Monte Carlo event generators tutorial

ASP 2016

Kigali, 03-04 Aug 2016

1 Introduction

This set of tutorials aims at giving you an understanding of the construction principles and inner workings of Monte Carlo event generators, supporting what you have learned in the lectures by hands-on experience. The tutorial will proceed in two stages. The first stage tries to give you an overview on how to run one of the generators and lets you explore the structure of an event. The second part then scrutinises different approximations made in the different event generation stages to give a feeling for their relative importance to different physics questions.

The software and setups used in both tutorial sessions are pre-installed on the desktop computers. To access them open a terminal ("Application→Accessoires→Terminal") and navigate to the directory `MCTutorial`

```
cd MCTutorial
```

Therein you will find the following subdirectories `install`, `Day1` and `Day2`, a pdf file `MC-Tutorial-Instructions.pdf` (this document) and a script `paths.sh`. At the beginning of each day (and for every shell you use) first execute

```
source paths.sh
```

to make the system aware of the additional software for the tutorial. Open the tutorial instructions with

```
evince MC-Tutorial-Instructions.pdf &
```

and begin with the tutorial.

1.1 The SHERPA event generator

SHERPA [1] is one of the standard complete Monte Carlo event generator frameworks in use at the LHC. As such it can be used to simulate the full spectrum of physics needed for high-energy reactions at hadron colliders. SHERPA emphasises describing the perturbative part of the event at the highest possible accuracy.

To this end, SHERPA is comprised of:

- Two matrix element generators, AMEGIC++ [2] and COMIX [3]. When supplied with the one-loop matrix element they are capable of next-to-leading order calculation.
- A parton shower, CSSHOWER++ [4], based on Catani-Seymour dipoles. It fully supports NLO-matching and multijet merging techniques.

1	d	11	e^-	21	g	111	π^0	511	B^0	113	ρ^0	2112	n
2	u	12	ν_e	22	γ	211	π^+	521	B^+	213	ρ^+	2212	p
3	s	13	μ^-	23	Z	311	K^0	221	η	333	ϕ	3112	Σ^-
4	c	14	ν_μ	24	W^+	321	K^+	331	η'	223	ω	3122	Λ^0
5	b	15	τ^-	25	h	411	D^+	130	K^0_L	313	K^{*0}	3212	Σ^0
6	t	16	ν_τ			421	D^0	310	K^0_S	323	K^{*+}	3222	Σ^+

Table 1: Some PDG codes for the most important particles. Antiparticles are denoted by a minus sign, e.g. -1 indicates a $\bar{d}$-quark. Mesons and baryons are constructed from their constituents with the final digit indicating the spin state in units of $2s+1$. A few exceptions are in place for mixed states such as K^0_L and K^0_S.

- A multi-parton interaction model, Amisic++ [5].
- A cluster hadronisation model, Ahadic++ [6].
- A hadron decay module.
- A soft-photon resummation module, Photons++ [7], to account for higher-order QED corrections.

Its output of stable hadrons can then be interfaced to full (Geant4) or fast (Pgs, Delphes) detector simulations if needed.

Sherpa is steered using input files, which consist of several sections grouping the parameters by physics range. A comprehensive list of all input parameters is given in the manual [8, 9]. For the purpose of this tutorial we will focus on the most relevant ones in the respective sections of the tutorial.

2 The structure of a Monte Carlo event

2.1 The event record

The event record stores every step of the evolution from the initial dissociation of the protons through the hard interaction to the final ensemble of stable particles that eventually hits the detector. The details of the event record, however, are specific to each generator and the models and approximations employed. All generators adhere to a common naming scheme for particles. A listing is available at

`http://pdg.lbl.gov/2008/mcdata/mc_particle_id_contents.html`

or summarised shortly in Tab. 1. These identifiers are used both for internal representation and in the common HepMC output format.

