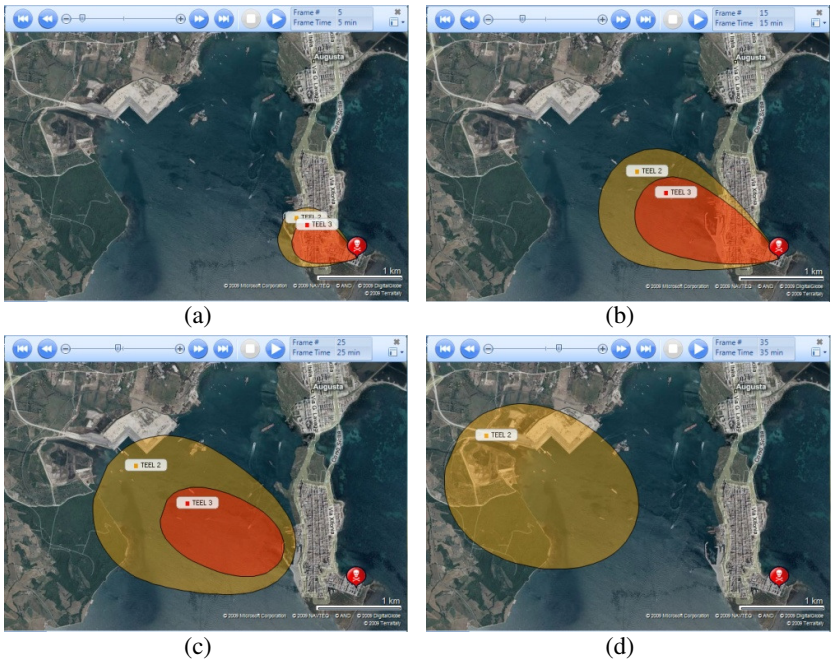
3 Risk scenarios and Models

The considered risk scenarios are:

- Toxic clouds dispersion,
- Various kind of fires triggered by the burning of toxic clouds,
- Explosions

These scenarios are frequently connected to one another (domino effect), but in the context of this paper, they are studied like isolated events.

Here we will discuss the most important models, their implementation details and their integration with Juan Chedan.



(a) (b)

(c) (d)

Figure 2: Toxic cloud shifting: these result are obtained considering a continue release of Chlopicrin (20Kg/s) for 20 minutes and a wind speed of 2m/s. a) Cloud's shape after 5 minutes. b) cloud's shape after 15 minutes. c) cloud's shape after 25 minutes. d) cloud's shape after 35 minutes.

3.1 Neutral gas dispersion

We have taken into consideration the dispersion of neutral gases, implementing the most popular mathematical model used for this kind of simulation: the Gaussian model [2,3,4].

The Gaussian model of Pasquill-Gifford considers a Gaussian dispersion along the horizontal and vertical axis, in a system with x-axis oriented in downwind direction, y-axis in crosswind direction and z-axis in vertical direction.

For a continuous release of material, if neglecting the absorption of the ground, the dispersion from a hole at a known height is:

$$C = \frac{G}{2\pi\sigma_y\sigma_z u}\exp\left[-\frac{1}{2}\left(\frac{y}{\sigma_y}\right)^2\right]\cdot\left\{\exp\left[-\frac{1}{2}\left(\frac{z-H}{\sigma_z}\right)^2\right]+\exp\left[-\frac{1}{2}\left(\frac{z+H}{\sigma_z}\right)^2\right]\right\} \quad (1)$$

Where:

C = concentration (kg/m^3) in (x, y, z)

G = flow (kg/s)

H = height of emission from the ground (m)

u = wind speed (m/s)

x, y, z = distance from the source (m), downwind (x), crosswind (y) and vertical (z)

σ_y, σ_z = dispersion coefficient in y and z

The obtained values are compared with toxicity limits (e.g. IDLH) or with inflammability limits if the cloud comes from inflammable gas dispersion. We then convert the concentration values (C, kg/m^3) to ppm or to volume percentage using the perfect gas law:

$$C_{ppm} = C(kg/m^3)\frac{R\cdot T}{P\cdot M}10^6 \quad (2)$$

$$C_{\%volume} = C(kg/m^3)\frac{R\cdot T}{P\cdot M}10^2 \quad (3)$$

Where

R = gas constant 0.0821 (lt atm /mol K);

T = temperature (K);

P = pressure (atm);

M = molecular weight (kg/kmol).

