20th International Conference
NUCLEAR ENERGY FOR NEW EUROPE 2011
Bovec /Slovenia /September 12-15

Towards 3D Time Dependent Visualization within ITM-TF infrastructure

Leon Kos[1], **Olivier Hoenen**[2], **Simon Kulovec**[1], **Philippe Huynh**[3], **Jožef Duhovnik**[1], **Frederic Imbeaux**[3]

[1]University of Ljubljana, Faculty of Mech. Eng., Aškerčeva 6, SI-1000 Ljubljana, Slovenia,
{leon.kos, simon.kulovec, jozef.duhovnik}@lecad.fs.uni-lj.si

[2] Strasbourg University, IRMA, 7 rue Descartes, 67084 Strasbourg, France,
olivier.hoenen@unistra.fr

[3]CEA, IRFM, F-13108 Saint Paul Lez Durance, France,
{philippe.huynh, frederic.imbeaux}@cea.fr

ABSTRACT

Complex visualizations are foreseen for upgraded fusion codes under the EFDA Task Force on Integrated Tokamak Modeling (ITM-TF). We present upgrades of the work that was carried under project EUFORIA (EU Fusion for ITER Applications) and then integrated into ITM-ISIP (Infrastructure and Software Integration Project) set of tools. Scientific workflows, managed by Kepler engine, are executed with encapsulation of fusion codes on variety of computing resources. For support of visualization on common fusion database layer, an adaptation of visualization tools are required. Time dependence of 3D results are normally shown in cross sections as animations. This is impractical for real scientific work as it is hard to quantitatively measure how parameters of interest evolve through time. Fusion experiments and simulation are dumped as shots into ITM database through UAL (Universal Access Layer). Visualization software can with the help of "representation fields" show any slice data from UAL at point of time, but not as time dependence. To provide such views, upgrade of visualization components is required. VisIt software with UAL plugin and actors developed are used as visualization engine under Kepler workflows.

1 INTRODUCTION

Integrated Tokamak Modeling Task Force (ITM-TF) is aiming to provide a unified framework for fusion codes that already exists in decoupled state, each providing modeling of some elementary physics problem. In order to synchronize numerical simulations with experimental data that is already captured into scientific databases like MDSplus (or MDS+), HDF5 [1, 2] further encapsulation of databases were proposed by ITM-TF to facilitate specifics of fusion modeling. Namely, *shot* and *run* are typical selectors for given ITM tokamak-database, while the underlying MDS+ and HDF formats are structured in such way, that objects stored follow prescribed structure of elementary physics problems. Description of data is done through the

