Automated generation of Isight-Models through a neutral workflow description

Wenzel H.

Dassault Systems SIMULIA

Gondhalekar A., Balachandran L., Guenov M., Nunez M.

Cranfield University

Abstract: At the Cranfield University, a tool named AirCADia has been developed. It enables the dynamic assembly of hierarchical computational processes from sub processes and/or from atomic models. The analyst selects the input variables and specifies the output variables for a particular design study. AirCADia then assembles the necessary simulation sequence automatically, based on the models available in it's library. In addition, AirCADia enables the dynamic application of relevant numerical treatments, such as multi-objective and robust optimization. Currently most of the models in AirCADia support only lightweight simulation tasks, typically used in aircraft preliminary design. The workflows created by AirCADia can be represented and exported in an abstract XML-notation.

Isight from Dassault Systems SIMULIA is a very powerful and flexible tool to build simulation workflows (simflows) ready for execution in a specific environment. It supports both lightweight simulation as well as full blown 3-dimensional FE or CFD simulation. There are situations, however, where the manual creation of the simulation process and data flow through the Isight-GUI gets too tedious and error prone. An example of this situation is the creation of simflows in preliminary aerospace design tasks, where tens to hundreds of instantiations of simulation components and thousands of parameters need to be managed.

In the EU FP7 project CRESCENDO a transformation tool has been developed that converts AirCADia workflows into Isight models. The transformation tool allows AirCADia workflows to be executed using the Isight-Library., as long as the interface, namely the input- and output-parameters and the component names are consistent. In this approach the workflow logic is effectively decoupled from the implementation details.

Also it is now easy to share simflows between different departments in one company or between different companies that might use different tools to perform the actual simulation and to migrate simflows from conceptual to preliminary, and to detailed design. Furthermore it is possible for the workflow creation experts to generate MDO-schemas using lightweight closed form simulations and for the simulation experts to generate robust simulation models. Only after both the logic and the implementation are mature, they are merged.

1. Introduction

In today's simulation departments much of the intellectual property (IP) is build around the usage of their simulation tools: how to call them, what parameters to tweak, how to use the material models etc. To keep their competitive advance, companies naturally want to keep that IP to themselves.
At the same time, with the products getting more and more complex, the time to market shorter, and the customer demands higher, the companies are forced to collaborate with their customers, suppliers and sometimes even with their competitors at all stages of the product design. To make these collaborations as effective as possible, each of the partners prefer to use their own tools and methods.
These two conflicting market requirements, IP-protection and need for collaboration, have given rise to standards representing data in a neutral/application independent format. Standards like STEP (1) (Standard for Exchange of Product model data) are, for example, widely followed in CAD/CAE community.
In a more complex design/analysis scenario, there is usually a set of models which are executed in a predefined sequence for the computation of specific outputs required in engineering decision making processes. This sequence of execution of simulation models is called "simulation workflow" or "simflow". There are some commercial tools

,such as Isight by Dassault Systems (2), to create simulation workflows manually. The research group at Cranfield University (3) is working on algorithms to automatically configure and execute simulation workflows at conceptual design stage), and a neutral description to facilitate their exchange in collaborative environment (4).. Sometimes when working in a collaborative design team, the partners based at different geographical locations may need to execute the same sequence of models. The tools used to execute each model may be different for different partners. In such cases, it is important that this simulation workflow is represented in a neutral format which can be shared with all partners, and can be parsed by computer programs.
Here the simflows can be shared in a "grey box" approach. This is in contrast to a black box approach, where the partner only sees the inputs and the outputs of a workflow, with the execution sequence totally opaque to him, and a white box approach, where he has access to every detail of the simflow . In a grey box the partner can see the execution sequence and the parameters of each execution step, but nothing else.
The idea of having a standard and neutral description for a process is discussed in literature in relation to process industry and business process modelling. There are standards like BPMN (Business Process Modelling Notation) set by OMG group (5) which uses graphical notations to represent business processes in a neutral way, independent of the implementation environment. There is also an ISO standard for representation of process plant lifecycle information (6). One can argue that these available standards can be used to represent simulation workflows. The workflow objects however, like models, treatments, and methods (e.g., solving simulation models, transforming data, Design of Experiments (DoE) techniques, optimization algorithms etc.) which are encountered in simulation processes, do not appear in process modelling or BPMN. Thus, for representing simulation workflows in a neutral format, there is a need to define a set of classes and their attributes which are specific to simulation workflows.