Plugin implementation: This algorithm is implemented by the GaussianDispersionPlugin.

The levels of concern suggested from the plugin are specific for each substance and are taken from a chemical database [5].

The produced outputs are: time-independent isoconcentration curves, overlaid on the map and representing suggested or custom LOCs, and a time-

dependent animation of the curves' variations in the time.

The plugin can also produce charts describing time-dependent information on specific points chosen by the user by positioning an observer on the map. In particular, the available charts are concentration and dose profiles in a specific point.

3.2 JetFire

In this work we are only going to describe one type of fire accident, although more fire plugins were actually implemented. The scenario we're going to consider is the JetFire, i.e. the release of an inflammable gas from a small break that can originate a long upward flame.

We suppose that no flame is originated where the combustible's concentration becomes less than the LFL (Lower Flammable Limit) value, and we also ignore the effects of wind and gravity on the flame (so there are no differences between horizontal, vertical or oblique jet): the danger zone stretches for about twice the flame's length. [6,7]

3.2.1 Punctiform AGA

The main model describing a JetFire accident, considers the source as a point localized in the centre of the flame. [3,6] The data required for computation are:

- Temperature
- Air humidity percentage
- Burning speed M_b in $\frac{Kg}{m^2 \cdot s}$
- Combustion heat of the liquid combustible H_c in $\frac{KJ}{Kg}$
- Part of heat irradiated F_{rad}

Algorithm: We compute the total heat of the flame, as $Q_r = M_b \cdot H_c$ (output in kW). Then we extract the heat irradiated from the flame as $Q = F_{rad} \cdot Q_r$

Now we need to compute the partial pressure of the steam in the air in Pa. We can simply obtain this information from the temperature and the relative humidity, this one is in fact the ratio between partial pressure of steam, p_w, and saturation pressure of the steam at a certain temperature, p_s. The equation of the steam pressure is:

$$\ln p_s = 18.3036 - \frac{3816.44}{T - 46.13} \tag{4}$$

Where
p_s = steam pressure [mm Hg]
T = temperature in Kelvin

In order to obtain the partial pressure steam, we convert p_s toPa. Then:

$$p_w = \frac{\text{relative humidity}}{100} \cdot p_s \quad [Pa] \tag{5}$$

We can now compute the transmissivity:

$$\tau = 2.02 \cdot (p_w \cdot d)^{-0.09} \quad (6)$$

Where d is the distance from ground between flame and observer.
We can then calculate the View factor in the point:

$$F_p = \frac{1}{4 \cdot \pi \cdot l^2} \quad (7)$$

Where l is the distance measured along the diagonal, between the flame centre and the observer.

Finally we extract the most important punctual value, which is the thermic stream on the target:

$$Q_{target} = Q \cdot F_p \cdot \tau \quad (8)$$

Q_{target} is the reference point for the iso-irradiation curves that we want to overlay on the geographical map in Juan Chedan. In order to draw the curves, we only need to know the distance d from the whole eqn.(8).

Notes and assumptions: An important parameter needed for the model is the flame's height. This could be a problem as, in order to extract *d* from eqn.(8), we need to know the relation between that and the diagonal distance *l* (in the view factor formula in eqn.(7)). We can in fact assimilate the flame profile with a rectangular triangle (we aren't considering the wind, so we assume the flame is not inclined), where we know the larger cathetus (distance from the ground *d*), but we don't know the hypotenuse (diagonal distance *l* between flame centre and observer).

In order to obtain this information and put *l* and *d* in relation, usually models assume that the flame's height H is known (and then the shorter cathetus H/2, given that we need a point in the middle height of the flame). Now we can assert:

$$l = \sqrt{d^2 + \left(\frac{H}{2}\right)^2} \quad (9)$$

without any problem to extract *d* from eqn.(8). In this context the problem was avoided by approximating the superficial distance with the diagonal one, then equalizing *l* and *d*; this approximation is allowed because the view factor used in this model, which is usually expressed like:

$$F_p = \frac{\cos\theta}{4 \cdot \pi \cdot r^2} \quad (10)$$

sets $\cos\theta$ as 1, supposing then a null angle, so that superficial and radial distance are coincident in this formula (eqn. (7))

Plug-in implementation: The component that implements this algorithm is

called JetFirePunctualIrradiationPlugin.

