# NEUTRAL DESCRIPTION AND EXCHANGE OF DESIGN COMPUTATIONAL WORKFLOWS

**Gondhalekar A. C.[1], Guenov M. D.[1], Wenzel H.[2], Nunez M.[1], Balachandran L. K.[1]**

## ABSTRACT

Proposed in this paper is a neutral representation of design computational workflows which allows their exchange and sharing between different project partners and across design stages. This is achieved by the de-coupling of configuration and execution logic. Thus, the same underlying workflow can be executed with different (fidelity) models and different software tools as long as the inputs and outputs of the constituent process are kept the same. To this purpose, an object model is proposed to define different simulation objects, their scope, and hierarchy in the simulation process. An XML based computer readable representation of workflows based on the proposed object model, is also suggested. The application of the proposed representation is demonstrated via a case study involving the exchange of workflows between two design partners. The case study also demonstrates how the same workflow can be executed using different execution tools and involving different fidelity models.

*Keywords: Computational workflows, data/information sharing, model-based design*

## 1. INTRODUCTION

Increasing market demands for more efficient and capable products in recent years have lead to an increasing product complexity. At the same time, the pressures for shorter lead times and better cost effectiveness have resulted in business models involving multiple partners. However, the difficulties recently encountered, for example, by the global aviation industry with regard to delivery have demonstrated that such complexity cannot be solved only by gradual improvement of existing practices. A major European project aiming to tackle these problems is CRESCENDO [13]. One of the major aims of the project is to research and develop enabling methods and tools which will make a reality the design of a complete virtual aircraft, up to certification. The paradigm adopted in CRESCENDO is named the **Behavioural Digital Aircraft (BDA)**. BDA is seen as the capability which will be able to federate and interact with different models thus enabling a seamless behavioural simulation of the aircraft throughout the development lifecycle and across the extended enterprise. In a sense BDA is expected to evolve into a Functional/Behavioral 'Mock-Up' and to become complementary to the Digital Mock-Up, which currently is indispensible with regard to geometry, topology and product lifecycle management (PLM) in general.

In this wider context and ambitious global aims, the objectives of the research presented in this paper have been to develop the first steps towards enabling interoperability and reusability of automated computational design workflows. Taking the extended enterprise as an example, there may be good reasons (such as protection of intellectual property, security, etc.) why the partners may prefer to use their own code/software to operate on shared computational workflows. Such policy is also relevant in cases of a transition from one stage of the design cycle to the next (for example, from conceptual to detailed design), where the physics is the same but the model-details are different. In such cases, although the underlying computational workflow logic is the same, the actual models and the tools/software used to execute these models may be different. It is therefore important that a neutral description of computational workflows is available, which can be shared easily. This in turn, necessitates that the logic of the dynamically configured computational workflows is decoupled from

[1] Cranfield University
[2] Dassault Systemes

the implementation. Using a biological term as metaphor, the neutral description is intended to serve as some sort of a ‘computational DNA' (not to be confused with bio-molecular computing) to facilitate the sharing and exchange of simulation information and associated knowledge.
In this paper, we present an XML based neutral representation of computational workflows, primarily focusing on low fidelity simulations. Currently the idea of employing a standard and neutral description for a process is discussed in the literature mainly in relation to the process industries and to business process modelling. There are standards such as BPMN (Business Process Modelling Notation) and BPML (Business Process Modelling Language), set by the OMG group [2][4], which use graphical notations to represent business process in a neutral way, independent of the implementation environment. Available is also an ISO standard for the representation of process plant lifecycle information [3]. The representation proposed in this paper follows the philosophy of the above mentioned standards, but in the specific context of model-based computational processes. It is more focused on addressing the needs of the design (simulation) community by incorporating simulation specific taxonomy. The intention is that the proposed XML implementation would allow workflow description to be parsed by computer programs in order to facilitate the automation and sharing of computational processes.
The rest of the paper is arranged in four sections. The next section explains in detail the proposed object model and XML syntax used for the development of the neutral description of computational workflows. Section 3 gives a practical example on exchanging computational workflows in a neutral format between two different partners. Finally, conclusions and proposed future work in this area are presented in Section 4.