2. Building Blocks for a Neutral Description of Simulation Workflows

To describe a simulation workflow in a neutral language, it is necessary to have common definitions for simulation objects. To be of practical use, these objects have to be on a suitable level of abstraction. They need to be general enough to be applicable to all relevant use cases, yet they need to be concrete enough for a concise and precise description of engineering simulation workflows.
This section defines different simulation objects in a typical simulation workflow, and proposes the attributes for each of these objects.

2.1 Parameters

Parameters are constant scalars which constitute the most fundamental object in the simulation workflow class hierarchy. Parameters can possess different attributes, listed in the table below.

Attribute	*Data Type*
name	string
nominal value	double/integer
min value	double/integer
max value	double/integer
unit	string

2.2 Data

Data is an aggregation of parameters into a specific structure, an array, a matrix, a binary or text file. Data attributes are given in the table below.

Attribute	*Data Type*
name	string
type	string
comment	string

The "type" attribute specifies the type of data held in the object. Options for type can be scalar, vector, tensor, matrix, file, image, etc.
The "comment" attribute gives more information on the data, like how the parameters are arranged, name of the tool which is used to arrange it, validity of the data, etc.

Figure 1: An example of instantiated parameter and data objects

Figure 1 shows an example of instantiated parameter and data objects and their attributes.

2.3 Model

A Model is an object in simflows which takes parameters and/or data as inputs, executes specific computational instructions and provides parameters and/or data as outputs.

Attribute	*Data Type*
name	string
inputs	array of string
outputs	array of string
validity	string

The attribute "validity" is useful when replacing one model in the workflow with another valid model.

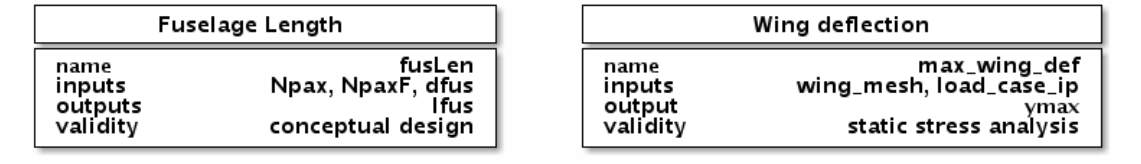

Figure 2: An example of instantiated model objects

Figure 2 shows examples of two different initialized model objects. The one on the left is a model to calculate fuselage length. It takes different parameters as inputs and executes an empirical equation to provide the length of fuselage as output. This model is valid for aircraft sizing at conceptual design stage. The one on the right is a FE model to calculate maximum wing deflection. It takes two data objects: wing mesh and load case input file as inputs. It gives the maximum deflection as output. The validity of this model is for static stress analysis only.

2.4 Modified Model

In the product design process, it is sometimes necessary to reverse the inputs and outputs of a model. In cases when an explicit equation is available, it may be possible to invert the model analytically. In other cases where it is not possible (e.g., when a numerical code for FE/CFD analysis or a black box are involved), the model can be reversed by applying numerical methods like Guess-Newton, or fixed point iteration (3). Such reversed models can be part of a simulation workflow. They are described using the object called modified model.

Attribute	*Data Type*
original model name	string
changed i/o	array of string

Figure 3 - the top half, exemplifies the concept of modified model by taking an example of a black box model to calculate the length of the fuselage with 3 different inputs. The same model can be used to calculate the diameter of the fuselage by reversing it, and adding a numerical treatment on it as shown in the bottom half of Figure 3.