The HepMC output itself, as it is aimed at storing events on disks and transferring information from the Monte Carlo generators to analysis software (Rivet, experiment internal software like Athena and Cmssw), is rather compact. We thus investigate Sherpa's own representation of the same information.

Go to the directory containing the first tutorial

```
cd Day1
```

In this directory you find a SHERPA input card, `Run.dat`. Open the file with a text editor and examine its context. Can you identify which process is calculated? What is the collider setup? How many events will be generated? Please note the settings for the output level and find out their meaning from the Manual.

Then start SHERPA with

```
  Sherpa -f Run.dat
```

telling it which input card to read in. The first output you will see is the logo and some information on the version used. After that, the particle content of the physics model used is listed, supplemented by a list of containers for abbreviating process declarations and the like. Now the PDFs, the shower and the hadronisation model are initialised, followed by the matrix element generator AMEGIC++. Finally, your first run will end by

```
AMEGIC::Single_Process::WriteLibrary :
   Library for 2_1__G__G__h0 has been written, name is P2_1_1_2_6_4_7_0
 done ( 18 MB, 0s / 0s ).
Matrix_Element_Handler::InitializeProcesses(): Performing tests Sherpa:
Amegic::PerformTests throws normal exit:
   New libraries created. Please compile.
-----------------------------------------------------------------------
Please cite the publications listed in 'Sherpa_References.tex'.
  Extract the bibtex list by running 'get_bibtex Sherpa_References.tex'
  or email the file to 'slaclib2@slac.stanford.edu', subject 'generate'.
-----------------------------------------------------------------------
```

with the important sections highlighted in red: AMEGIC++ informs you that it just created new source code, stored in `Process`, that now needs compiling. This can be done by

```
  ./makelibs
```

Now the matrix elements are set up and can be used. Thus, starting SHERPA for a second run

```
  Sherpa -f Run.dat | less -R
```

The output of SHERPA is piped into the reader `less` such that it is browsable. The option `-R` ensures the correct display of formatted output. The modules are initialised as in the first run, only now instead of writing out the process source code AMEGIC++ loads the compiled libraries

```
Matrix_Element_Handler::BuildProcesses(): Looking for processes .. done ( 18 MB, 0s / 0s ).
Matrix_Element_Handler::InitializeProcesses(): Performing tests  done ( 18 MB, 0s / 0s ).
```

Can you identify what happens next (`Initialising hard decay tables.`)? After this is done, the multiple interactions are initialised. As SHERPA employs a model basing on multiple soft, but perturbative parton scatters their cross section has to be integrated to generate unweighted multiple interactions later. Its results are tabulated in a grid. Which processes constitute the multi-parton interactions? Why?

This is followed by the integration of the hard process. Again, remember which kind of information is necessary to generate unweighted events. This is book-kept at this stage. Also the process independent integrators are adapted to the process and cuts at hand.

```
Start calculating the hard cross sections. This may take some time.
Process_Group::CalculateTotalXSec(): Calculate xs for '2_1__j__j__h0' (Amegic)
Starting the calculation at 13:35:51. Lean back and enjoy ... .
5.72079 pb +- ( 0.0749908 pb = 1.31085 % ) 5000 ( 5000 -> 100 % )
full optimization:  ( 0s elapsed / 4s left ) [13:35:51]
5.73361 pb +- ( 0.0531471 pb = 0.926941 % ) 10000 ( 10000 -> 100 % )
full optimization:  ( 0s elapsed / 5s left ) [13:35:51]
5.73329 pb +- ( 0.0432981 pb = 0.755205 % ) 15000 ( 15000 -> 100 % )
full optimization:  ( 0s elapsed / 5s left ) [13:35:52]
...
5.79041 pb +- ( 0.00628112 pb = 0.108475 % ) 160000 ( 160000 -> 100 % )
integration time:  ( 2s elapsed / 3s left ) [13:35:55]
2_1__j__j__h0 : 5.79041 pb +- ( 0.00628112 pb = 0.108475 % )  exp. eff: 70.8628 %
  reduce max for 2_1__j__j__h0 to 0.995595 ( eps = 0.001 )
Calculating the hard cross sections has been successful.
```