## 2. APPROACH

In order to achieve a neutral description of a computational workflow, it is necessary in the first instance, to agree upon common definitions of the building blocks (objects) of any simulation process such as data, models, simulation tools and so forth. Such a set of definitions is referred to in the field as ontology, taxonomy or the more practical term - object model. We have chosen to use the latter term in this paper. Once the object model is established, the next task is to decide on the representation of this object model in a computer readable format, XML in this case.
It is important to mention at this stage that the simulation objects can have different meaning depending on the fidelity level of the simulation. For example, at conceptual design stage, when dealing with low fidelity computation, the models are seen as executable programs whereas in high fidelity FEA (Finite Element Analysis) and CFD (Computational Fluid Dynamic) simulations, meshed geometry can be seen as a model. Within the CRESCENDO project, a considerable effort is being made to agree upon a global object model which covers different fidelity levels in simulation processes. The object model presented below is an extension of the object model presented in [1]. Historically it has been tailored to meet the needs of low fidelity simulations within conceptual and preliminary design stage. It is demonstrated later on in the paper how such workflows can be shared with tools enabling higher fidelity simulations.

**Object Model:**

- Data

  Data includes strings, scalars or arrays which constitute the most fundamental objects in the computational workflow class hierarchy. Data comprise the inputs and outputs of a model. Data attributes include: name (string), type (string), nominal value (string/double/integer/double array), unit (string), min (integer/double), max (integer/double).
- Model

  Model is an object in computational workflows that provides a set of output data after performing a series of computational operations upon specific input data objects. From an implementation point of view, such an approach allows to integrate within workflows any model object as a generic black box, regardless of whether it is an empirical formula, a proprietary computer code, or a script to run external applications.
  Attributes: Name (string), Inputs (Data), Outputs (Data), Auxiliary (Boolean)

Methods: Execute

Here, *Auxiliary* attribute determines whether the model will be considered while configuring/scheduling the workflow. For models with *'Auxiliary=true*', the models act as global functions accessible from other models without being part of the computational workflow. For example, in aircraft design, a function to calculate air properties for a given altitude can be implemented as an *Auxiliary model* which can be accessed by several models within the computational workflow, potentially with different input values.

- Sub-process

  Sub-process is an object which stores a collection of suitable models for allowing the computation of a specific simulation, along with their sequence of execution. It is important to note that the execution sequence of a given set of models may be different depending on the inputs and outputs required by the user. This is due to, not least to the fact that, choosing a variable to be independent, i.e. to be an input into the computation, while its default state is actually an output of a model, would require numerical inversion of the latter. This in turn would dictate a possible change to the execution logic [8].
  Attributes: Name (string), Execution Sequence (string array), Inputs (Data), Outputs (Data)
  Methods: Execute

- Strongly connected components (SCC)

  When models are grouped together to form a sub-process with user defined inputs and outputs, some models may be involved in feedback loops due to shared input/output variables. The models can be re-arranged so as to minimize the number and the length of feedback loops. This will reduce the computational time while solving the coupled system. In the final arrangement, the models with feedback loops are grouped together in a structure called strongly connected components (SCCs). These SCCs are then solved iteratively by applying an appropriate numerical method. The SCC class is similar to the Sub-process class with an additional treatment to solve it iteratively.
  Attributes: Name (string), Execution Sequence (string array), Inputs (Data), Outputs (Data), feedback_variables (Data), numerical treatment (Treatment)

  Figure 1 shows an example of a strongly connected component in which Model 5 feeds back to Model 3 via Models 6 and 4. This type of arrangement has to be solved using iterative algorithms.

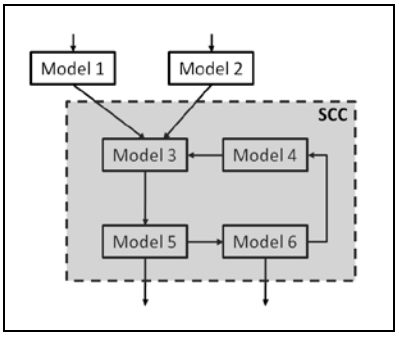


*Figure 1. An example of a strongly connected component (SCC)*

- Modified Model:

  A modified model is essentially a reversed model with one or more of the inputs changed to outputs and vice versa. A reversed model is solved using appropriate numerical methods such as fixed point iteration or Gauss-Newton method.
  Attributes: Name (string), Inputs (Data), Outputs (Data), original model (Model)

For object model implementation, the four classes described above (viz. Model, Sub-process, SCC, and Modified model) are inherited from an abstract class *ModSub* because of their commonality.

