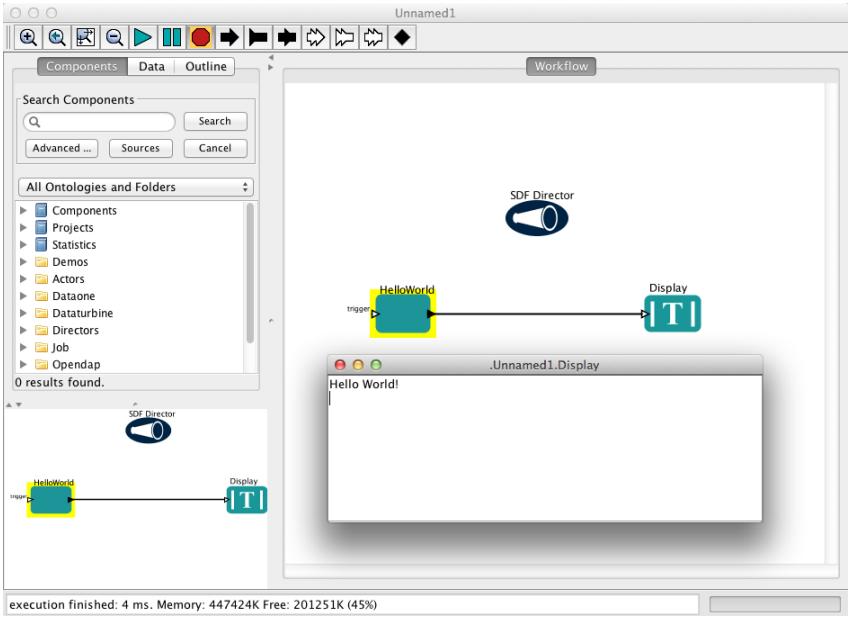
Automating fusion research through

Scientific workflows

Evguenia Usoskina



This project involves source code modification for an open sourced visualisation tool VisIt and integrating the changes into a an actor for a scientific workflow software Kepler. The final outcome is a software patch, the application of which will aid researchers in visualisations of fusion reactor simulations.

The essence the project is adding new functionality to existing software. However, as software is only a means to an end, a question arises right away: what is it going to be used for? The answer is an exciting one: for simulations of a fusion reactor! Fusion is a fascinating and currently a very ”hot” subject.

The reactor simulated in the project is tokamak: a word derived from Russian for ”toroidal chamber with magnetic coils”, which is essentially a very large doughnut-shaped vacuum chamber. A gas of ionised deuterium and tritium particles is pumped into the vacuum, until a high density is reached. Then, a beam of high velocity neutral deuterium particles is fired into the chamber to collide with the gas particles, increasing the temperature of the system greatly. A pulse of electric current is then induced, heating the system even further, and turning ionised gas into plasma. The high temperatures of around 100 million degrees Celsius cause the deuterium and tritium particles to fuse together, thus releasing energy, just as happens inside stars. Joint European Torus (hereafter affectionately called JET) is the world’s largest tokamak fusion reactor located in Oxfordshire, UK. From the official JET website, it ”investigates the potential of fusion power as a safe, clean and virtually limitless energy source for future generations”. To see what JET looks like in action, click here. However, not all the research is done experimentally - simulations also play an important role in understanding the science. While simulating plasma behaviour is relatively easy, problems arise when trying to model what happens near the walls of the tokamak: the high voltage drop causes a so-called ”’singularity” - something that is virtually impossible to model. That is the sort of cutting-edge science we will be working with - assisting in visualising JET reactions using HPC.

There are numerous benefits to using nuclear fusion, as it is one of the most promising options for generating large amounts of carbon-free energy in the future. According to current estimates, the costs of using fusion energy can be comparable with that of fission, renewables and fossils.