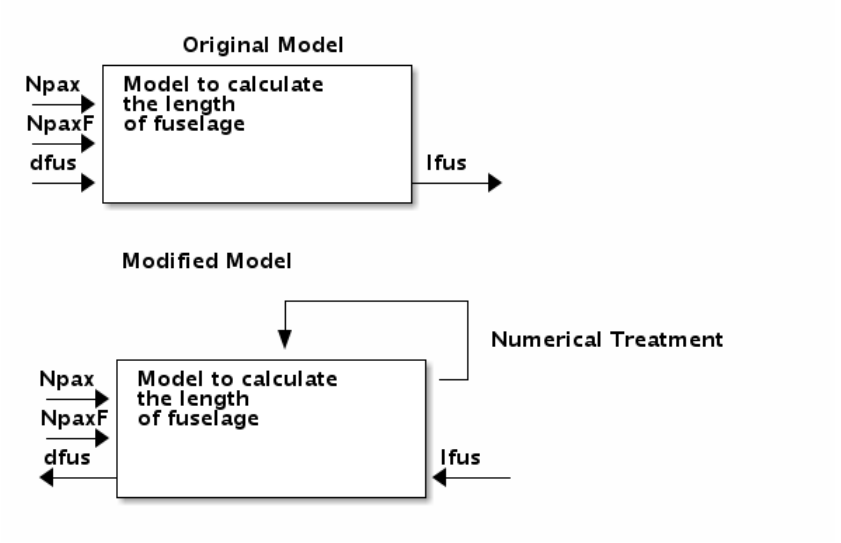

Figure 3: Concept of a modified model

2.5 Sub-process

A sub-process is an object which stores a collection of models or sub-processes grouped together for a specific purpose. For example, in aircraft conceptual design, models related to aircraft geometry can be grouped together in one sub-process, and models related to engine can be grouped in another sub-process.

Attribute	*Data Type*
name	string
list of models	string array

Figure 4 shows an example of instantiated sub-process object in which different models for aircraft geometry to calculate wing area, horizontal tail area, vertical tail area, length of fuselage, diameter of fuselage etc. are grouped together.

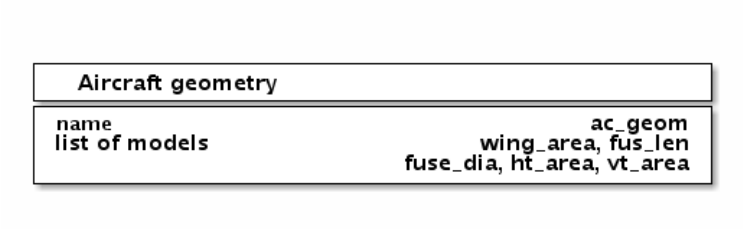

Figure 4: Instantiated sub-process

2.6 Strongly Connected Components (SCC)

When models are grouped together to form a sub-process with user defined inputs and outputs, some parameters and data may form feedback loops.
A typical example is the deformation of an airplane wing in flight, if the flow calculation and the structural calculation are performed by two different, loosely coupled models. The flow calculation predicts the pressure distribution on the wing surface, which is the input to the structural calculation that gives the deformation of the wing. This deformation of the wing in turn influences the flow field. Therefore the physical phenomenon needs to be solved iteratively.

For numerical efficiency, the models can be re-arranged so as to minimize the number and the length of feedback loops. 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.

Attribute	*Data Type*
name	string
list of models in execution sequence	array of string

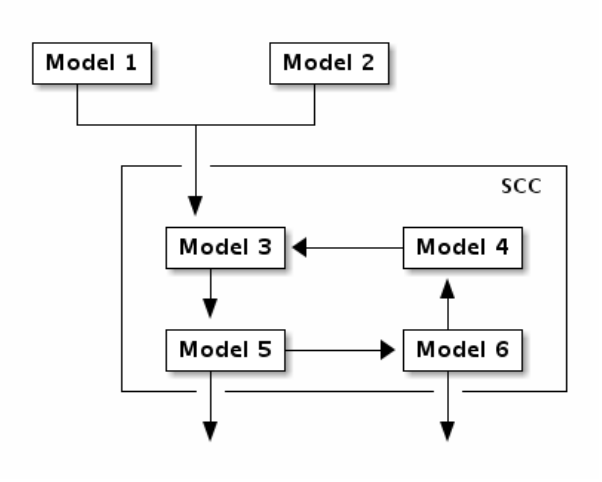

Figure 5: Example of a strongly connected component

It can be seen from Figure 5 that Model 6 feeds back to Model 3 via Models 4 and 4. Thus, in this case, Models 3, 4, 5 and 6 constitute a strongly connected component within the sub-process.