- Treatment:

  A Treatment object consists of numerical methods which can be applied at different levels in the hierarchy of a computational workflow. In the current implementation, a treatment can be applied on models, or sub-processes. Examples of a treatment can be a numerical differentiation treatment applied on a model, or an optimization treatment applied on a sub-process. A treatment applied on a model or a sub-process results in a set of output data. A treatment object has an attribute *Template*, which describes all relevant parameters for the treatment. For example, for an optimization treatment, the template may store objective functions, constraints, design variables, and other settings required by the optimizer. For a Genetic Algorithm optimization, these settings may be the number of generations, population size, etc.

  Attributes: Name (string), Template (string array)

- Study:

  A Study object is an assembly of a specific model/sub-process with a particular treatment applied on it to carry out a required design process. A study can be considered to be similar to a sub-process, but at a higher level in the class hierarchy. Moreover, the realization of complex design processes can be achieved by nesting study objects.

  Attributes: Name (string), model (*ModSub*), treatment (Treatment), Output (File), studies (Study)

  Methods: Execute

Figure 2 shows the class diagram using UML notations.

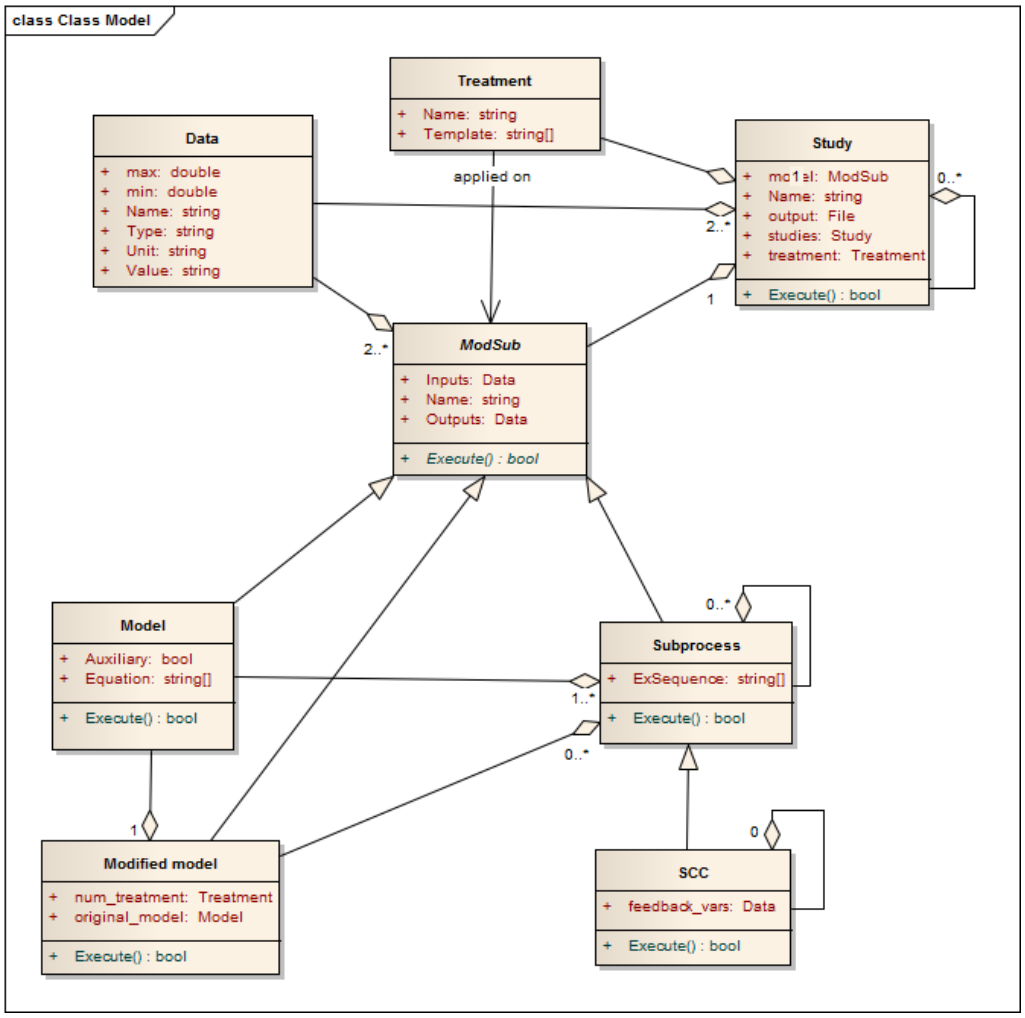


*Figure 2. UML representation of the object model*

Though the SCC class inherits from the Sub-process class, by definition, an SCC cannot contain another SCC inside it. This is represented in the object model diagram using an additional constraint applied on the SCC class.