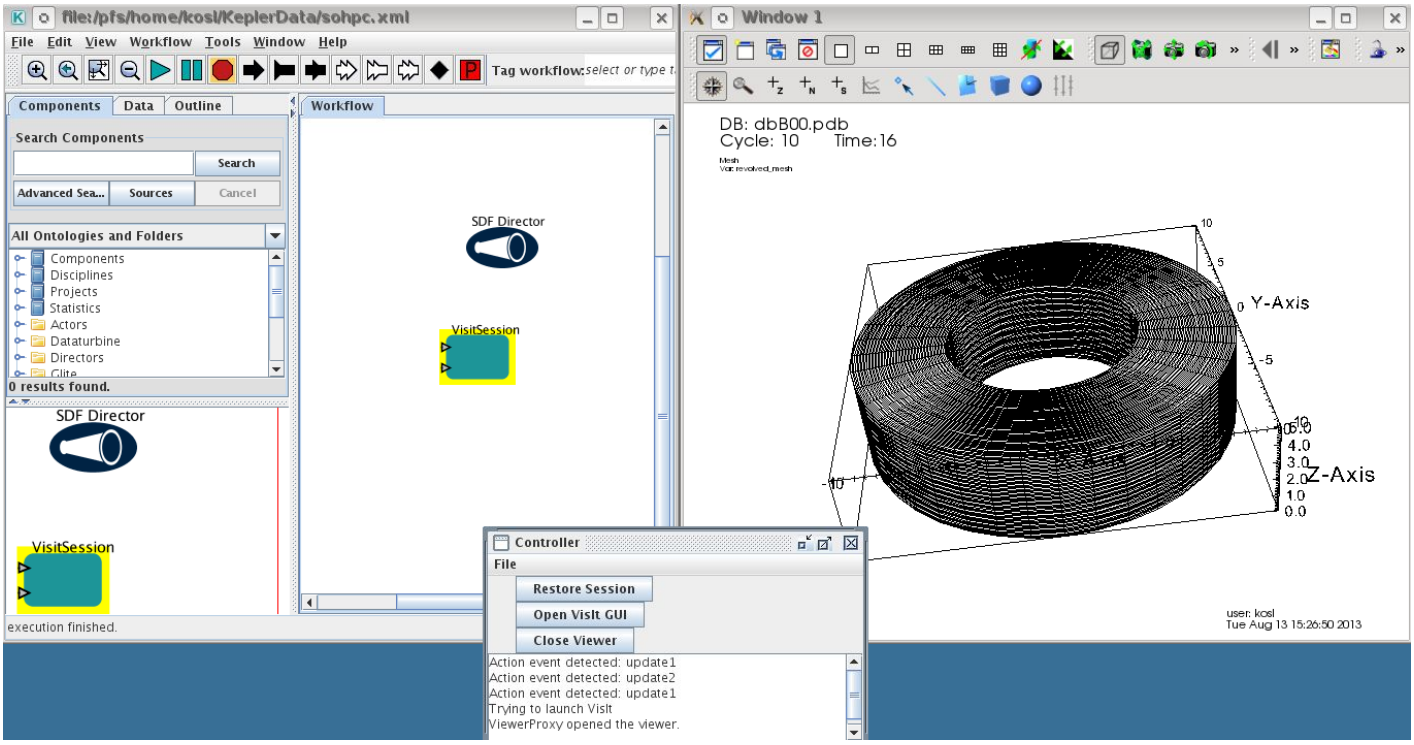
As described earlier, the fuel used in fusion is tritium and deuterium, and their supply is essentially inexhaustible. While deuterium can be extracted from water, tritium can be produced from lithium, which is found in the earth’s crust.

In addition to that, there are numerous other advantages of fusion, namely:

- No emission of harmful toxins
- Only helium is produced in the process, which is already abundant in the atmosphere and will not contribute to global warming
- Efficiency: 1kg of fusion can provide the same amount of energy as 10 million kg of fossil fuels
- Safety: as only small amounts of fuel are used at a time, large-scale nuclear accidents are virtually impossible

VisIt

VisIt is an interactive visualisation tool used for graphical analysis of scientific data. It is a client-server application: the client sends the commands for visualisation to the server, which computes the data using high-performance parallel resources to generate the visualisation that is shown in the client’s display. Thanks to this operation, it can handle very large data set sizes in the terascale range. For this reason, VisIt is ideal for visualising complex simulations such as the ones of the tokamak fusion reactor, which is precisely the one I am going to work with. It is also possible to run it



A simple workflow created with Kepler

on your own computer with small data sets in the kilobyte range.

VisIt's paradigm is to create one or more plots in a visualisation window. If there is more than one visible plot at the same time, they are combined. It is possible to apply operators to the variables used to create the plots. One example of an operator could be the Slice: if you have a 3D plot you can "cut" a slice of it and just show that slice. In that case, it will use a portion of the data to create the plot.

Kepler

SWFs are used to automate and record data transformations, and in my case come in the form of software. For example, imagine a researcher who wishes to simulate molecular dynamics. This works in the following way: he has some initial input data (i.e. the initial positions of particles, their velocities, etc.), which he then feeds into his program. For the purpose of scientific workflows, we can treat the program as a black box - we don't care how it performs the data transformation, or what language it is written in - all that concerns us is the output. Thus, for the molecular dynamics example we want the output to be the states of the particles after a given time period. This is an example of a very simple scientific workflow - we have transformed the initial data. However, usually more than one such transformation is required, where data is passed between different programs, often written in different languages.

