CHALLENGES IN EUROPEAN AEROSPACE
5TH CEAS AIR & SPACE CONFERENCE

CEAS

A multidisciplinary design optimization advisory system for aircraft design

Maurice F. M. Hoogreef
Flight Performance and Propulsion, Faculty of Aerospace Engineering, Delft University of Technology
Ph.D. candidate, Kluyverweg 1, 2629 HS, Delft, The Netherlands, M.F.M.Hoogreef@tudelft.nl
Roberto d'Ippolito
Noesis Solutions N.V., Research and Innovation Manager
Robin Augustinus
Flight Performance and Propulsion, Faculty of Aerospace Engineering, Delft University of Technology, M.Sc. student
Gianfranco La Rocca
Flight Performance and Propulsion, Faculty of Aerospace Engineering, Delft University of Technology, Assistant Professor

ABSTRACT

This paper presents the status of development of an MDO advisory system, coupled to an optimization configurator for a process integration and design optimization (PIDO) system. This advisory system can support non-experts in the application of MDO by giving advice on the type of MDO architecture to use and assist in the implementation of the actual problem inside a PIDO system. Both the advisory system and optimization configurator are implemented by means of knowledge based technologies that allow for the inclusion of semantics to data and allow for semantic reasoning. An overview of the functionalities of the system and the background technology that is involved is presented in this paper. A use case is presented to illustrate part of the capabilities. This use case focuses on the (re)generation of (partial) MDO workflows. It is demonstrated that based on, user-specified, desired outputs only that part of a workflow that is relevant to these outputs can be automatically generated inside the PIDO system and executed.

1 INTRODUCTION

1.1 Context and Motivation

Today's latest aircraft in service (A380 and B787) and in production (A350, C-Series) show both an outstanding level of performance and a reduced impact on the environment. This is the result of a continuous improvement of the airframe and engine design in particular.
However, such performance improvements have been achieved thanks to the progressive mastering by engineers of the different interactions between systems and disciplines and a continued application of the classical iterative engineering approach, rather than by the implementation of a systematic Multidisciplinary design optimization (MDO) approach. The required step changes in performance, especially those concerning environmental impact, cannot be achieved without the development and implementation of new design methods and tools that take into consideration and exploit the various multi-disciplinary synergies in order to optimize the complete product (aircraft, rotorcraft, other) and its sub-systems according to specific criteria. MDO can provide designers with the means to further improve the performance of already mature solutions and support the exploration of innovative complex designs. However, the application of MDO to such complex cases is not feasible without the prior development of suitable methods and tools to handle high dimensional, heterogeneous product models and knowledge and provide efficient means for collaborative distributed design.

1.2 Problem Description

Although MDO has been around for quite some time, it has not yet been used up to its full potential in industry. Some of the reasons for the limited exploitation of MDO in industry are related to the intrinsic complexity of MDO and its mathematical foundations together with the lack of awareness and understanding of the many available MDO architectures and their specific suitability to problems of different natures.
This paper describes the current state of development of an innovative MDO advisory system, devised with the purpose of lowering the entry level of MDO technology and increasing it exploitation in industry. The MDO advisory system enables the user, not necessary a MDO expert, to specify first the design problem at hand and then, on the basis of previously stored knowledge, provides him/her with a ranked list of suitable MDO architectures, with (links to) relative documentation. The advisory system is coupled to an optimization configurator that can generate the executable workflow and solve the optimization problem. The coupled system supports the user with the technical implementation of the formulated problem and, finally, its execution, by means of a commercial process integration and design optimization (PIDO) tool.

1.3 Structure of paper

The body of this paper is split into 3 main sections. Section 2 provides background information on the current state of the art related to MDO, simulation workflow management (SWFM), process integration and design optimization (PIDO) and semantic web technologies, these being the cornerstones of the proposed advisory system. Section 3 describes the MDO advisory system currently being developed. This includes the overall design rationale of the system, the different subcomponents and the design processes supported by the system itself. In Section 4 the technical implementation of the system is addressed with an assessment of some features of the system in a preliminary stage. Conclusions and plans for future development are then illustrated in the closing section.

2 BACKGROUND

This chapter provides a brief overview of the background material for this paper. An introduction to MDO is presented, together with the current state of the art. Furthermore a description of SWFM and PIDO systems and their state of the art is provided, followed by an introduction to semantic web technologies. The section is concluded by an overview of the limitations of the current application of optimization in industry, which need to be overcome by the MDO advisory system.

