EIC simu decision input

	Fun4All	**g4e**
Maturity	Proven code base that's been used to design a detector coming under construction. Synergy between experiments, taking advantage of 8 years of experience 100% of described functionality is implemented and in active use. It is impossible to compile outside of BNL Even with the emailed instructions from developers (no documentation?) No one was able to independently compile fun4all	A mix of recent, new and not yet existing code. One of Elke's new EIC postdocs (Markus met him on Friday), who has been working with his own standalone G4 simu, assessed g4e and prepared these slides. Their strong message comes from his assessment, not any bias. The postdoc is definitely out of context, disinformed and biased because of this. Still, his proposal on the slide #6 is exactly what was in the presentation in G4E implementation. Basically in "Lumi monitor from the software point of view" part of his presentations the postdoc confirms, that G4E is what he needs for the lumi monitor. New development solely optimized for EIC. P
User base	Large existing user base performing detector and physics studies. 30-40 people across 8 institutions (listed below). Incorporates many test beam analyses: easy to use for verification of test beam data, feeds into more accurate simu How many of those users are EIC users?	JLab only? G4E backend by developers serving software for Halls A, B, C, D. Only the list of collaborating institutions will be around 100. How this relates to EIC and Yellow Pages users?
Collaborative contributions	jleic detector implemented (between Jul & Sep) jleic beam line magnets implemented (Nov) geometry hierarchy implemented (between Jul & Sep) Sep graphics issue promptly fixed Oct questions on C++ G4 geom support addressed EicRoot - Jana2 integrated (between Jul & Sep) Which is a success of Jana2. Why it is mentioned as a success of fun4all?	Little apparent activity between Jul and Sep? Response to Sep agreement in favor of Fun4All was to proceed with g4e, leading to Monday's presentations with no previous circulation of slides True. But the question relates to the organization of the meeting rather than to G4E
Contributing effort	Quick effective response to feedback has been demonstrated. Can draw (to a certain degree) on new postdocs working on EIC simu - same for G4E Establishing how the code base will be organized for collaborative development will be a priority	"Anticipated 20-30 developers to start committing to the codebase" How left and right relates? One can switch those columns without any difference.
Generator input	All event generators from eic-smear available via generic eic-smear interface + sartre. Pythia6/8 and sartre run inside process loop, no separate running into some ascii file needed. eic-smear is not opening HepMC files.	Particle gun, cone, LUND (Pythia6), HEPMC (Pythia8, Herwig), BEAGLE It ships universal file opener which can open any file eic-smear reads and more. It is faster, highly customizable, etc. It is a good example to have extended explanation of proposed and fun4all approaches
Pileup	Multiple event sources can be used in parallel (needed for pile up studies) Embedding into existing hits files is supported	multiple text files can be read in parallel? Yes. Done in Geant4 through G4VUserPrimaryGeneratorAction calling multiple generators.
Geant4 geometry hierarchy	Implemented hierarchy under world volume in response to our discussions.	Hierarchy under the world volume
Modular subdetector geometries	~~Fully modular not modular~~ not modular at all, pretty highly coupled software that one can't take apart and used separately or use parts of it in other software so easily. , supports geometries in G4 C++, ROOT macros, GDML, … ~~Can do rapid prototyping in quick macros without writing any code~~ macros **ARE** the code. Example! Moreover, with very poor documentation, and implement full detail in G4 C++.	G4 C++ based geometry. 1. As stated "is OK" in July meeting minutes. 2. Even now it ships VGM that allows also TGeo import and others. Being close to raw Geant4 opens a possibility to put DD4Hep or whatever community solution for geo exchange.