**XML representation of simulation objects**

The next step, once the objects which are manipulated during the simulation at hand have been defined, is to represent a computational workflow in a computer readable format. In the present work an eXtensible Markup Language (XML) based representation is chosen. XML was originally created to structure, store, and transport information [5], and thus it is inherently better structured compared to other ASCII text formats. In addition, XML can be easily parsed by different computer programming languages.
In the proposed XML format, each object in the computational workflow hierarchy is represented by respective containers. Thus, the XML file will have different containers, namely, data, model, sub-process, SCC, modified model, treatment, and study containers. Each container takes the structure of the respective class, and describes all attributes of that class. Figure 3 shows, as an example, the structure of the data container and the model container in XML format.

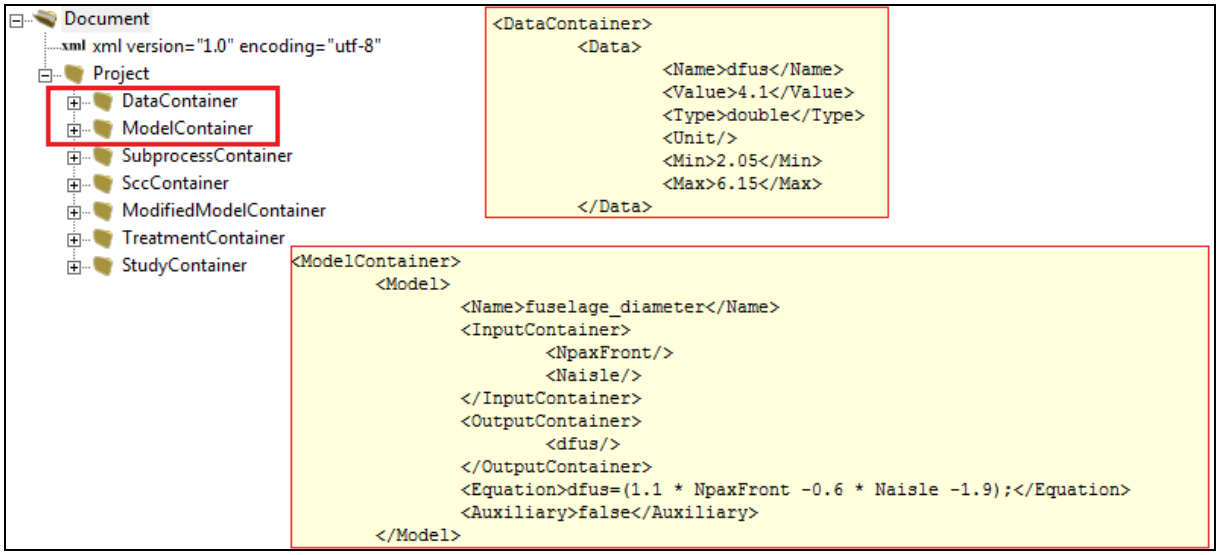


*Figure 3. An example of data and model container in XML format*

The top level container is the project container, which stores all simulation objects, treatments, and studies. In the current XML notation, unlike in BPMN, the connectivity of different simulation objects is implicit and can be derived from the description available in the XML file. In future, another section can be added in the XML file to specify the connections explicitly by using graph modelling notations such as the one proposed in GraphML [6].