The levels of concern suggested by the plug-in are fixed by the (Italian) normative DM 15.05.96 and DM 20.20.98. [8,9,10]

This plug-in produces only a time independent output, a set of concentric circles overlaid on the map, with a variable radius based on the suggested or manually inserted levels of cares (LOCs). An example of iso-level radiation plot is shown in Fig.3.

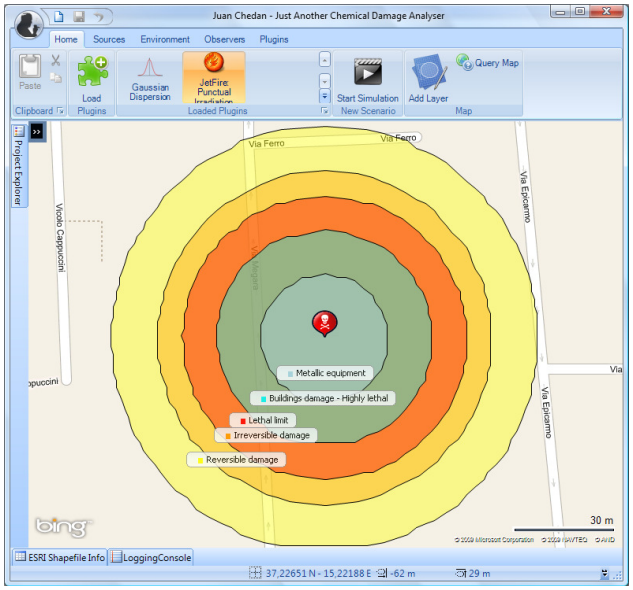


Figure 3: Iso-radiation footprint from the time independent computation of JetFirePunctualIrradiation. These results have been obtained considering a 100 Kg/s release of Chloropropane.

3.3 UVCE

One of the most dangerous explosion accidents is the UVCE (Unconfined Vapour Cloud Explosion) [3,6,7]; it has its source in the combustion of inflammable gases, so the simulation will only be allowed if the user sets up an incident scenario involving an inflammable gas. A simple but reliable formula for computing the amount of substance contained in the toxic cloud is:

$$Q = \frac{F \cdot I}{v} \tag{11}$$

Where:

F = vapour flow,

I = distance (along the cloud centre line) where an LFL concentration is found

v = wind speed

3.3.1 TNT-Equivalent Model

This model is based on the transposition of the mass of a substance involved in an explosion to an equivalent mass of TNT, whose effects are known.

Algorithm: The size of the inflammable cloud must be evaluated (i.e. the concentration has to be higher than LFL). The LFL levels are specified in a chemical database, and are generally expressed in ppm, so we need to convert them to a concentration value expressed in $\frac{Kg}{m^3}$, using eqn. (2) and (3).

The plug-in evaluates the inflammable mass of the cloud. Then it computes the equivalent TNT mass according to the formula:

$$M_{tnt} = \eta * m * \frac{H_{comb}}{H_{tnt}} \quad (12)$$

Where:

η = efficiency factor (default value 0.05)

m = mass of inflammable substance inKg,

H_{comb} = burning heat (default value $10 * M_{tnt} \left[\frac{J}{Kg}\right]$)

H_{tnt} = calorific power of the TNT ($4106 \frac{J}{Kg}$.)

A UVCE in fact consumes only a minimal part of the whole energy involved in the combustion, so the efficiency is a value between 1% and 10%.

At this point we can extract the scaled distance using:

$$z = \frac{R}{\sqrt[3]{M_{tnt}}} \quad (13)$$

Where:

z = scaled distance

R = distance from the explosion [m]

M_{tnt} = mass of equivalent TNT, obtained by the eqn.(12) [Kg]

Once the scaled distance is known, it's possible to extract different kinds of information from a diagram describing the well known effects of the TNT obtained with experimental tests: the arrival time of the overpressure wave, the overpressure value reached at a certain distance, and the duration of the wave [12].