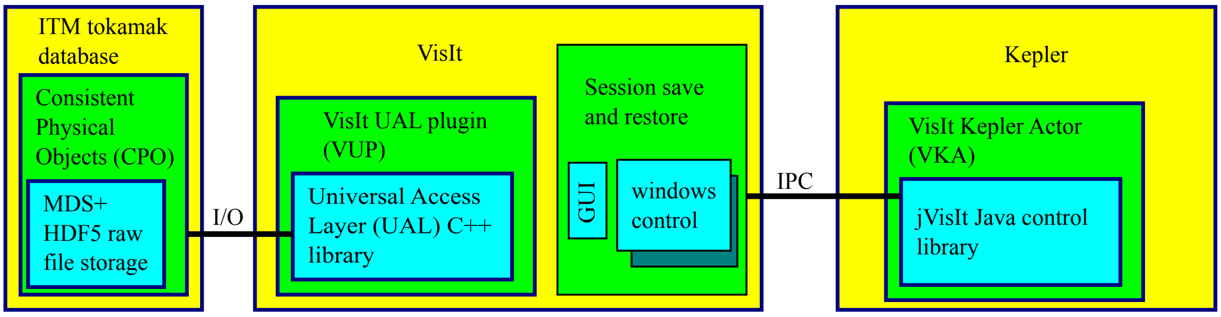
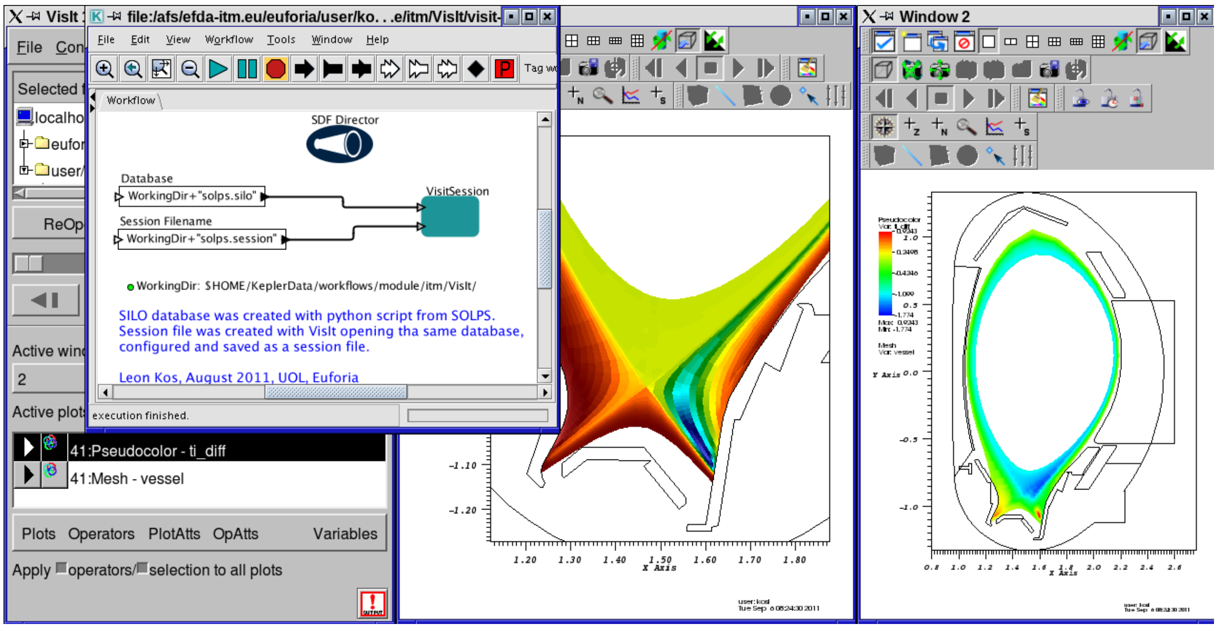

ITM tokamak database
Consistent Physical Objects (CPO)
MDS+ HDF5 raw file storage
I/O
VisIt
VisIt UAL plugin (VUP)
Universal Access Layer (UAL) C++ library
Session save and restore
GUI
windows control
IPC
Kepler
VisIt Kepler Actor (VKA)
jVisIt Java control library

Figure 1: Schematics and communication of basic components for visualization with VisIt under Kepler workflows.

notion of Consistent Physical Object (CPO) data model [3] with the aim to unify description of specific physical properties used within experiments and/or numerical simulations. Such encapsulation is necessary to maintain *consistent* description of “objects” that are stored/retrieved from ITM database using various “fusion codes” in *Integrated Tokamak Modeling* pursuit. Consistency enforced by CPO data model requires that existing codes need to be “adapted” to be able to retrieve/store results into prescribed CPOs. This task requires developer’s knowledge of the code and CPOs description that needs to be mapped to each other in enforced units (MKSA and eV). After (necessary) adaptation, code can be executed as a back-box within “scientific workflows” that couples different codes in a compatible way and sequence.

As the adapted code can store results in a compatible way (in CPOs) during simulation, there is a possibility to visualize them using general tools that are able to read data described by CPOs. Such CPO-aware visualization tools were nonexistent and developed [4] by EUFORIA/JRA4 project where VisIt software [5] was selected as a general visualization open-source software maintained by LLNL. VisIt is a popular and extensible visualization software that provides various representation modes and data post-processing in interactive and batch/scripting mode. “Adaptation” of VisIt to be able to read CPOs was developed via plugin in similar way like any code adaptation. To simplify CPO access and consistent maintenance, ITM-TF uses XML description of CPO and XSLT translation that generates necessary libraries and CPO description (include files) for different programming languages. Besides language compatible CPO description one needs also library routines that helps retrieving and storing CPOs in a “physical” database. This library named as *Universal Access Layer* [6] (UAL) provides a set of routines for “middle-ware” database access tailored especially for CPO transfer. UAL is “universal” in a way that different languages use the same routine names and that data transfer is simplified without compromising efficiency when using coupled codes in a workflow. To simplify workflow modeling ITM-TF selected Kepler [7] software that provides graphical user interface (GUI) for describing information flow between computing units called “actors”. An actor can be *encapsulated* fusion code or, in our case, VisIt visualization tool. Fig. 1 shows schematics of the layers in tools introduced required just for visualization part within workflow. Specialized *actors* for Kepler has been developed [4] to provide direct visualization acting under workflows directly. *VisIt Kepler Actor* (VKA) provides VisIt launching and controls UAL reading of selected CPOs in a user configurable way. VisIt provides a session configuration that can facilitate a visualization with applied data operators, window positions, ranges, labeling, etc, to be stored/restored for reuse. Communication between VKA and VisIt is made with jVisIt library that VisIt provides when Java (Kepler native) language is required. Figure 2 shows our “universal” VKA in a visualization-only workflow. Input database is single (VisIt native) SILO