## 3. CASE STUDY

The concept of sharing the neutral description of computational workflow, presented in the above section, was demonstrated as one of the Proof Of Concepts (POCs) at a CRESCENDO project workshop. For the case study, AirCADia and Isight/SEE [9] were used as the candidates for computational workflow exchange. AirCADia is a flexible generic tool for complex systems early design, analysis and optimization, and is based on the Workflow Management Device (WMD) kernel presented in [11,12]. Unlike existing capabilities for model-based design which link (high-fidelity) tools in a predetermined sequence, AirCADia dynamically builds a computational workflow depending on what variables are given to the system as inputs by the analyst. Isight/SEE is a tool which provides an extremely flexible framework for assembling and managing ready-to run computational workflows. A sample computational workflow automatically configured using AirCADia was exported in XML format to Isight. The workflow was then re-created inside

Isight/SEE, by using an XML parser program and an appropriate translator. Figure 4 shows a schematic representation of the case study.

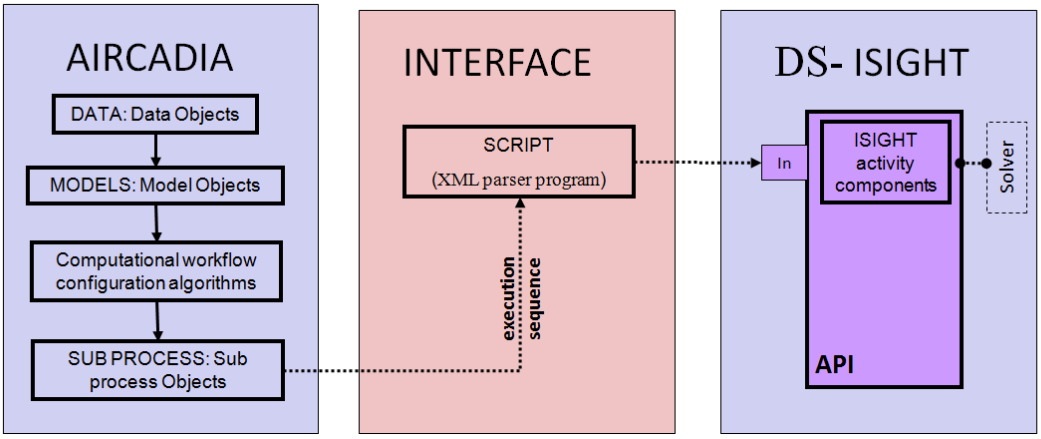


*Figure 4. Schematic representation of the case study*

A sub-set of models from an aircraft sizing test-case, Ultra Simplified Model of Aircraft (USMAC), was used for this case study [7]. USMAC is an edited aircraft conceptual design test case supplied by a major aircraft manufacturer, consisting of a set of 97 models and 125 variables. Despite being simplified, USMAC is still representative of a complex multi-physics system description based on non-linear models. The illustrative sub-set considered in this case study consists of only 4 models used for aircraft fuselage and tail sizing. In AirCADia, these models were put together to form a sub-process. The computational workflow scheduling algorithms developed during earlier research [7] were used to schedule the execution sequence. Figure 5 - the top half, shows a matrix representation of the workflow in Incidence Matrix (IM) form [1], where the models appear in the rows and the variables in the columns. The shaded entries (green and red) indicate that a variable is an input to or an output from a model(s), respectively. The graphical representation of the same execution sequence is shown in the bottom half of Figure 5.