	Modules are flexibly composed into a full detector via ROOT macro. A blob of macros with code and constants referencing each other, which are hard to debug and navigate. Very non modular and requiring knowing Geant4 compiled part, ROOT but also poorly documented fun4all “language” **Moreover heavy reliance on ROOT macros makes it very difficult to fully integrate it with Jupyter and python and is rather bad than good.** **Here is an example** Large selection of mature detector implementations. Not for EIC.	3. Defined interfaces allows to use different technologies. E.G. Geant4 ships Python bindings. As a proof of concept it was checked that it is possible to create sub detectors in python and use it from C++. It might be very useful for students and postdocs as well as can be used in Jupyter tutorials. 4. Proof of concept on root macros is done too. So there is actually a full spectrum of possibilities here. ~~Header files included into a giant #ifdef steered detector construction.~~ It is JLEIC dedicated **but not G4E** issue and is being fixed. No generic detectors (cylinders/boxes/cones which can be positioned at will in arbitrary numbers) what? ~~No GDML support.~~ Up ~~SubDetectorInterface presented in the slides does not exist~~ - Those where screenshots on the slides. ~~while the functionality does exist in Fun4All~~ - In terms of there is also C++ inheritance used in Fun4All. In reality, goals are different.
Geometry parameters	Flexible easily modifiable params in ROOT macros, or can bake into G4 C++ code if you choose. All is the same C++ code and C++ programming just scattered between compiled and dynamically invoked ROOT .C macros part without any chance of clear separation of logic and configuration. It is as “flexible” as an old C++ code. There is just nothing else. And disadvantages of how it is done are written above. Full persistency of parameters in output. Another good example	Hard coded parameters in C++ No persistency of parameters in output There is no one the best single option for this. All options have a lot of compromises. With G4E we have a very good chance to choose one, that community finds best for EIC and current tasks.
Stepping action	Detector specific stepping actions	~~A single stepping action serving all detectors. Unclear this preserves all the functionality of using detector specific stepping actions and leaving their appropriate invocation to G4. Volumes are identified by name, cut and paste detectors will likely collide~~ General stepping actions are not recommended for physics and users as they should work on level of hits and sensitive volumes. Each detector is recommended to have its own SD processor. Still there is a possibility to subscribe for a stepping action for a dedicated detector or put your own stepping action without interfering with other stepping actions.
Multithreading	On the ToDo list, but not high, because not needed in the near future. No memory issue.	SteppingAction does not look thread safe EventAction is not thread safe
MC truth	Full truth info available (every G4 hit can be traced back to any of its parents, any parent can be tracked to detector hits). Full persistency of MC truth info.	Incomplete MC truth recording, and no output of MC truth traceability
Physics lists	easily modified via macro with no recompile How many this saves and how often one changes the lists on the fly?	can’t be changed on the fly without recompiling If really a requirement it is doable via standard geant4 tools (messengers). On my laptop: Recompile time after changes - 17s Recompile and run (no changes) - 2s Root cold start with a single empty macro - 11s Root start with a single empty macro - 2s
Simu output	Supported, includes all config , simu and MC truth data. Not opposed to have a module which writes whatever output in whatever form is needed. No uproot support. Very problematic for such tools. Relies on C++ classes automatic IO which means it is hard to have controllable specification. Configs are given in strings with no documentations like hcal->set_double_param("light_balance_outer_corr", NAN); Those parameters are scattered between	Supported, simu and MC truth data. Not opposed to have a module which writes whatever output in whatever form is needed. Because of clear code/configuration separation there is no such hot problem of saving configs. Flat trees with nice structure for uproot+python and root DataFrame quick analysis.

	compiled C++ and macro C++ codes which are scattered between 2 different repos.	Considering there are 2 ways to save different details in G4E and one more in Geant4 itself. It would be beneficial for the software group to come up with event record and output specification.
Simu features	Neutron flux can be "measured" Geantino scans to verify detector positions Black holes to find e.g. calorimeter leakage Evaluators exist which you need to actually evaluate the sims (how is this acceptance coming?)	
Reconstruction. Thomas's mandate was for simulation framework ASAP, but the needs don't stop at simulation. And it is now Nov, not July. In a few months we'll be in an intensive detailed detector/physics study phase; reco will be important. Monday's presentations put reco on the table, and it belongs there as a point of comparison.	Mature support for full simu/reco chain	Simu only Reco is done in a separate package. Compared to fun4all we have a modular stack of separated software. JLEIC analysis where done with Genfit, Rave (same used for fun4all) and FastJet. Ejana not only have plugins for Genfit+Rave stack and now Alexandar put EicRoot there, there are fast simulation (eic-smear), python bindings and Jupyter GUI presented on Paris meeting. It is just a separate software so we didn't discuss it during full simulation topic. There shouldn't be a problem to read G4E output and run through fun4all reco if it is modular, right?
Parameterization development	Supports the full chain through reco that is required for developing detector parameterizations for fast simu. eg. Cannot parameterize tracking efficiency without reconstructing tracks.	Supports user workflow oriented user experience with support of Jupyter lab, python orchestrator etc.
Code QA	very well debugged (valgrind, insure, coverity, scanbuild, clang, cppcheck) fixed seed -> 100% reproducibility **How one debugs this? How well those tools designed for enormous dynamic ROOT macroses (which are also randomly load each other and libraries on the fly) ?**	valgrind reports 1.5mio errors for 2 events (that's not unusual for new code but should be dealt with) Gazillions of warnings during build Thousands of lines of output on the screen I'm not sure, what was examined and how it relates to the proposed G4E for YP
User interfaces, front ends	Jin Huang is integrating this into Jupyter, some test beam analysis is in a Jupyter notebook, we will be able to run hijing soon where we have to farm out the processing.	Integrated with Jupyter Initially developed with various integrations in mind.
packaging, dist/install, container support, documentation, examples, tutorials	email below	https://g4e.readthedocs.io/en/latest/
Repository	https://github.com/sPHENIX-Collaboration/Fun4All Here are more https://github.com/sPHENIX-Collaboration/coresoftware https://github.com/sPHENIX-Collaboration/macros Macros and compile parts are scattered between repos. Very difficult to navigate. Impossible to take and build.	https://gitlab.com/jlab-eic/g4e/
Future path	No desire or expectation to carry this framework into the EIC era. It's a pragmatic, mature tool to use for immediate needs, freeing us to design/develop a new framework as one team.	**The described future is G4E reality**.
Priority todo's if selected	Identify people to help with the top priorities (cf. aforementioned EIC postdocs) Establish how the code base will be organized for collaborative development Packaging, download/build/install/run, documentation, examples, support	

