Multi-modality medical image simulation of biological models with the Virtual Imaging Platform (VIP)

Adrien Marion[1], Germain Forestier[2], Hugues Benoit-Cattin[1], Sorina Camarasu-Pop[1], Patrick Clarysse[1], Rafael Ferreira da Silva[1], Bernard Gibaud[2], Tristan Glatard[1], Patrick Hugonnard[3], Carole Lartizien[1], Hervé Liebgott[1], Svenja Specovius[1], Joachim Tabary[3], Sébastien Valette[1], Denis Friboulet[1]

[1] Université de Lyon, CREATIS; CNRS UMR5220; Inserm U1044; INSA-Lyon; Université Lyon 1, France

[2] INSERM / INRIA / CNRS / Univ. Rennes 1, VISAGES U746, Rennes, France

[3] CEA-LETI-MINATEC, Recherche technologique, 17 Rue des martyrs, 38054 Grenoble Cedex 09, France

Abstract

This paper describes a framework for the integration of medical image simulators in the Virtual Imaging Platform (VIP). Simulation is widely involved in medical imaging but its availability is hampered by the heterogeneity of software interfaces and the required amount of computing power. To address this, VIP defines a simulation workflow template which transforms object models from the Intermedi-Ate Model Format (IAMF) into native simulator formats and parallelizes the simulation computation. Format conversions, geometrical scene definition and physical parameter generation are covered. The core simulator executables are directly embedded in the simulation workflow, enabling data parallelism exploitation without modifying the simulator. The template is instantiated on simulators of the four main medical imaging modalities, namely Positron Emission Tomography, Ultrasound imaging, Magnetic Resonance Imaging and Computed Tomography. Simulation examples and performance results on the European Grid Infrastructure are shown.

1. Introduction

Image simulation is widely involved in medical imaging procedures but simulators are hardly interoperable and often have steep learning curves, making the design of multi-modality simulations a tedious task. These difficulties are worsened by the amount of computing time required by realistic simulations and the volume of data generated.

The Virtual Imaging Platform (VIP) is a web platform for multi-modality medical image simulation. It targets (i) interoperability issues among simulators, (ii) the sharing of object models and (iii) the handling of heavy simulations by the use of Distributed Computing Infrastructures (DCIs). VIP includes example simulators for four of the main medical imaging modalities, namely FIELD-II for Ultrasound imaging (US) [4], PET-Sorteo for Positron Emission Tomography (PET) [7], SIMRI for Magnetic Resonance Imaging (MRI) [1] and Sindbad for Computed Tomography (CT) [9]. This paper presents the framework used to integrate simulators and object models in VIP.

Object models and image simulators are the two main components of a medical image simulation. Simulators are usually considered as legacy codes which cannot be modified and have to be integrated in the platform as off-the-shelf components. In these conditions, their porting on DCIs uses data parallelism instead of code parallelisation: the simulation is split into independent tasks over which existing simulation codes are concurrently executed.

Object models represent biological objects or phantoms. They may contain information about anatomy, pathology and physiology and can be dynamic. They also must carry modality-specific information about physical parameters involved in the simulation. These may have various representations, *e.g.* spatial parameter maps or look-up tables. Besides, a simulation scene defines the spatial relations between an object model and simulators. For that different geometrical conventions may be used by the simulators, further complicating the definition of multi-modality simulations.

VIP integrates simulators and object models by constructing workflows orchestrating object model manipulation codes and simulation components. Workflows have been used for several years to describe scientific applications [5]. They provide a structured representation of processes, which is useful to exploit parallelism, to track data dependencies and to automatically adjust pipelines to new object models or simulators, which VIP plans to offer.

This paper details our workflow framework to address compatibility issues between object models and simulators as well as the porting of simulators on DCIs. It complements our companion paper defining a semantic approach for model sharing in the platform [2]. The following Section describes the workflow template as a composition of four workflow components. Section 3 defines the simulation scene, describing the object model format and explaining the geometrical conventions adopted. Object preparation and core simulation WFCs are instantiated on the platform simulators in Sections 4 and 5. WFCs are implemented in the Gwendia language [6]. They are publicly available from myExperiment[1] and can be launched from VIP. Finally, simulation results are reported in Section 6.