The arrival time is one of the punctual values that the plugin returns, and it is used to create stand-alone charts describing the advancement profile of an overpressure wave in a point. Analogously we can compute the scaled duration, and then the actual overpressure duration in seconds.

In order to compute the overpressure at a specific distance we could have used the same technique, but we adopted an approximation present in literature, obtained from interpolation [11]:

$$Ps = 67 \cdot \frac{1}{z} + 370 \cdot \frac{1}{z^2} \quad (14)$$

This equation returns an overpressure value expressed in Kpa.

This value is the last punctual output produced by the plug-in. It is shown as

a label for the observers and is also used to draw the stand-alone overpressure trend chart. This is also the value used to create the map overlays. An example of overpressure iso-level curves is shown in Fig. 4.

Plugin implementation: The component that implements this algorithm is called TntExplosionPlugin.

The levels of concern suggested by the plug-in are fixed by the (Italian) normative DM 15.05.96 e DM 20.10.98. [8,9,10]

This plugin produces a time-independent output, in the shape of concentric circles overlaid on the map, with a variable radius based on the suggested or manually inserted LOCs. It also produces a time-dependent animation describing the wave front progress.

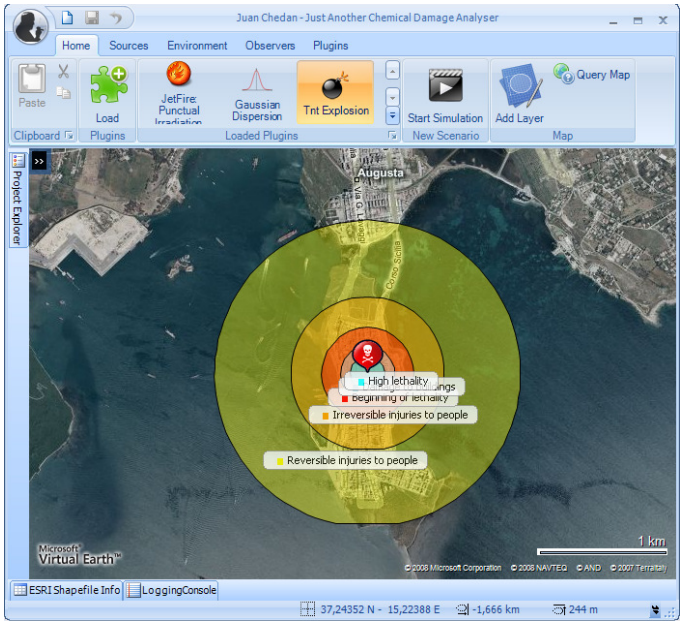


Figure 4: Steady state representation (time independent) of the Overpressure LOCs by the TntExplsionPlugin for a 100Kg/s release of inflammable substance

The plugin also generates some stand-alone charts, describing the profile of the wave front when it meets an observer. In particular, the chart shows the time of the arrival of the wave front, the maximum value of overpressure and the overpressure duration (see Fig.5).

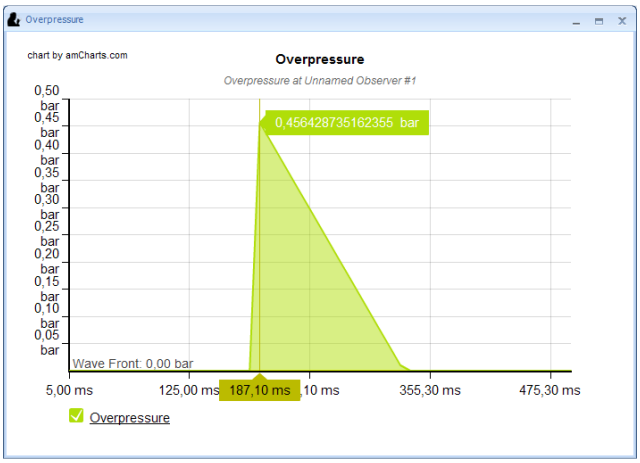


Figure 5: Wave front in an observer point located at 190m from the source.