Visit
File
Selected
localho
eufor
user
ReOp
Active win
Active plots
Plots Operators PlotAtts OpAtts Variables
Apply operators/selection to all plots
file:/afs/efda-itm.eu/euforia/user/ko. . .e/itm/VisIt/visit-
File Edit View Workflow Tools Window Help
Workflow
SDF Director
VisItSession
Window 2
X Axis
Y Axis

Database: WorkingDir+"solps.silo"

Session Filename: WorkingDir+"solps.session"

WorkingDir: $HOME/KeplerData/workflows/module/itm/VisIt/

SILO database was created with python script from SOLPS.
Session file was created with VisIt opening tha same database, configured and saved as a session file.

Leon Kos, August 2011, UOL, Euforia

execution finished.

41:Pseudocolor - ti_diff

41:Mesh - vessel

Figure 2: Simple Kepler workflow with VisItSession visualization actor (VKA) on top of the VisIt windows (GUI, divertor detail and vessel mesh) showing simulation results from SOLPS.

file format. VisIt use in workflow is therefore not bonded to specific (UAL) type of input. Inter-process communication (IPC) between VKA and VisIt with it's own GUI provides a possibility to user to further tailors visualization aspects and session configuration. For "regular" codes this is normally not the case as they needs to be additionally "wrapped" for inclusion in Kepler workflows. The reason for this is that codes are usually not written in Java as required by Kepler. To facilitate that ITM-TF provides actor generator tools (fc2k, hpc2k, ws2k, ...) which make this process straightforward for Fortran and C++ codes.

2 TIME DEPENDENT VISUALIZATION

Unless there is some "steady-state" phenomena simulated, time is a key physical quantity of a tokamak modeling. ITM database structure reflects experiments identified by *shot* and *run* on each *machine* by CPOs that are time-varying during a plasma pulse. Present simulation codes used in tokamak modeling are all based on time evolution that resembles experiment as close as possible. So, there is already established close correlation between simulation and experimental data. ITM database is thus just a "unification" of both established data models with carefully crafted hierarchy named CPOs as shown in Table 1. During experiment nearly all CPOs are time dependent (i.e. varying during a plasma discharge). Exceptions to that are usually geometrical and positional quantities that are not repeated when time evolves. Time dependence (TD) of CPOs can be carried at different (monotonic) sampling times. Reasoning for different time-bases is due to different data purpose of each CPO. Some effects can be very fast, and other relatively slow-varying or even constant for some physics simulated. Each time dependent CPO in Table 1 is therefore represented as an array of CPOs with assigned sample times. TD CPO thus automatically means array of look-alike CPOs. Internal CPO structure is hierarchical (or object oriented) and can be quite complex with several levels in depth. For detailed CPO description see Ref. [3]. Physical quantities in TD CPOs are structured as multi-dimensional arrays (up to 6D in the present implementation). Single TD CPO can be extracted