2.7 Treatment

A treatment object consists of mathematical tools which can be applied at different levels in the hierarchy of a workflow. A treatment can act on parameters, data, models, or sub-processes. Examples of a treatment can be a meshing treatment applied on geometrical data, a numerical differentiation treatment applied on a model, or an optimization treatment applied on a sub-process

Attribute	*Data Type*
name	string
inputs	array of string
outputs	array of string
user definable	array of string
context	**string**

The “inputs” attribute refers to parameter, data or model objects, the “output” attribute can consist of parameter or data objects. The attribute “user definable” contains any specific parameters for the treatment. For example, the user definable attribute may contain information about integration step size for numerical integration treatment, or it may contain tuning parameters like number of generations, population size, etc. for a genetic algorithm optimization treatment.

2.8 Simulation Workflow

The highest level object in the class hierarchy is a simulation workflow. It is described using data, models, modified models, SCCs, sub-processes, treatments which contribute to the workflow. It also states the sequence of execution of these entities.

Attributes	*Data Type*
name	string
sequence of execution	string array

3. XML representation

The Extensible Markup Language (XML) is very well suited to represent hierarchical information, such as the neutral workflow description proposed in this work. Since it also is easily processed by computer programs, it is the natural choice to be used as the exchange platform when the workflows are shared between partners.

In the proposed XML format, each building block in the simulation workflow hierarchy is represented by a respective container. Thus, the XML file will have different containers like parameter containers, data containers, model containers, modified model containers, sub-process containers, treatment containers, and workflow containers. The objects of a class are stored in the respective container. Each container takes the structure of the respective class, and lists all attributes of that class.
An example of such an XML representation is given in the following case study.

4. Case study

For the case study, AirCADia and Isight/SEE were used as the candidates for computational workflow exchange. AirCADia is a flexible generic tool for complex systems early design, analysis and optimization, and it is based on the WMD kernel presented in (9,10). 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 and which parameters are requested as outputs. Whereas in Isight/SEE the execution sequence is prescribed manually in the Isight Graphical-User-Interface (GUI), and the parameter flow is established only afterwards by mapping the parameters from one component in the sequence to the other, AirCADia works the other way round. The user specifies the outputs of interest and the inputs he/she wants to prescribe. AirCADia then refers to it's library of models and sub-processes and determines the execution sequence by which the outputs can be calculated from the inputs.