2. Simulation workflow template

As represented in Fig. 1, a simulation workflow (SWF) is a composition of workflow components (WFCs) that can be simulation parameter generation, object preparation, core simulation and post-processing. In the figure, dotted-line rectangles represent WFCs and in/output ports are figured by yellow/orange rectangles. Lines represent data links and trapezoids are inputs and outputs. The template takes as input an object model in the IAMF format (see description in Section 3), a scene definition and simulation parameters.

Simulation parameter generation WFCs assemble simulator parameter files from numerical, textual or file input. They simplify the simulator interface by hiding the parameter file formats. Presets exposing only the parameters relevant to a particular group of users could also be defined.

Object preparation WFCs adapt IAMF models to the native simulator formats. They take as input an IAMF object and scene transformation matrices as defined in Section 3. First, they split the object into independent static models to enable data parallelism. Then, they perform format conversions, scene transformation, and physical parameter generation to adjust the object model to the target simulator. Simulator parameters (*e.g.* geometry settings) could also result from object preparation.

Starting from the prepared objects, core simulation WFCs compute the simulation on DCIs. To further exploit data parallelism, temporal splitting is complemented by spatial data splitting into 3D, 2D or even 1D simulation chunks or into simulations with reduced intensity (*e.g.* of the X-Ray source in CT or of the radiotracer in PET). The simulation code is then concurrently iterated on the simulation chunks. This way, no modification of the simulator code is required. Once computed, simulation chunks are merged to produce a (static) simulated data set. Finally,

[1] http://www.myexperiment.org/workflows/[2068-2071]

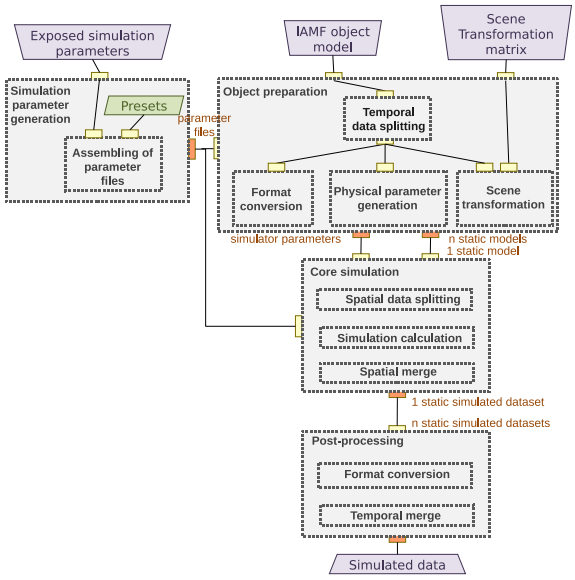


Figure 1. Simulation workflow template

post-processing performs format conversion and temporal merging of the dynamic simulated data set.

The next Sections describe object preparation and core simulation. Parameter generation and post-processing will be addressed in the next months of the project.

3. Definition of the simulation scene

A simulation scene includes a single object model in a pivot format called IAMF and defined by the project. Using a pivot format improves the compatibility between object models and simulators, reducing from $m \times n$ to $m + n$ the complexity of adapting m object models to n simulators. It also facilitates model sharing, browsing and visualization.

IAMF consists of a set of data files annotated using terms and concepts of the VIP ontology (see detailed description in [2]). These annotations describe the content of the model and can be used to define rules specifying validity constraints on the models for a given modality or simulator.