4 Integration with Juan Chedan

In order to be integrated with the Juan Chedan platform, a plugin must implement the software interface IAccidentPlugin, which acts as a communication protocol with the Juan Chedan core. [1]

The methods that the public interface requires, are:

- **GetCurve**

GetCurve (LOC, source, computation instant).

The output is a "Curve" object, that is a list of coordinate pairs corresponding to points of iso-level. This method is used to create the iso-level curves for the specific accident on the map.

- **GetValuesInPoint**

GetValuesInPoint(observer point, source)

The output is a set of one or more values related to an observer point. This information is used in the Juan Chedan platform to show the maximum value of risk in a certain point, for a certain scenario.

- **GetValuesTrendInPoint**

GetValuesTrendInPoint(observer point, source)

The output is a list of "trend" objects, where each trend contains a curve with a set of (x,y) couples, the axis labels and units.

This output is used by the Juan Chedan Core to build stand-alone charts describing time-dependent data about a geographic point.

- **GetValues**

GetValues(instant)

The output returns a list of heterogeneous geographically unrelated values.

- **GetValuesTrend**

GetValuesTrend(instant)

Analogously to the GetValues method, the output is a list of heterogeneous geographically unrelated "trends".

- **GetRequirements**

This method is called by the Core before starting a computation, and returns an object describing the minimum set of information that the plugin requires to execute successfully.

- **GetCapabilities**

This method allows the plugins, to describe their functionalities. E.g., if a plugin is able to produce an animation, or which optional interface methods are implemented.

- **GetSuggestedLOCs**

GetSuggestedLOCs(source, flag for time-dependent use of the LOCs)
The output is a set of suggested levels of concern, with labels and units.
This information is used to provide the user with a set of standard levels of concern, specific to each different scenario.

5 Conclusions

After a short analysis of a few different incidental model plugins, we can assert that the plugins developed and discussed in this work are not only a valid support for risk analysis but are also a very simple tool for the end user. They enable emergency responders to use complex mathematical models in a very quick and easy way, with almost no need for specific skills or training. New plugins implementing more models can be easily added to the suite to simulate new scenarios, and new functionalities can thus be added without introducing the need for further operator training.

6 Acknowledgment

This work was partially supported by the Isemiha project, (1999.IT.16.1.PO.011/3.14/5.2.13/0312).

References

[1] Arena, P., Patanè, L., Anastasi, M., Caruso, S. & Cannata, A., *An integrated system for disaster management*, Int. Conf. On Disaster Management 2009, UK.

[2] Hanna, S.R. & Drivas, P.J., *Guidelines for use of vapour cloud dispersion models*, Center for Chemical Process Safety, AIChE, New York, 1987

[3] Arena, P., Italia, F. & Patanè, L**.** *SoftComputing per previsione e controllo di Rischio d'Area.* Catania : Edizioni Cavallotto, 2003.

[4] *Handbook of chemical hazard analysis procedures*, FEMA-EPA, 1989

[5] *CAMEO Chemical.*Available on line at: http://cameochemicals.noaa.gov/

[6] Mazzarotta, B**.**.*Fire and Enissions Risk Course*, 2006-07 available on line at: http://ingchim.ing.uniroma1.it/~mazzarot/pagina mia internet/corso RIE.htm.

[7] Contini, S., *Rassegna di modelli per la valutazione degli effetti di esplosioni*, ISEI/IE 2397/93, Commissione delle Comunità Europee, Ispra (VA), 1993

[8] Quest Consultants Inc. The Quest Quarterly. *Quest Consultants Inc.* 1999. Available on line at: www.questconsult.com.

[9] Center for Chemical Process Safety. *Guidelines for Chemical Process Quantitative Risk Analysis, 2nd edition.* New York : AIChE, 2000.

[10] Lees,F.P. *Loss Prevention in the Process Industry, 2nd edition*, Vol. 2. London : Butterworths, 1996.

[11] Brasie, W.C. & Simpson, D.W. *Guideline for Estimatin Damage from Chemical Explosion* – St.Louis, MO: AIChE, 1968

[12] Caruso, S**.,** *Data Interpolation with SPLINE in C#. JuanDoNeblo, 2009.* Available on line at: http://geekswithblogs.net/JuanDoNeblo/archive/2007/10/25/Data Interpolation-with-SPLINE-in-Csharp.aspx.