Finally, SHERPA is ready to start the event generation.

```
---------------------------------------------------------
-- SHERPA generates events with the following structure --
---------------------------------------------------------
Perturbative       : Signal_Processes
Perturbative       : Hard_Decays
Perturbative       : Jet_Evolution:CSS
Perturbative       : Lepton_FS_QED_Corrections:None
Perturbative       : Multiple_Interactions:Amisic
Perturbative       : Minimum_Bias:Off
Hadronization      : Beam_Remnants
Hadronization      : Hadronization:Ahadic
Hadronization      : Hadron_Decays
--------------------------------------------------------
```

Now follows SHERPA's representation of the structure of the event. The event record can be traversed from vertex (in SHERPA called `Blob`) to vertex along the particles that run in between. A typical vertex has the following structure and displays the following information

```
Blob [16]( 10, Beam                    , from Beam 0, 1 -> 6 @ (0,0,0,0)
Incoming particles :
[a] 2 P+                  63 (  13 ->   10) [( 4.0000e+03, 0.0000e+00, 0.0000e+00, 4.0000e+03), p^2= 8.8035e-01, m= 9.3827e-01] (  0,  0) 0
Outgoing particles :
[I] 1 G                    6 (  10 ->    3) [( 3.3031e+02,-5.0344e-01,-1.2838e+00, 3.3031e+02), p^2=-1.3097e-10, m= 0.0000e+00] (650,649) 0
[I] 1 G                   33 (  10 ->    5) [( 3.2599e+02, 1.0054e-01,-1.8059e+00, 3.2599e+02), p^2=-1.4552e-11, m= 0.0000e+00] (653,650) 0
[I] 1 G                   46 (  10 ->    7) [( 5.7038e+01, 4.0442e-01, 1.8314e+00, 5.7007e+01), p^2=-4.6311e-07, m= 0.0000e+00] (649,651) 0
[I] 1 G                   56 (  10 ->    9) [( 1.0611e+03,-1.1099e-01, 1.8785e+00, 1.0611e+03), p^2= 2.3283e-10, m= 0.0000e+00] (651,652) 0
[B] 4 ud_1                65 (  10 ->   14) [( 2.0249e+03, 9.9913e-01,-8.3609e-01, 2.0249e+03), p^2=-4.6566e-10, m= 0.0000e+00] (  0,653)
[B] 4 u                   66 (  10 ->   14) [( 2.0047e+02,-8.8966e-01, 2.1595e-01, 2.0047e+02), p^2= 7.2760e-12, m= 0.0000e+00] (652,  0)
```

Therein, the first line contains information on the vertex (`Blob`). First, in angle brackets, its status is displayed, which is of no consequence once the event is written out. Second, in parentheses, its number, name, incoming and outgoing particle multiplicities and spatial position are detailed. This is followed by the associated particles. Every particle is listed in turn. For each particle the data displayed is its status identifier (both the character in brackets as well as the following number), its name, its number (each uniquely identifies a particle in each event). Its production and decay vertices, if existent, are displayed in parentheses, followed by its momentum (in the convention (E,p_x,p_y,p_z)) and its colour indices.

Can you identify the initial state protons and all the particles leaving the interaction as free stable final state to hit the detector? Hint: particles without a production vertex were not produced within the calculating, this constitute the initial state. Similarly, particles without a decay vertex do not decay within the calculation but leave it as stable final state.

Try to find the hard interaction vertex. Recalling the lecture, it is the first thing calculated and therefore the vertex has the number `1`. It is called `Signal Process`. Then try to work your way outwards, along the initial state particles towards the incoming protons, and along the final state particles through the hadronisation towards the final stable set of hadrons, photons and leptons. A visualisation of the SHERPA event record is displayed in Fig. 1, do you find the same structure? Which of the displayed information is physical, which is depending on the specifics of how the calculation is set up?