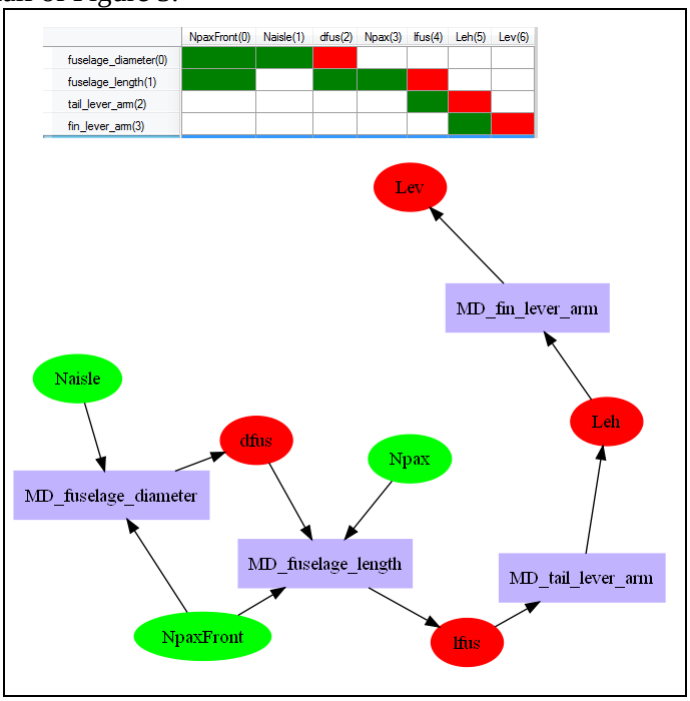


*Figure 5. IM and graphical representation of the sub-process*

This computational workflow created in AirCADia was exported in the XML format to another partner at a different location. Figure 6 shows a screenshot of the XML file with different class containers. Since there are no SCCs, Modified models, Treatments, and Studies in the workflow considered for the case study, the corresponding containers appear empty in the XML representation.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Project>
  - <DataContainer>
    - <Data>
        <Name>dfus</Name>
        <Value>4.1</Value>
        <Type>double</Type>
        <Unit />
        <Min>2.05</Min>
        <Max>6.15</Max>
      </Data>
    - <Data>
        <Name>NpaxFront</Name>
        <Value>6</Value>
        <Type>double</Type>
        <Unit />
        <Min>3</Min>
        <Max>9</Max>
      </Data>
    - <Data>
        <Name>Naisle</Name>
        <Value>1</Value>
        <Type>double</Type>
        <Unit />
        <Min>0.5</Min>
        <Max>1.5</Max>
      </Data>
    - <Data>
        <Name>Lev</Name>
        <Value>19.8009875</Value>
        <Type>double</Type>
        <Unit />
        <Min>9.90049375</Min>
        <Max>29.70148125</Max>
      </Data>
    - <Data>
        <Name>Leh</Name>
        <Value>19.8009875</Value>
        <Type>double</Type>
        <Unit />
        <Min>9.90049375</Min>
        <Max>29.70148125</Max>
      </Data>
    - <Data>
        <Name>lfus</Name>
        <Value>40.25</Value>
        <Type>double</Type>
        <Unit />
        <Min>20.125</Min>
        <Max>60.375</Max>
      </Data>
    - <Data>
        <Name>Npax</Name>
        <Value>150</Value>
        <Type>double</Type>
        <Unit />
        <Min>75</Min>
        <Max>225</Max>
      </Data>
    </DataContainer>
  - <ModelContainer>
    - <Model>
        <Name>fuselage_diameter</Name>
      - <InputContainer>
          <NpaxFront />
          <Naisle />
        </InputContainer>
      - <OutputContainer>
          <dfus />
        </OutputContainer>
        <Equation>dfus=(1.1 * NpaxFront -0.6 * Naisle -1.9);</Equation>
        <Auxiliary>false</Auxiliary>
      </Model>
    - <Model>
        <Name>fin_lever_arm</Name>
      - <InputContainer>
          <Leh />
        </InputContainer>
      - <OutputContainer>
          <Lev />
        </OutputContainer>
        <Equation>Lev=Leh;</Equation>
        <Auxiliary>false</Auxiliary>
      </Model>
    - <Model>
        <Name>tail_lever_arm</Name>
      - <InputContainer>
          <lfus />
        </InputContainer>
      - <OutputContainer>
          <Leh />
        </OutputContainer>
        <Equation>Leh=(( -0.0002 * lfus + 0.5 ) * lfus);</Equation>
        <Auxiliary>false</Auxiliary>
      </Model>
    - <Model>
        <Name>fuselage_length</Name>
      - <InputContainer>
          <dfus />
          <Npax />
          <NpaxFront />
        </InputContainer>
      - <OutputContainer>
          <lfus />
        </OutputContainer>
        <Equation>lfus=5 * dfus + (0.75 * Npax/NpaxFront + 1);</Equation>