An IAMF model is represented using two temporal scales distinguishing longitudinal follow-up (*e.g.* for multiple sclerosis) from a dynamic acquisition (*e.g.* echocardiographic exam). Thus a model consists of one or several time points defined as a date and a time. Each time point is made of one or several instants defined by a temporal offset in the time point. Then each instant of the model consists of object parts belonging to one of anatomical, pathological, geometrical, foreign-body or external agent layers. Each of these object parts can be described by voxel maps and/or by meshes. In addition, physical parameters are described as voxel maps or look-up tables linking object

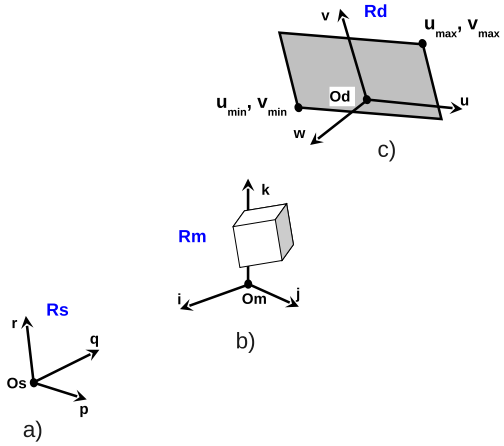


Figure 2. Scene coordinate system with a) Source `Rs` frame, b) Object `Rm` frame and c) Detector `Rd` frame

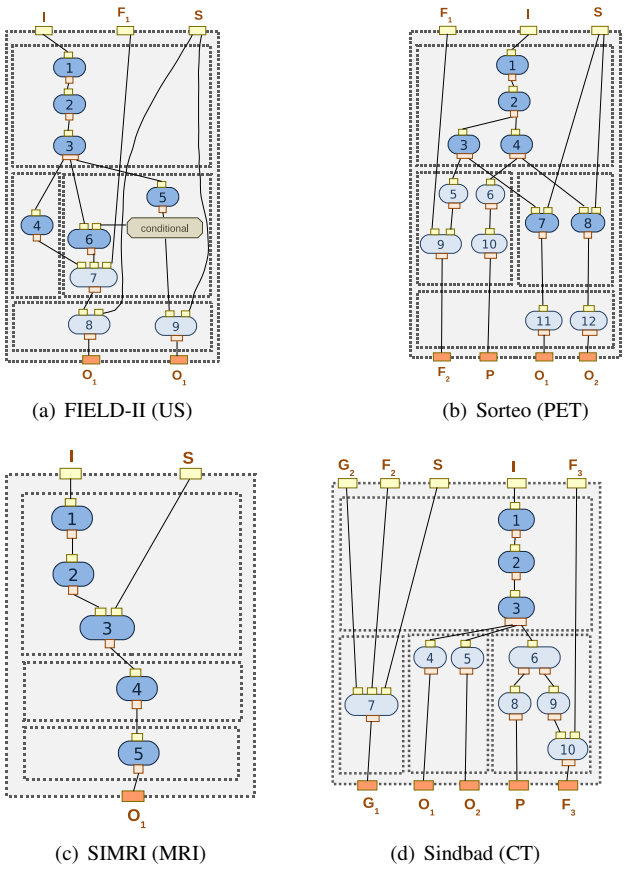
parts to physical properties. Due to their wide use, VTK formats[2] were chosen to describe the data files. Meshes are defined as vtkPolyData `.vtp`, images as vtkImageData `.mhd/.zraw`, scatterers (physical parameters used for US simulation) as vtkUnstructuredGrid `.vtu` and look-up tables as plain text or XML files. An IAMF model is embedded in a `.zip` archive containing the different data files and an RDF file where annotations are stored.

The object model of a multi-modality simulation is interpreted by one or more simulators of PET, US, MRI or CT images. Thus, it has to contain physical parameters for several modalities. To facilitate the definition of physical parameters for each object part a physical parameters database is available in VIP. In this database, tissues are characterized by physical properties that can be echogenicity (spatial and amplitudes distributions of scatterers) for US, magnetic properties (T1, T2, T2*, susceptibility χ) and proton density ρ for MRI or chemical composition for CT and PET.