pink - Sergey Furletov
blue - Dmitry Romanov

Fun4All howto email

---------- Forwarded message ---------
From: **pinkenburg** <pinkenburg@bnl.gov>
Date: Mon, Oct 28, 2019 at 8:35 PM
Subject: Re: EIC Software meeting, October 28, 1:00 p.m. (EDT)

To: Markus Diefenthaler <mdiefent@jlab.org>, Alexander Kiselev <kisselev@mail.desy.de>, Andrea Bressan <andrea.bressan@ts.infn.it>, David Lawrence <davidl@jlab.org>, Dmitry Romanov <romanov@jlab.org>, Julia Furletova <yulia@jlab.org>, Kolja Kauder <kkauder@bnl.gov>, Makoto Asai <asai@slac.stanford.edu>, Nathan Brei <nbrei@jlab.org>, Sylvester Johannes Joosten <sjoosten@anl.gov>, Torre Wenaus <wenaus@gmail.com>, Wouter Deconinck <wdconinc@jlab.org>

Hi folks,

let me put this in front where everyone reads it. As Markus rightly pointed out - I have been working with this for a while and I am sure I forgot some important details in the following. It should really work out of the box, let me (using this list) know if you get stuck. Here is the link I used to install cvmfs on my ubuntu desktop: https://cernvm.cern.ch/portal/filesystem/quickstart

There are a few approaches one can take, the "easiest" is to follow
https://github.com/sPHENIX-Collaboration/Singularity
and download and run the updatebuild.sh
which creates local copies of our cvmfs volumes.

If you use cvmfs, I updated our cvmfs volume /cvmfs/sphenix.opensciencegrid.org which is visible on the OSG which should be accessible without keys. It contains all 3rd party libs and a complete build of our software. If everything works according to plan doing for (t)csh:

source /cvmfs/sphenix.opensciencegrid.org/x8664_sl7/opt/sphenix/core/bin/sphenix_setup.csh -n
for bash:

source /cvmfs/sphenix.opensciencegrid.org/x8664_sl7/opt/sphenix/core/bin/sphenix_setup.sh -n

should set everything up to run (in our container). CentOS7 is close enough to SL7 and should work if the necessary system libraries are installed.

No matter what you do the next steps are:

Once this is functional you should be able to run our tutorials
git clone https://github.com/sPHENIX-Collaboration/tutorials.git

and the Fun4All_G4_*.C macros from
git clone https://github.com/sPHENIX-Collaboration/macros.git
(in the macros/macros/g4simulation subdir - the gleic Fun4All macro is in macros/macros/g4jleic)

If all of this works, the build script - convoluted perl, touch at your own risk :) in
git clone https://github.com/sPHENIX-Collaboration/utilities
(utilities/utils/rebuild). The setup script should set all its environment variables and with

./build.pl --workarea="<wherever you want to install it>"

it will create a "new" subdir in the workarea and you'll find the sources in the "new/install" subdir. Depending on the cpu - the build might take 2 hours, you can tail the log new/rebuild.log to see where you are.

If you want to try your installation source the setup script as mentioned above for the initial environment, then do

export OFFLINE_MAIN=<wherever you want to install it>/new/install
export ONLINE_MAIN=<wherever you want to install it>/new/install

and source the setup script without -n. Your LD_LIBRARY_PATH (and PATH and ROOT_INCLUDE_PATH) should now point to your installation.

Last not least I attach the keys for the /cvmfs/sphenix.sdcc.bnl.gov volume mentioned in the singularity page which gives you access to a large variety of builds (archived builds, new G4 version, debug build, gcc 8.3, whatever we feel like build) - sorry the insure builds needs a license only available when running at BNL. But if needed I can copy those to the OSG volume (the OSG volume is meant for production, so the initial plan was to put stable production builds there but it's up to us what we do with this). These files go (under ubuntu, no idea about MACs) under /etc/cvmfs in their respective subdirs:
/etc/cvmfs/keys/sdcc.bnl.gov/sphenix.sdcc.bnl.gov.pub
/etc/cvmfs/domain.d/sdcc.bnl.gov.local
/etc/cvmfs/config.d/sphenix.sdcc.bnl.gov.local

and now off to the kangaroos :)

Chris

Fun4all EIC institutes

Institutions actively working on spin and eic with Fun4All:
BNL
LANL
Glasgow
Stony Brook
Iowa State
U of Michigan
Georgia State
RIKEN