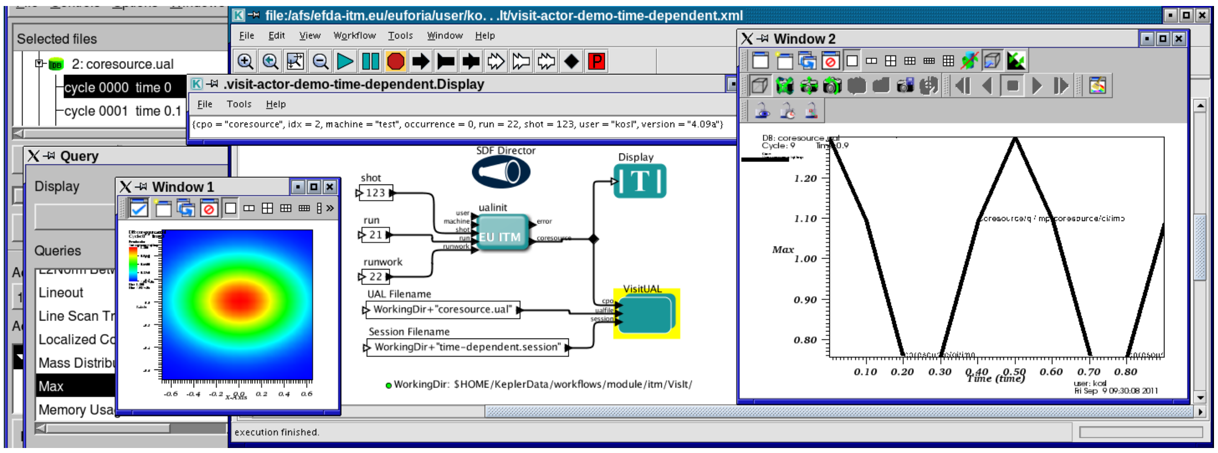
Table 1: Time dependence (TD) for various CPO structures

CPO name	Description	TD
Topinfo	General information about the database entry	No
Summary	Set of reduced data summarizing the main simulation parameters for the data base catalog	No
Antennas	RF antenna list	Yes
Controllers	Description and parameterization of a feedback controller	Yes
Coredelta	Generic instant change of the radial core profiles due to pellet, MHD, etc.	Yes
coreneutrals	Core plasma neutrals description	Yes
coreimpur	Impurity species (i.e. ion species with multiple charge states), radial core profiles	Yes
coreprof	Core plasma 1D profiles as a function of the toroidal flux coordinate, obtained by solving the core transport equations (can be also fitted profiles from experimental data)	Yes
coresource	Generic source term for the core transport equations (radial profile)	Yes
coretransp	Generic transport coefficients for the core transport equations (radial profile)	Yes
equilibrium	Description of a 2D, axi-symmetric, tokamak equilibrium; result of an equilibrium code	Yes
interfdiag	Interferometry; measures line-integrated electron density [m^{-2}]	Yes
ironmodel	model of the iron circuit	Yes
launchs	RF wave launch conditions	Yes
limiter	Description of the immobile limiting surface for defining the Last Closed Flux Surface	No
MHD	MHD linear stability	Yes
magdiag	Magnetic diagnostics	Yes
msediag	MSE diagnostic	Yes
neoclassic	Neoclassical quantities (including transport coefficients)	Yes
orbit	Orbits for a set of particles	Yes
pfsystems	Description of the active poloidal coils, passive conductors, currents flowing in those and mutual electromagnetic effects of the device	Yes
polardiag	Polarimetry diagnostic; measures polarization angle [rad]	Yes
sawteeth	Description of sawtooth events	Yes
scenario	Scenario characteristics, to be used as input or output of a whole discharge simulator	Yes
toroidfield	Toroidal field	Yes
vessel	Mechanical structure of the vacuum vessel	No
waves	RF waves propagation	Yes

(and interpolated) at specified time for codes that operate at selected slice (or cycle in VisIt terminology).

VisIt as a general visualization tool is capable of complex visualizations that are primarily 3D. Experimental data that is imported into UAL by ITM-TF tool *exp2itm* mostly comes in 1D TD form and can be inspected by surplus of graphing tools. ITM-TF for UAL signal inspection provides *Integrated Simulation Editor* (ISE) that will combine analyses and workflows in a “study”-like manner. VisIt role in workflows is dedicated to inspection of 3D or even 4D data resulting from “integrated modeling”. Normally, insight into 3D data is extracted by variety of *operators* at selected cycle (moment in time). Animation and *time-slider* can be useful TD tools for general overview of inspected phenomena. VisIt normally uses *FlipBook animation* for showing time-varying databases. Controls for moving though cycles/time are similar to many software players (or VCR). Scientific visualizations normally do not require time-cycle relation between time and animation but rather investigate interesting phenomena. For that purpose additional key-framing can be applied to animation for the sole purpose of directing animation to final movie creation. During key-framing VisIt operators with view and plot attributes can be applied at specified frames. When multiple time-varying databases are used one can use different (padded, stretched, time, cycle) correlations to align temporal states among them. Similar temporal alignments (closest, previous) can be obtained from UAL database with getSlice() method.