The simulation scene is defined in a normalized coordinate system consisting of 3 frames and represented on Fig. 2. The object model is defined in an arbitrary `Rm` frame (`Om`,`i`,`j`,`k`). The position of the simulator detector is defined by frame `Rd` (`Od`,`u`,`v`,`w`) where `Od`, `u`,`v` and `w` are given in `Rm`. The dimensions of a planar detector are given by (u_{max}, v_{max}) and (u_{min}, v_{min}) in `Rd`. Device coordinates in pixels (*i.e.* number and spacing between sensors) are specified where appropriate. The simulator source is defined by frame `Rs` (`Os`,`p`,`q`,`r`) where `Os`, `p`,`q` and `r` are given in `Rm`. `Rd` (resp. `Rs`) is omitted in case the simulator only consists of a source (resp. detector).

In general, the scene definition produces two 4*4 rigid

[2] `http://www.vtk.org`



(a) FIELD-II (US) (b) Sorteo (PET)

(c) SIMRI (MRI) (d) Sindbad (CT)

Figure 3. Object preparation WFCs

transformation matrices (3 rotations and a translation) linking `Rm` to `Rd` and `Rs` to `Rd`. A graphical user interface is in preparation to facilitate scene description.

4. Object preparation

Object preparation is simulator-specific but it always consists of format conversion, scene definition and physical parameters definition. It takes as input the scene definition matrices described in the previous Section, optional sequence parameters files and an IAMF model valid for the simulator. Outputs are data files converted into native simulator formats and optionally parameters files or physical parameter definitions.

Object preparation workflows are shown on Fig. 3. Light-blue workflow activities represent legacy simulator codes while dark blues denote Java code directly embedded in the workflow description. The following letter code is used for sources and sinks: `I` is the IAMF model assumed valid for the simulator; `S` are the scene transformation matrices; $\texttt{O}_i$ are object files in the native simulator format; $\texttt{F}_i$ are simulation parameter files; `P` are physical parameter def-

initions; G_i are geometry definition files.

For FIELD-II, the IAMF valid model is split into time points and instants by activities 1 and 2 in the WFC of Fig. 3(a). Dynamic simulations are split into independent static simulations to simplify IAMF definition (no elaborated time transformation representations such as motion fields is required) and to enable data parallelism on the simulation instants.

Since existing simulators cannot process multi-layered object models, IAMF *flattening* is performed by activity 3. This is implemented by merging the model layers together and the parameter layers together. For voxel maps, it replaces voxels of layers *n* by those of layers *n+1* where the latter are not null. For meshes, it simply adds new meshes. The flattening operation is performed by Algorithm 1. First, `object` layers are sorted and iterated in decreasing priority values for the flattening. This ordering is based on a rough layer priority guess assuming that geometry and anatomy are always superseded by pathology that is in turn overlaid by external bodies and foreign agents. The flattened object (`finalObject`) and the simulation parameters (`simulatorParameterSet`) are then initialized and the layers are iterated in the order described above. At each iteration the `finalObject` voxel map, LUT, meshes and mesh LUT are merged with the content of the `object` layer. Then, the physical parameters maps and LUTs of `finalObject` are merged with the content of the `object` layer parameters.

After this step, the FIELD-II workflow tests if scatterers are embedded in the model in activity 5. If yes, the transformation matrix is applied to the scatterers by activity 9. Otherwise, scatterers are generated as a function of the label of each tissue in activity 7 which also needs the model converted into native format by activity 4 and the physical parameter LUT retrieved by activity 6. A set of scatterers is generated for each voxel from the parameters of the statistical distribution associated to the voxel label. Finally, the scene transformation matrix is applied to the scatterers by activity 8. Note that the total number of scatterers can be in the order of 10^8, which may lead to memory issues. Besides, the position of scatterers used for US simulation has to be controlled during a dynamic sequence to preserve coherency between two images. Thus a dynamic IAMF model valid for US has to define the scatterers at each instant.

For Sorteo, the IAMF valid model is split into time points and instants by activities 1 and 2 by the WFC of Fig. 3(b). The flattening is repeated twice, once for the emission object (activity 3) and once for the attenuation object (activity 4). The emission radioactivity is read in workflow activity 5 and inserted into the protocol by activity 9. Activity 6 reads labels from the attenuation object that are used by activity 10 to compute cross sections and probabilities of attenuation phenomena. Then the rotation parameters are used to transform the voxel map and the translation parameters are put in the parameter files by activities 7 and 8. Finally, format conversion is done by activities 11 and 12.