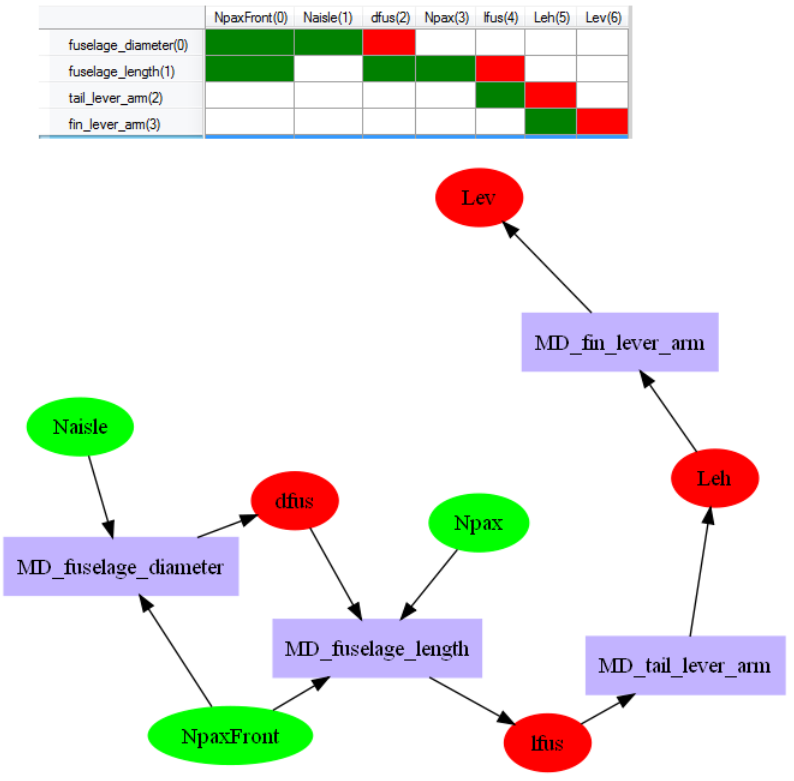

Figure 6: Incidence Matrix and graphical representation of the sub-process

A sub-set of models from an aircraft sizing test-case, Ultra Simplified Model of Aircraft (USMAC), was used for this case study (9). 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, which are representative of real design procedures. 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 were used to schedule the execution sequence. Figure 6 - 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 6.

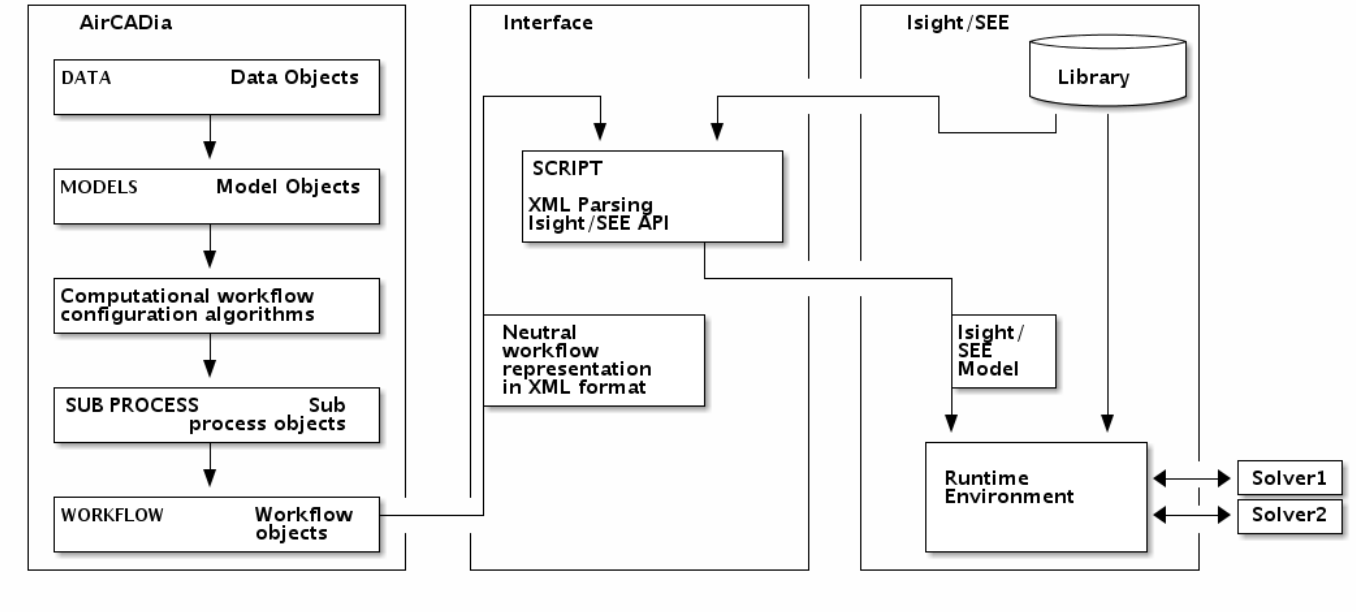

Figure 7: Schematic representation of the case study

This computational workflow created in AirCADia was exported in the XML format to another partner at a different location. Figure 8 shows a screenshot of the XML file with different class containers. As there are no SCCs and no mathematical treatments in the workflow considered for the case study, the corresponding containers do not appear in such XML representation example.