For quantitative analysis of such data VisIt provides *queries* that can also be time dependent. Resulting *Time curve* as shown in Fig. 3 workflow is easily evaluated as time dependent *max* value. Other queries like *pick* or some functional applied on data can be scripted using built-in set on functions. Result of all such queries is always single TD curve. Multiple (different) queries can be stacked in one graph for condensed analysis. Time range and *stride* sampling is used for limiting interest and computational effort with complex data. For 2D and 3D TD



```
{cpo = "coresource", idx = 2, machine = "test", occurrence = 0, run = 22, shot = 123, user = "kosl", version = "4.09a"}
```

Figure 3: Minimal Kepler workflow with VisitUAL actor reading *coresource/qi/imp* CPO field from UAL database as specified by *ualinit* actor. Time dependent maximum value of the *coresource/qi/imp* CPO field is displayed in Window 2. 2D representation at (plasma breakdown) time=0.0 (cycle 0000) is shown in Window 1. Query window shows some possible queries-over-time built-in. Window Display prints structure passed to VisitUAL input describing database and CPO.

queries VisIt provides *lineout* operator that samples values along a line in space. Result is graphed as a set of curves at selected time steps. The principal problem with *lineout* operator is that no operator can be applied after it. One might conclude that representing time with VisIt is limited to x/y graphing. In fact, many 2-4D TD data is difficult to represent on 2D surface. That's why animation is useful and default kind of inspection for TD data. VisIt advantage is a "large" set of operators that "reduce" 3D data to the area of interest. Operators are considered as filters and as such are stacked for complex visualizations. As there are many possibilities within VisIt for creation of specialized visualizations, one can also apply operators on TD database slices disconnected from *Time Slider* and combine them in a single window. The result is timed 3D representation of data on 2D surface. Fig. 4 is an example of such operations of time-varying 2D scalar value . Window 1 in Fig. 3 is a single slice. Elevated (and clipped) slices are translated along desired "time" axis in Windows 1 and 2 of Fig. 4. Slices are samples and non-continuous time representation is normally a must, unless some interpolation is desired. To achieve this, GUI operations with operators are not sufficient. To compensate special desires, VisIt provides Python scripting that can be part of built-in functionality. For example, *lineout along path* and slicing along time produces 2D mesh data that can be further inspected with surface operators. For 3D TD data reduction of dimension is required with pick operators prior to extension with time. In fact, for 3D TD representation, scripting is required, as there is no *standard* operators readily available. Slicing 4D data to get 3D is one possibility. But not as operator as there is no treatment in VisIt for 4D and higher dimensions. VisIt provides custom *plot plugins* just like database plugins. With 3D TD (i.e. 4D) slice plot one could configure reduction surface of interest and add time dimension instead to get 3D volume data for further processing. Such developments are still under consideration by ITM-TF when visualizations of such type will be required. For now, built-in python scripting suffices for our TD visualizations.