        <Auxiliary>false</Auxiliary>
      </Model>
    </ModelContainer>
  - <SubprocessContainer>
    - <Subprocess>
        <Name>sample_case</Name>
      - <InputContainer>
          <NpaxFront />
          <Naisle />
          <Npax />
        </InputContainer>
      - <OutputContainer>
          <dfus />
          <Leh />
          <Lev />
          <lfus />
        </OutputContainer>
      - <ExecutionSequence>
          <fuselage_diameter />
          <fuselage_length />
          <tail_lever_arm />
          <fin_lever_arm />
        </ExecutionSequence>
      </Subprocess>
    </SubprocessContainer>
    <SccContainer />
    <ModifiedModelContainer />
    <TreatmentContainer />
    <StudyContainer />
  </Project>
```


*Figure 6. Computational workflow description in XML format*

At the partner's location, the XML file is parsed using a tailor-written *groovy script*, which is a dynamic language for Java Virtual Machine (JVVM) [10] to replicate the same workflow in Isight/SEE. The equations for these models were automatically converted to either pre-defined MATLAB or Excel components in Isight/SEE. Figure 7 shows the screenshot of the same computational workflow in the Isight/SEE environment.

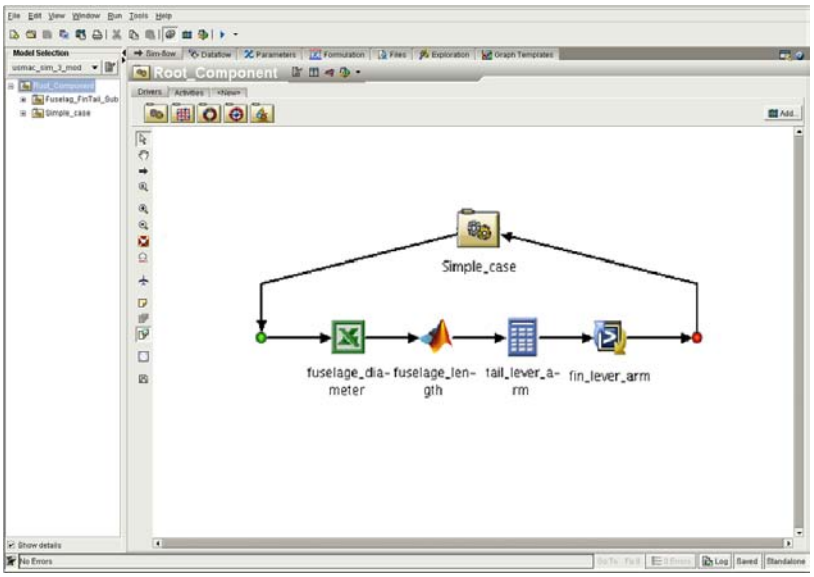


*Figure 7. Reproduction of the same computational workflow in ISIGHT*

As discussed earlier, representing the workflow in a neutral format essentially de-couples the two tasks, workflow configuration and workflow execution. To demonstrate this further, the model

'fin_lever_arm' from the original workflow, which calculates the length of fin lever arm, was replaced by a 4-step process to calculate the same output, but this time with a higher model fidelity. As long as the input and output of the new detailed 4-step process are kept the same as in the original model, the same computational workflow can be used. Figure 8 shows the screenshot of Isight/SEE with the updated process which has the same execution sequence but with modified execution details.

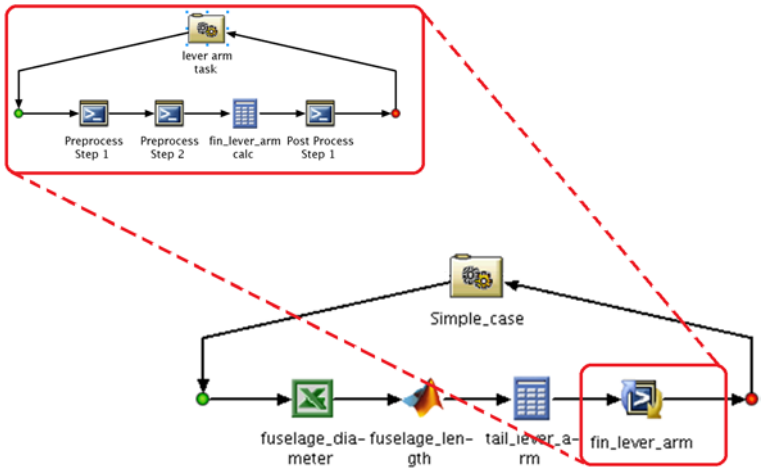


*Figure 8. Replacing a component in the original workflow with 4 step process*

## 4. CONCLUSIONS

The proposed approach to neutral description of computational workflow enables the exchange of computational workflows between different design partners. It also allows workflow logic to be effectively decoupled from the execution details. Such decoupling is extremely useful in the cases of collaborative design, where partners intend to use proprietary methods and tools for executing some parts of the workflow.

The proposed approach also enables to couple simulations of different granularity in the same workflow. The XML representation had been found to be a structured way of sharing simulation object models, and the availability of well tested XML parsing routines in different programming language is expected to make it a preferred choice for workflow description and exchange.