Figure 8: Neutral computational workflow description in XML format

At the partner's location, the XML file is parsed using a tailor-written Groovy-script, a dynamic language for Java Virtual Machine (JVM) to replicate the same workflow in Isight/SEE, see figure 7. The equations for these models were automatically converted to either pre-defined MATLAB or Excel components in Isight/SEE, published to Isight/SEE's library. Figure 9 shows the screenshot of the same computational workflow in the Isight/SEE environment.
Groovy was used here for its strong XML-processing features and the fact that it runs on a JVM and therefore can access the Isight-API directly.
Hiding all implementation details, the following pseudo-code shows the structure of the conversion script.

```
function recursiveAddComponent(component, parent_component) {
   add_parameters_to_component
   if (component is sub-process){
      put_component_in_Isight_model
      forall(subcomponent in sub-process-model-list){
        recursiveAddComponent(subcomponent,component)
      }
   } else {
      put_component_in_Isight_model
   }

}
```

parse_XML_file
recursiveAddComponent(workflow,null)
The function "*recursiveAddComponent*" is called on each workflow, sub process and model object in the XML-file. It calls the function "*put_component_in_Isight_model*" that looks up the right component in the Isight-library and inserts that the. into the Isight-workflow. Then it adds the associated parameters and data objects to the Isight component.

If the current model is of type workflow or sub-process the function calls itself recursively on all models or sub processes below.

The script starts by parsing the XML-file and building up the lists of sub-processes, models and parameters. Then the actual conversion starts by calling the "recursiveAddComponent" function on the workflow-object of the neutral description.

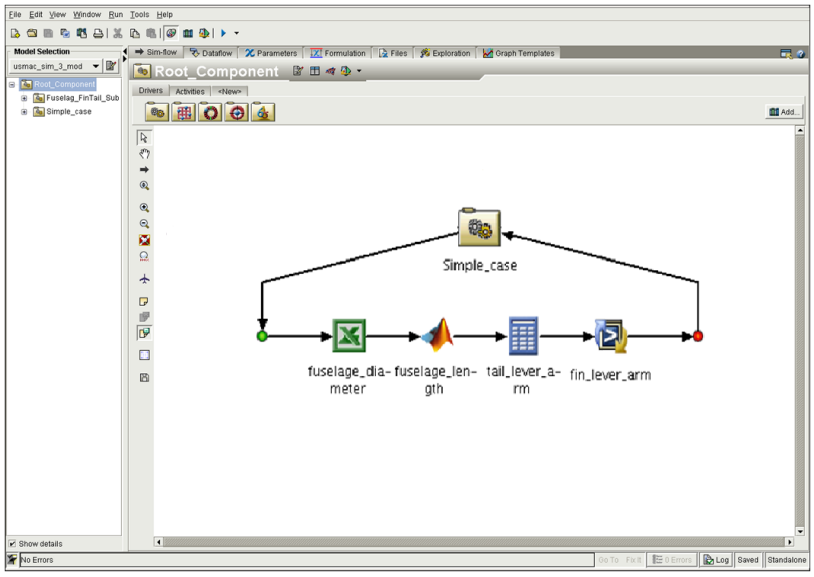

Figure 9: Instantiated workflow in Isight

As discussed earlier, representing the workflow in a neutral format essentially decouples the two tasks, logical workflow configuration and detailed execution specification. 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 fidelity model. 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 10 shows the screenshot of Isight/SEE with the updated process which has the same execution sequence but with modified execution details.