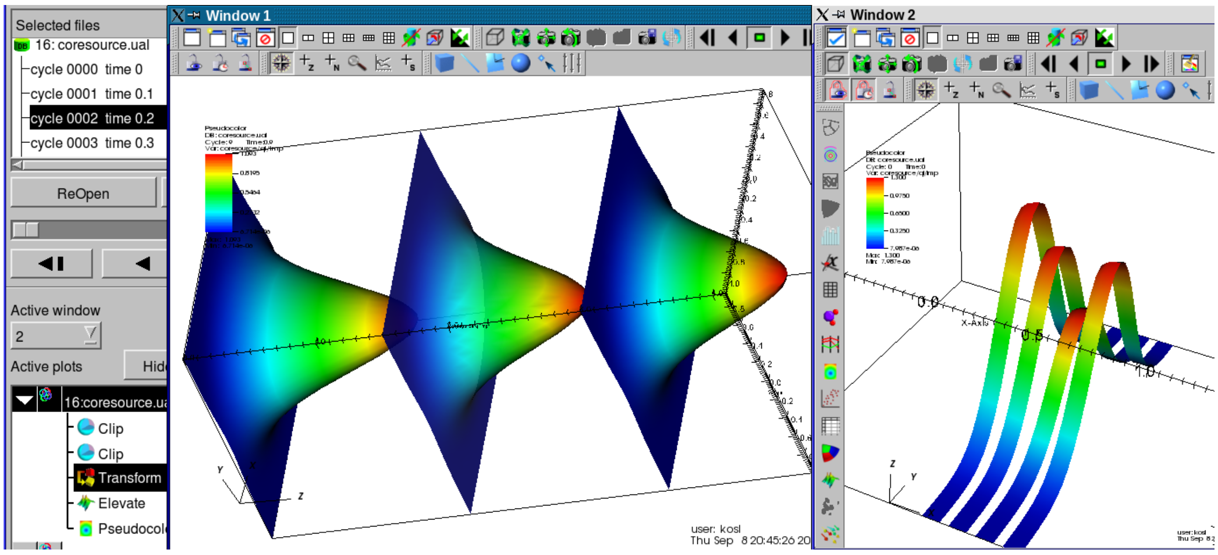


Figure 4: Time dependent visualization of sample *coresource/qi/imp* CPO field with applied stack of VisIt operators. Elevated and translated samples at 3 selected times are shown in Window 1. Slightly different representation in Window 2 with additional central clipping and translation along x-axis is more traditional 3D presentation of time evolution.

3 VISUALIZATION COMPONENTS

Here we briefly describe two major components that were upgraded for TD visualizations under Kepler workflows. VisIt actor is placed in Kepler workflows, while UAL-plugin provides UAL read connectivity for VisIt.

The VisIt Kepler Actor (VKA) provides VisIt visualization software capabilities to the Kepler platform. Although VKA can be unified to support both kinds of databases we decided to have two actors for sake of generality of the first. The second one is then just an extension of the general VKA that requires VisIt plugin for UAL to operate. For this reason VisitUAL actor was “renamed” to VisitSession to emphasize it’s generality while new VisitUAL composite actor was created for CPO/UAL visualizations.

VisitSession actor shown in Fig. 2 can restore session created previously within VisIt but with new source database. Session saved within VisIt GUI consists of all windows, applied plots, operators and attributes. This means that user can configure session under VisIt GUI inside or outside Kepler. Then user builds a particular visualization for the quantities of interest. Finally, user stores a VisIt session file which contains all the information needed to restore the state of VisIt when the session file is generated. In the second stage, the user launches Kepler, inserts the VKA to his workflow and add as parameter of the VKA the session file generated at stage 1. When the workflow is executed, once the VKA is activated by the workflow, one or several visualization windows will pop up with the data contained in the incoming database. The resulting picture is rendered with the visualization parameters contained in the session file. Of course an error is raised if the data used to make the session file are not in the incoming database. This means that session files are special tailored descriptions of visualizations with expected inputs. Normally most programs that require visualization also generate scripts for visualization software like IDL, gnuplot, ... VisitSession is a similar approach but with GUI that saves XML session. Such session can then be shared and collected for visualizations of interest and used as standalone or within Kepler workflow. Such approach is useful for fusion codes and experiments that output complete results in UAL database but researchers are interested in specific phenomena and selected variables and ranges that needs to be monitored within larger scientific workflow supported by Kepler. VisitSession actor was extended to search for its input files from standard data locations ($HOME/.visit for sessions and $VISIT_DATA for databases) when no path is given as an input filename.

For UAL database and CPO support composite actor VisitUAL was created as a convenience actor that includes standard Kepler actors together with VisitSession actor. Coupling VisIt to UAL is supported by ual_reader database plugin. VisIt by design needs to open database provided as filename. UAL database plugin thus needs .ual file provided before it can open UAL within VisIt. Creation of this file can be manual with one line specifying CPO, shot and run. After UAL database is opened with VisIt than one can create a session file for restore within Kepler workflow. Fig. 3 shows such a workflow with user supplied visualization settings. Location of ualfile is given for automatic creation at specified location. This location can be the same as previously manually generated session file. It should be noted that this *ualfile* is then generated from workflow using VisitUAL actor shown in Fig. 3. Composite actor is therefore responsible just for passing appropriate arguments through file when opening UAL. VisitSession actor then takes this as an input and restores session as usual.