The heart of this research is the simulation object model, which is currently restricted to cater for low-fidelity simulations used predominantly during the early design stages. A challenging task lies ahead in the CRESCENDO project – to develop a global object model, which will enable complete range of simulations, spanning the whole product development cycle. It is expected that these efforts will result in a standard for computational workflows description, in-line with existing data exchange standards such as STEP [14].


## ACKNOWLEDGEMENTS

The research reported in this paper has been carried out within the CRESCENDO Integrated Project, which is partly sponsored by the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement No: 234344 (www.crescendo-fp7.eu)

The authors would like to thank our industrial partners for the fruitful discussions during the research and development of this concept.

## REFERENCES


[1] Balachandran, L. K. and Guenov, M. D., Object Oriented Framework for Computational Workflows in Air-vehicle Conceptual Design., in *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, September 2009, Hilton Head, South Carolina

[2] White, S. A. and Miers, D, *BPMN Modeling and Reference Guide: Understanding and using BPMN*, Future Strategies Inc., Lighthouse Pt, FL, 2001.

[3] *ISO standard for the representation of process plant life-cycle information*, International Organization for Standards, 2009. ISO 15926.

[4] *Object Management Group (OMG)*, Unified Modelling Language, http://www.uml.org/

[5] *w3schools*, http://www.w3schools.com/xml/default.asp.

[6] Brandes, U., Eiglsperger, M., Lerner, J., *GraphML primer*, GraphML Project Group, 2004.

[7] Balachandran, L. K. , Fantini, P. F. , Guenov, M. D., Computational Process Management for Aircraft Conceptual Design, in *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, September 2007, Belfast, Northern Ireland.

[8] Balachandran, L.K. and Guenov, M.D., Computational Workflow Management for Conceptual Design of Complex Systems, *AIAA Journal of Aircraft*, 2010, Vol. 47; No 2; pp. 699-702.

[9] *Isight and Simulia Execution Engine (SEE)*, Dassault systemes, http://www.simulia.com/products/isight.html

[10] *Groovy, An agile dynamic language for Java Platform*, http://groovy.codehaus.org/

[11] Guenov, M. D., Fantini, P. F., Balachandran, L. K., Maginot, J. P., Padulo, M, and Nunez, M., Multidisciplinary Design Optimization Framework for the Pre Design Stage, *Journal of Intelligent and Robotic Systems (Springer)*, 2010, Volume 59, Issue 3 , pp 223-240.

[12] Guenov, M.D., Fantini, P., Balachandran, L. K., Maginot, J. P., and Padulo, M., MDO at Pre Design Stage, in Kesseler, E. and Guenov M.D (Eds), Advances in Collaborative Civil Aeronautical Multidisciplinary Design Optimization, *Progress in Astronautics and Aeronautics Series, 233 Published by AIAA*, 2010.

[13] CRESCENDO Project No.: 234344, Call Identifier: FP7-AAT-2008-RTD-1, *Description of Work (DoW)*, R1.2 - 09/09/2009

[14] Pratt, M. J., Introduction to ISO 10303-the STEP standard for product data exchange, *ASME Journal of Computing and Information Science in Engineering*, 2001, Volume 1, pp. 102–103.