For SIMRI, the IAMF model is split into time points and instants by activities 1 and 2 in the WFCs of Fig. 3(c). The flattening step is then performed by activity 3 and activity 4 writes the physical parameter distributions in the format of the simulator. Finally, activity 5 applies the scene transformation to the voxel object representation. No format conversion is needed since SIMRI directly reads VTK files.

For Sindbad, the IAMF valid model is split into time points and instants by activities 1 and 2 in the WFCs of Fig. 3(d). The flattening step is then performed by activity 3 and the format conversions for voxel and mesh representations are respectively done by activities 4 and 5. The generation of physical parameters is performed by calculating the cross sections and probabilities of attenuation phenomena from chemical compositions retrieved in activity 6. It is done in activities 8, 9 and 10. Activity 7 edits the geometry parameter file.

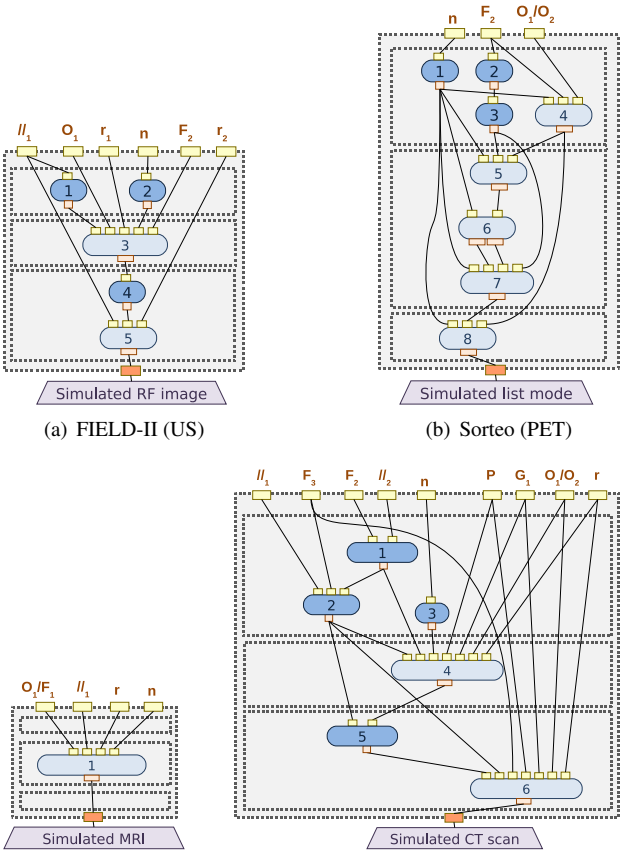
5. Core simulation

Core simulation WFCs for the 4 exemplar simulators are shown on Fig. 4. The same letter code is used, adding the following notations: n is the simulation name, used to generate unique file locations, r_i are releases of the simulator code and $//_i$ are parameters specific to the parallelization. In FIELD-II, spatial data splitting (activity 1 on Fig. 4(a)) chops the 2D or 3D input data into a set of independent 1D radio-frequency (RF) lines. Activity 2 is only used to define a unique output directory for the simulation. Then activity 3 simulates RF lines by calling the FIELD-II Matlab API[3] using a release of the US probe code and a probe-specific Matlab parameter file. Finally the simulated RF lines are collected and merged in a Matlab image by activities 4 and 5. The merging procedure also requires a probe-specific release of the merge code. A simulated RF image is finally produced. Depending on the application it is converted into a B-mode image in the post-processing workflow.

In Sorteo, the simulation can be split into any number of jobs in which only a fraction of the total number of emitted positrons is simulated. The job number is parsed in the simulation parameter file by activity 2 and the jobs are generated by activity 3. Activity 4 compiles the simulator textual parameter file into a binary representation and activity 1 defines the simulation output directory. Activities 5, 6 and 7 call Sorteo binaries that compute the simulation. Finally activity 8 merges all the results in a list mode file.