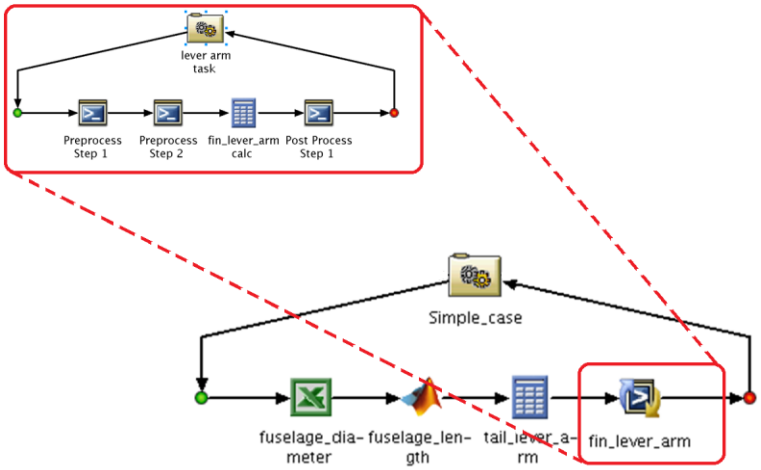

Figure 10: Replacing a component in the original workflow with 4 step process

5. Summary

Since product design has probably become more collaborative and more complex than ever before, there is a strong need to formalize simulation processes used in the product design. This work proposes a way that enables not only the sharing of static data, but also of these simulation processes between partners in a grey box approach, where the sequence of the simulations and the key inputs and outputs are exposed, while all implementation details are hidden. This approach not only enables the exchange of simflows between companies, but also throughout the engineering product design cycle, e.g. between conceptual and preliminary design. Making the workflow available to someone else has advantages even for performing a single step in the sequence, since the workflow provides the simulation context and makes it easier to judge the validity of its models and results.

To facilitate this, a minimal set of building blocks for simulation workflows and their relationships has been defined in this work. The feasibility of this concept has been demonstrated in a case study from the context of the CRESCENDO project.
One of the findings of this case study was that it is necessary to make the object model of the neutral simulation workflow description easily extendable. An additional attribute "equation" has been added to the object "model", so that a closed form equation could be communicated between different partners.
For real industrial applications the simple mapping through a naming convention that was employed in the use case might be replaced by using more descriptive languages like RDF (7) or OWL (8). This will enable users to specify more complicated relationships between the building blocks of their simflows, like synonymous objects, equivalences etc. Such information-rich description of workflows will be useful in replacing an object in the existing workflow with another equivalent object which is available at other partner's workplace, yet using a different taxonomy.

6. Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 234344 (www.crescendo-fp7.eu/).

7. Bibliography

1. ISO standard for the exchange of product model (STEP) ISO 10303. s.l. : International Organization for Standards. ISO 10303.
2. Isight. SIMULIA. [Online] Dassault Systems. http://www.simulia.com/products/isight.html.
3. Computational Process Management for Aircraft Conceptual Design. Balachandran L., Fantini P., Guenov M. s.l. : 7th AIAA ATIO Conference, 2007.
4. Gondhalekar A. C., Guenov M. D., Wenzel H., Nunez M., Balachandran L. K., "Neutral Description and Exchange of Design Computational Workflows", (submitted),18th International Conference on Engineering Design (ICED), Copenhagen, Denmark, August-2011.
5. White S. A., Miers D. BPMN Modeling and Reference Guide: Understanding and using BPMN. s.l. : Future Strategies Inc., Lighthouse Pt, FL, 2001. ISBN-13: 978-0977752720.
6. ISO standard for the representation of process plant life-cycle information. s.l. : International Organization for Standards, 2009. ISO 15926.
7. Klyne G., Carroll J. Resource Description Framework (RDF): Concepts and Abstract Syntax. s.l. : The World Wide Web Consortium (W3C), 2004.
8. McGuinness D. L., Harmelen F. OWL Web Ontology Language Overview. s.l. : The World Wide Web Consortium, W3C, 2004.

9. 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, ISBN-10: 1-60086-725-1; ISBN-13: 978-1-60086-725-5
CRESCENDO Project No.: 234344, Call Identifier: FP7-AAT-2008-RTD-1, Sescription of Work (DoW) R1.2 - 09/09/2009