The main goal of SWF is to ease scientific research. If a simulation requires multiple data manipulations (as real simulations almost always would!), one could manually "move" the data from one program to another, but not only is that tiresome, but also very prone to mistakes. On the other hand, if one first creates a workflow, the process becomes automated - all that is needed is to "feed" in the data and receive the final output, much like merging all the little "black boxes" (separate programs) into one big "black box" (the workflow), which can be "zoomed into" and edited, if desired.

Another goal of SCWF, often overlooked, is provenance. Provenance means providing evidence of reproducibility of scientific findings. Scientific work is of good provenance if it is documented in such detail that it can be reproduced without doubts. Data provenance means that is it archived, recorded how it was collected, and under what conditions transformations were done. In principle, evidence (input data) should not be contaminated in any case. All of this may sound like some definitions from archaeology, art history, bio-informatics and other fields of science where data is difficult to collect. However in computational science provenance is just as important Đall the findings have to be reproducible.

Despite computing being heavily used in the business field, a major portion of it is coming from the sciences. What is more, scientific research is becoming more and more computationally intensive now - it is no longer possible for a scientist to sit in his office with a just logarithmic ruler and perform calculations. That is why workflows are an essential tool in High Performance Computing (HPC), where the submitted jobs can be very complicated and take a very long time (thus making mistakes becomes progressively more expensive, and automation of the process is essential).

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis,

diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Over 100 different scientific workflow products now exist, providing different functionalities, but the one I am working with is Kepler. The image shows my first workflow created in Kepler - a ”Hello World!” program I wrote. What happens is the ”actor” (or the black box), called HelloWorld outputs the desired sentence, and the display actor (another black box) displays that output.

Figure 1: As far as scientific workflows are concerned, all the calculations are black boxes - the user only cares about the input and output!

Code modifications

VisIt is a visualisation software used for presenting data in an intuitive way. It is widely used by scientists, specifically for visualising very large datasets that require using HPC and parallelising tasks.

VisIt is open sourced, which means that anyone is free to look at the source code and modify it. To clarify the latter - one is free to modify the code for his or her own use, however, if one feels he has made an improvement that could greatly benefit the whole community, the modification can be submitted to the main developers of the code in Lawrence Livermore National Laboratory. This is done in a form of a patch - a diff file, containing the differences between your version and the original source code. The so-called "benevolent dictators" (ie. the main developers) will review and make the final verdict upon all the suggested changes, and if the changes are approved, the patch will be publicly released.

Unfortunately, software development follows Murphy’s laws when it comes to bugs - if a program is useful, it will have to be changed. Note that here I’m referring to bugs not only as a software failing to perform something correctly, but also to the case where an option to perform something is not there (the software failing to have a functionality it is expected to have). The first version of VisIt was released in 2002, while the current version is 2.6.2, and it’s no surprise that new features are continuously added.

My task involved fixing a few of such bugs/functionality deficiencies. More specifically, I started by modifying the VisIt Java library. While VisIt itself is written in C++, this library allows for using VisIt through a Java program. The biggest problem I set out to tackle was the fact that if the user manually closed VisIt, Java didn’t register it, fruitlessly continuing its communication attempts, and hung, getting no reply.

So, one may wonder why would one want to write a Java library for a software written in C++? Kepler uses actors that are written in Java. One of these is a VisIt Kepler actor, which lets the user visualise the desired dataset with VisIt. ThatÕs where my modifications come in handy - if the VisIt session is closed manually, it is still desirable to be able to open it again, without stalling the workflow.

After editing and re-factoring code, following functionalities have been added to the VisIt Kepler actor:

1. **additional input** - the user can now specify more initial parameters, such as the inclusion of GUIs, previous session name, etc.

2. **manual closing** - the user can now close VisIt and re-open it again

3. **session restore** - it is now possible to restore a previously saved VisIt session

4. **customised Graphical User Interface (GUI)** - allows the user to easily control VisIt

PRACE SoHPCProject Title
Visualization support for scientific workflows with VisIt Kepler Actor

PRACE SoHPCSite
University of Ljubljana, Slovenia

PRACE SoHPCAuthors
Evguenia Usoskina

PRACE SoHPCMentor
dr. Leon Kos


Evguenia Usoskina