The SIMRI core simulation WFC is a simple wrapping of the simulator executable since it was already parallelized

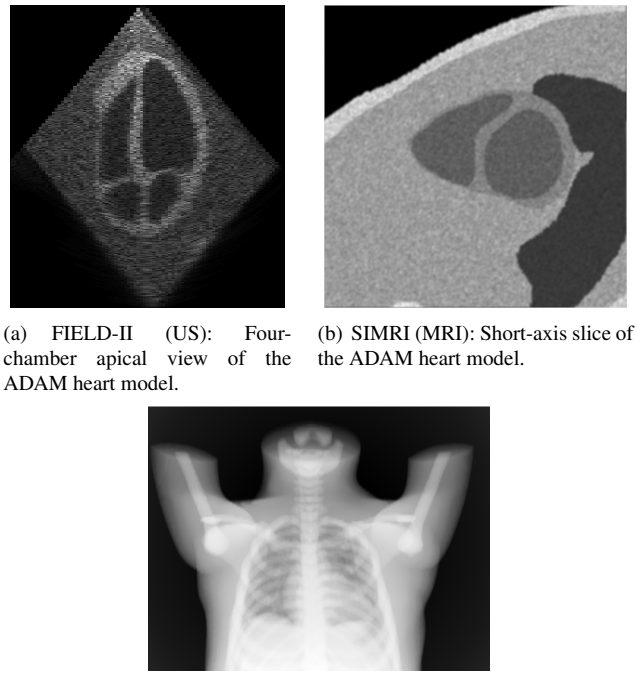
[3] http://server.electro.dtu.dk/personal/jaj/field/



(a) FIELD-II (US) (b) Sorteo (PET)

(c) SIMRI (MRI) (d) Sindbad (CT)

Figure 4. Core simulation WFCs. Light-blue activities wrap simulation codes.



(a) FIELD-II (US): Four-chamber apical view of the ADAM heart model.

(b) SIMRI (MRI): Short-axis slice of the ADAM heart model.

(c) Sindbad (CT): 1 projection of a 3D scan of the XCAT model

Figure 5. Benchmark simulations.

	FIELD-II	SIMRI	Sindbad
CPU time (s)	1,523,387	2,432	6,278,220
Elapsed (s)	52,690	575	85,800
Speed-up	29	4	73

Table 1. Performance of the benchmark simulations on EGI.

using the Message Passing Interface (MPI). It performs data splitting, simulation calculation and data merging.

In Sindbad, two levels of data splitting are exploited. First, the simulation is split into independent 2D simulations of the CT projections. Each 2D simulation has an analytical and a Monte-Carlo components that respectively simulate the direct and scattered radiations. Secondly, the Monte-Carlo part is split into chunks simulating only a fraction of the number of photons specified by the user. This two-level splitting is done by activities `1`, `2` and `3` on Fig. 4(d). It is controlled by scanner parameters (in particular the number of 2D projections), the allowed maximum number of jobs, the maximal number of photons per job and simulation parameters describing the analytical and Monte-Carlo parts. The Sindbad executable is then iterated concurrently on all the 2D projection chunks. Finally, activity `5` merges the Monte-Carlo chunks and activity `6` merges the analytical and Monte-Carlo parts.

6. Results

Benchmark simulations were executed on the *biomed* virtual organization of the European Grid Infrastructure[4] to validate the core simulation workflows. Three simulated results are shown on Fig. 5. For US, a four-chamber apical view was simulated from the ADAM cardiac model [3] with a density of 5 scatterers by resolution cell. A sectorial probe with 64 elements was used to produce a 128-line image. For MRI, we used SIMRI to simulate a 256x256 short-axis slice of ADAM. For CT, a 360-projection 3D scan was simulated on the XCAT model [8] using 10^8 photons for the Monte-Carlo simulation. Besides, performance figures are reported on Table 1. Speed-up is computed as the ratio between the consumed CPU time and the elapsed time of the simulation.