UAL plug-in for VisIt (ual_reader) enables VisIt fusion data reads through the UAL in order to build the resulting visualization. VisIt is built on top of *Analysis and Visualization Toolkit* (AVT) that has strong correlation to VTK Toolkit made by Kitware. Data structures from UAL needs to be represented in AVT compatible way. When CPO is read into memory all data is available and information on how this data can be "naturally" represented in coordinate system is being amended by physicists. "Natural" *representation* of data of interest is related to conventions and physicist's way of thinking. For example, one needs to decide which field represent x and y-axis for *1D Curve*. Adding such *representation* tags in XML CPO definitions is ongoing task of ISIP-TF. With this representation information added in the CPO XML description VisIt should provide a very simple way to make nice plots in a couple of clicks. Source code for building *ual_reader* is supported by XSLT code generation that simplifies frequent CPO definition upgrades. Latest XSL Translator development enabled time dependent representation as shown in Figs. 3 and 4. VisIt move from Makefile's to *cmake* simplified code maintenance and now several versions of *ual_reader*s can coexist in plugin directory.

4 CONCLUSION AND FUTURE WORK

Latest enhancements of ITM-ISIP VisIt-related tools targeted at UAL and Kepler workflows enabled time-dependent representation that was previously missing and was a major requirement for widespread use under ITM-TF. As VisIt is under active development at LLNL and now the only *powerful* visualization software that runs under Kepler, we believe that it will play a major role supporting integrated ITER modeling. Present article is thus aimed into promotion of recent advances. User feedback, collaboration and implementation of ideas is necessary to overcome recognized deficiencies. One such idea comes from *ualconnector* ITM-TF software developed recently by H.-J. Klingshirn that "exploits" special VisIt library *libsim*. Simulation library *libsim* allows VisIt to connect to running simulation and acquires desired plot data on the fly. Such "disconnected" use of VisIt could be in principle more versatile for Kepler workflows. For now *libsim* provides Python, Fortran and C interface, while Java is used in Kepler. Dynamic representation generation by using Python grid service library to analyze the CPOs can be more flexible than having to update the representation information in the CPO description for simple plots based on the *General Grid Description* [8] used in the CPOs. Presently, our tools and *ualconnector* are complementary, while ideally, both tools would like to have counterpart features.

ACKNOWLEDGMENTS

This work was supported by the European Commission under: (i) EUFORIA agreement number 211804 within the 7th Framework Program; (ii) Contracts of Associations MHEST, CEA between The European Atomic Energy Community (EURATOM), and (iii) Ministry of Higher Education, Science and Technology of the Republic of Slovenia No. FU06-CT-2007-00065. It was carried out within the framework of the European Fusion Development Agreement, under the EFDA Task Force on Integrated Tokamak Modeling. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1] "MDSplus – data acquisition and storage tools," http://www.mdsplus.org (2011).

[2] "HDF – hierarchical data format," http://www.hdfgroup.org/ (2011).

[3] Frederic Imbeaux, J. B. Lister, G. T. A. Huysmans, W. Zwingmann, M. Airaj, L. Appel, V. Basiuk, David Coster, Lars-Goran Eriksson, Bernard Guillerminet, Denis Kalupin, C. Konz, Gabriele Manduchi, M. Ottaviani, G. Pereverzev, Y. Peysson, O. Sauter, J. Signoret, and Par Strand, "A generic data structure for integrated modelling of tokamak physics and subsystems," Computer Physics Communications **181**, 987–998 (2010).

[4] Matthieu Haefele, Leon Kos, Pierre Navaro, and Eric Sonnendrücker, "Euforia integrated visualization," in *PDP 2010 - The 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing* (Pisa, Italy, 2010) pp. 498–502.

[5] "VisIt official homepage," http://visit.llnl.gov/ (2011).

[6] G. Manduchi, F. Iannone, F. Imbeaux, G. Huysmans, J.B. Lister, B. Guillerminet, P. Strand, L.-G. Eriksson, and M. Romanelli, "A universal access layer for the integrated Tokamak Modelling Task Force," *6th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research*, Fusion Engineering and Design **83**, 462–466 (2008).

[7] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock, "Kepler: An extensible system for design and execution of scientific workflows," in *SSDBM '04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management* (IEEE Computer Society, Washington, DC, USA, 2004) p. 423, ISBN 0-7695-2146-0.

[8] Hans-Joachim Klingshirn, *Adaptive grids and numerical fluid simulations for scrape-off layer plasmas*, Ph.D. thesis, Technical University Muenchen (2010), http://mediatum.ub.tum.de/node?id=970446.