2.2 Playing with the setup

The setup used so far computes Higgs production in gluon fusion. Let us now try modify this setup, such that the Higgs is produced in association with a Z boson. To that end, change the process declaration to

```
Process 93 93 -> 23 25;
```

Running SHERPA again, it will generate the new source code for the new processes which needs to be compiled. When this is done SHERPA can be restarted

```
Sherpa -f Run.dat
./makelibs
Sherpa -f Run.dat | less -R
```

Now, SHERPA will not need to recalculate the multi-parton interaction cross sections as they have been stored in `MIG_P+P+_8000_ct10_1.db` and can be read in now.

Try now to implement $pp \to t\bar{t}h$ production. What is the corresponding PDG ID of the top quark? Which process has to be declared? Do you see an increase of the number of stable hadrons produced? Can you estimate the size of the increase?

3 The event stages

In the second part of the Monte Carlo tutorial we are going to examine which parts of typical observables are effected by the physics simulated in the different event stages. We will further try to understand which types of observables are susceptible to physics at which scales. Representative observables under consideration are the transverse momentum of the Higgs boson, its rapidity, the transverse momentum of the leading jet, and its rapidity. As an example we are going to examine Higgs boson production in gluon fusion. All setups for this purpose are contained in the `Day2` subdirectory, thus

```
cd Day2
```

3.1 The hard interaction

At first, we are going to study how our chosen observables are described by only calculating the hard interaction at leading order accuracy. Thus, no parton shower is used to approximate higher orders, no corrections through multi-parton interactions or hadronisation are effected. The input card for this first step is called `LO.dat`. Have a look at it and try to identify which process is calculated.

First, again, the source code for the matrix elements needs to be created and integrated

```
Sherpa -f LO.dat
./makelibs
Sherpa -f LO.dat
```

Now we can start generating events and let the analysis fill the predefined histograms. Let us start with `10k` events.

```
Sherpa -f LO.dat EVENTS=10k
```

As defined in the input card, the RIVET analysis package is used to create histograms in the YODA format. At the end of the run the histograms are written in the file `LO.yoda`. It is a plain text file and can be opened in a text viewer, e.g.

```
less LO.yoda
```

for the transverse momentum of the Higgs boson. It is much more useful, however, to plot these histograms. To this end, use the supplied `makeplots.sh` script, which employs `gnuplot`, and takes as argument the histograms you want to have included in your figures. Thus, execute

```
rivet-mkhtml --mc-errs LO.yoda
```

Now open the newly created webpage

```
firefox plots/index.html &
```

and browse through the different observables. Do you understand why the observables are described the way they are?

3.2 Parton showering

As a second step we take the leading order calculation from the previous section and complement them with approximated higher order corrections from the parton shower, thereby evolving the partons from the high scale at the hard interaction and to low scales near the transition to non-perturbative QCD. The input card to be used now is `LOPS.dat`.

As the matrix elements have already been created and integrated in the last step we can start the event generation straight away.

```
Sherpa -f LOPS.dat EVENTS=10k
```

The resulting histograms are written to `LOPS.yoda` this time and can now be compared to the pure leading order calculation of the hard interaction

```
rivet-mkhtml --mc-errs LOPS.yoda LO.yoda
```

Simply update the webpage as the plots contained were updated with the new information. Which observables are affected by the added corrections? Where do you expect the collinear approximation of the parton shower to give a reliable description?

3.3 Multi-parton interactions

Now multi-parton interactions are added. As discussed in the lecture and depicted in Fig. 1, these consist of additional softer parton scattering, supplemented by parton shower corrections of their own. The relevant input card now is `LOPSMI.dat`.