[4] `http://www.egi.eu`

7. Conclusion

We presented a workflow framework for the integration of medical image simulators in the Virtual Imaging Platform. A simulation workflow template was described and object preparation and core simulation were instantiated for 4 simulators. Core simulation workflows are public and can be launched from VIP[5] with a registered account. Object preparation, simulation parameter generation and post-processing workflows will be progressively available. Tools for object model sharing and retrieval will also be provided.

Simulator integration was exemplified on 4 simulators but the template is meant to be applicable to other simulators. It consists of generic parts common to each simulator. Based on this workflow template, a specific workflow designer will be developed to help simulator developers integrate their own tools in VIP. In the long term, semi-automatic composition of simulation workflows is targeted.

8. Acknowledgment

This work is funded by the French National Agency for Research under grant ANR-09-COSI-03 "VIP". We are grateful to Simon Marache for helping with the PET simulations and to Carlos Gines Fuster and Gabriel Levy for bootstrapping the FIELD-II and PET-Sorteo core simulation workflow components. Johan Montagnat provided the stylesheet for the workflow descriptions. We thank the European Grid Initiative and "France-Grilles"[6] for providing the computing infrastructure and user support.

References

[1] H. Benoit-Cattin, G. Collewet, B. Belaroussi, H. Saint-Jalmes, and C. Odet. The SIMRI project : a versatile and interactive MRI simulator. *Journal of Magnetic Resonance Imaging*, 173(1):97–115, 2005.

[2] G. Forestier, A. Marion, H. Benoit-Cattin, P. Clarysse, D. Friboulet, T. Glatard, P. Hugonnard, C. Lartizien, H. Liebgott, J. Tabary, and B. Gibaud. Sharing object models for multi-modality medical image simulation: a semantic approach. In *Proceedings of IEEE CBMS*, 2011.

[3] R. Haddad, P. Clarysse, M. Orkisz, D. Revel, and I. Magnin. A realistic anthropomorphic numerical model of the beating heart. *Innov Tech Biol Med - RBM*, 26(4):270–272, 2005.

[4] J. Jensen and N. B. Svendsen. Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 39(2):262–267, 1992.

[5] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *CCPE*, 18(10), 2006.

[6] J. Montagnat, B. Isnard, T. Glatard, K. Maheshwari, and M. Blay-Fornarino. A data-driven workflow language for grids based on array programming principles. In *Workshop on Workflows in Support of Large-Scale Science (WORKS'09)*, pages 1–10, Portland, USA, Nov. 2009.

[7] A. Reilhac, G. Batan, C. Michel, C. Grova, J. Tohka, D. Collins, N. Costes, and A. Evans. PET-SORTEO : Validation and development of database of simulated PET volumes. *IEEE Transactions on Nuclear Science*, 52(5), 2005.

[8] W. P. Segars, B. M. Tsui, D. S. Lalush, E. C. Frey, M. A. King, and D. Manocha. Development and application of the new dynamic nurbs-based cardiac-torso (ncat) phantom. *Journal of Nuclear Medicine*, 42(5), 2001.

[9] J. Tabary, P. Hugonnard, and F. Mathy. SINDBAD: a realistic multi-purpose and scalable X-ray simulation tool for NDT applications. In *Proceedings of International Symposium on Digital industrial Radiology and Computed Tomography*, Lyon, France, 2007.

[5] `http://vip.creatis.insa-lyon.fr`

[6] `http://www.france-grilles.fr`

Algorithm 1. Flattening of a multi-layered IAMF model.

```
// in: object - Multi layered object model
// out: finalObject - Flattened model
sortedLayers ← sort(object.layer)
init object and finalObject
for layer in sortedLayers do
   finalObject.VoxelMap.merge(object.layer.VoxelMap)
   finalObject.VoxelLUT.merge(object.layer.VoxelLUT)
   finalObject.Mesh.merge(object.layer.mesh)
   finalObject.MeshLUT.merge(object.layer.meshLUT)
   for param in simulatorParameterSet do
      finalObject.param.VoxelMap.merge(object.layer.param.VoxelMap)
      finalObject.param.LUT.merge(object.layer.param.LUT)
   end for
end for
```