Again, we can reuse the matrix elements and integration results from the previous runs, but, as in `Day1`, the multi-parton interactions grid needs to be generated. Thus, execute

```
Sherpa -f LOPSMI.dat EVENTS=10k
```

and the grid is created before 10k events are generated. The resulting histograms are written to `LOPSMI.yoda` this time and can now be compared to the above two caluclations with

```
rivet-mkhtml --mc-errs LOPSMI.yoda LOPS.yoda LO.yoda
```

Simply update the webpage as the plots contained were updated with the new information. Which observables are affected by the added interactions? Why are these observables affected and others not?

3.4 Hadronisation and hadron decays

Let us now complete the event generation by converting all partons into hadrons and then decay them until only stable (on time scales of the order of the size of the detector) remain. The relevant input card now is `LOPSMIHAD.dat`. Generate 10k events and add the resulting histograms to your plots using

```
Sherpa -f LOPSMIHAD.dat EVENTS=10k
rivet-mkhtml --mc-errs LOPSMIHAD.yoda LOPSMI.yoda LOPS.yoda LO.yoda
```

Which observables are affected by the added hadronisation and subsequent decays? Why are these observables affected and others not?

4 Further improvements

In the previous exercises we have examined how the individual stages of event generation (corresponding to physics happening at different momentum scales) impact on different typical observables. In this final section of the tutorial we investigate how improvements on the description of the perturbative stages of a hadron collider event affect our observables. In particular, we are replacing the leading order calculation of the hard scattering by a next-to-leading order calculation. All relevant setups are also contained in the `Day2` subdirectory, thus

```
cd Day2
```

from the top level directory.

4.1 The hard scattering at next-to-leading order accuracy

Elevating the leading order computation of the hard scattering to next-to-leading order requires the computation of both virtual and real emission corrections. Both are available within Sherpa. However, new matrix element libraries need to be created and integrated, thus run

```
Sherpa -f NLOPS.dat
./makelibs
Sherpa -f NLOPS.dat
```

Now events can be generated. As there are more components to this calculation, we increase the number of events to `100k`

```
Sherpa -f NLOPS.dat EVENTS=100k
```

They can now be compared to the leading order result

```
./makeplots.sh -o plots-nlo --mc-errs LOPS.yoda NLOPS.yoda
firefox plots-nlo/index.html &
```

Which of these observables are actually described at next-to-leading order accuracy? How can we examine the accuracy of the calculation? Were would you expect the next-to-leading order calculation to be a good approximation?

References

[1] T. Gleisberg, Stefan. Hoeche, F. Krauss, M. Schonherr, S. Schumann, et al. Event generation with SHERPA 1.1. *JHEP*, 0902:007, 2009.

[2] F. Krauss, R. Kuhn, and G. Soff. AMEGIC++ 1.0: A Matrix element generator in C++. *JHEP*, 0202:044, 2002.

[3] Tanju Gleisberg and Stefan Hoeche. Comix, a new matrix element generator. *JHEP*, 0812:039, 2008.

[4] Steffen Schumann and Frank Krauss. A Parton shower algorithm based on Catani-Seymour dipole factorisation. *JHEP*, 0803:038, 2008.

[5] S. Alekhin, G. Altarelli, N. Amapane, J. Andersen, V. Andreev, et al. HERA and the LHC: A Workshop on the implications of HERA for LHC physics: Proceedings Part A. 2005.

[6] Jan-Christopher Winter, Frank Krauss, and Gerhard Soff. A Modified cluster hadronization model. *Eur.Phys.J.*, C36:381–395, 2004.

[7] Marek Schonherr and Frank Krauss. Soft Photon Radiation in Particle Decays in SHERPA. *JHEP*, 0812:018, 2008.

[8] Sherpa-2.1.1 online Manual. *https://sherpa.hepforge.org/doc/SHERPA-MC-2.1.1.html.*

[9] Sherpa 2.1.1 Manual. *open with `firefox install/share/doc/SHERPA-MC/Sherpa.html` &.*

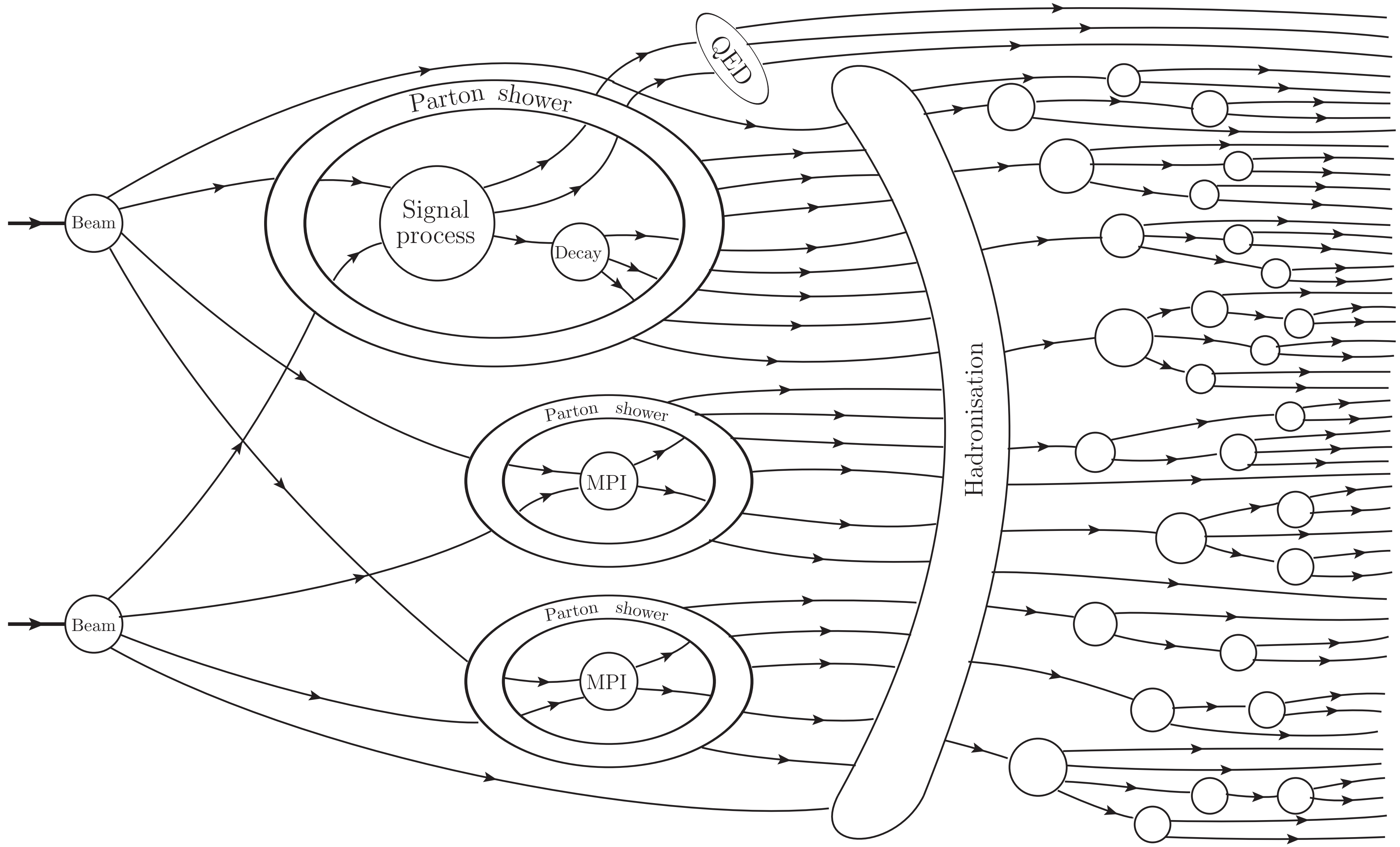

Figure 1: Idealised visualisation of a SHERPA event record.