2.1 Multidisciplinary Design Optimization

Multidisciplinary design optimization (MDO) is a methodology that combines analyses and optimizations contained within individual disciplines with the analyses and optimizations contained within the entire system of the disciplines. One of the earliest notions of the term "multidisciplinary optimization" can be found in a technical memorandum from the National Aeronautics and Space Administration (NASA), by Jaroslaw Sobieszczanski-Sobieski, from 1989 [1]. This technical report introduced a hierarchic decomposition for a design problem, instead of the conventional sequential design process. Not much later, the Technical Committee for MDO (TC-MDO) was established by the American Institute for Aeronautics and Astronautics (AIAA), which, in 1991, published the first out of two white papers on the current state of the art in MDO [2]. This white paper provides an overview of the conceptual components of MDO. Sobieszczanski-Sobieski provided the first taxonomy of MDO in another NASA technical memorandum in 1993 [3]. This report examines the definition of MDO and the components that together make up the MDO engineering discipline.
The way of organizing an MDO problem is often referred to as MDO architecture. The architecture describes the couplings between disciplinary models and optimizer(s), i.e. the structure of the optimization problem, and how this problem is solved. This should not be confused with the

optimization algorithm, which is the set of (mathematical) steps required to solve the optimization problem. Different MDO architectures exist, which differ in the way they suit optimization problems of different nature. As Alexandrov and Kodiyalam [4] note, to fully use all advantages of a specific architecture would require extensive tuning of the problem statement and its implementation. However, this can provide too much overhead for a conceptual design phase, or for a problem that will be solved only once or a few times.
In 2013, Martins and Lambe published an extensive survey of MDO architectures [5]. This survey lists most, common architectures with a unified description and includes a classification based on the problem formulation and decomposition strategy used, as well as strengths and weaknesses of the architectures. The focus is primarily on methods that solve MDO problems with a single objective function and continuous design variables. The assumption is that optimality of a solution corresponds to satisfying the Karush-Kuhn-Tucker (KKT) optimality conditions and that the objective and constraint functions are differentiable. Next to this, there are many examples in literature that focus on the problem definition, such that a specific architecture can be used [6] [7] [8] [9] [10] [11] [12] [13]. There is, however, little advice on which architecture to use for a specific problem. Also, in modern day MDO frameworks, this advisory functionality is rarely present and if is present at all, its advice is limited to suggesting the type of algorithm (e.g. gradient based or evolutionary). [14] [15]
Some of the reasons for the limited exploitation of MDO in industry, in contrast with the growing interest and knowledge that is developing in academia, are related to the intrinsic complexity of the discipline and its mathematics, together with a lack of awareness and understanding of the many available MDO architectures and their specific suitability to problems of different nature. The implementation of an advisory system that is capable of advising users in their decision for a certain MDO architecture and optimization algorithm in relation to the problem at hand would also address another point that is commonly found in MDO literature, "... the lack of education [in MDO] itself, not only at the university level, but also within industry and research organizations." [16]. This lack of education in MDO is also discussed in [17] & [18]. To the author's knowledge, there is currently no system that assists the user in the selection of a suitable MDO architecture for the problem at hand, aids in the appropriate definition of the optimization problem and helps to integrate the MDO architecture in an executable workflow.

2.2 Simulation Workflow Management

A workflow can be defined as a sequence of activities where each of them follows the previous one without any gaps or delays in switching between activities, in order to achieve a specified goal. Thus, workflows can be seen as an abstraction of any real work. Workflow management systems allow users to define, modify, control, and share workflows and automate iterative tasks. They are used to integrate different systems or pieces of software.
In engineering, two main workflow categories can be identified: business process workflows and simulation (or computational) workflows. For simulation workflows, and similarly for business process workflows, a big effort has been made to enable their interoperability and reusability. Decoupling (or at least loose coupling) of the workflow logic from the implementation has become fundamental for information reuse. Therefore, it is important to have a neutral description of the simulation workflows. In most cases, this is achieved by adopting an XML representation of the workflow. As long as the inputs and outputs of the process are kept the same, the workflow can be executed on different platforms and exchanged across different organizations. Simulation Workflow Management (SWFM) systems generally support distributed computations to achieve, e.g., cross-organization simulation collaboration, execution time reduction, and reuse of simulation assets.
All workflow management systems are based on a formalism or language that can support different features. In [19] a set of requirements that a workflow management system has to satisfy have been expressed in form of patterns that are often used to compare different workflow systems. Van der Aalst [20] introduced YAWL, a new workflow language inspired by Petri-nets that allows a direct and intuitive support of the patterns proposed in [19].

Overall, some key trends have emerged for workflow systems and represent the current state of the art. These trends are:

- Standardization of the workflow description, particular attention is given to the XML format;
- Multi-level workflows that allow management of problem decomposition;
- Deep integration of distributed computation approaches (different from parallel computing)
- Support new, smarter workflow operations such as: synchronization, cancellation and termination as well as multi-instances.

However, each of these trends also represents a challenge that needs to be overcome in order to be beneficial to MDO problems. The work in this paper intends to provide a possible solution to the proper setup of the above mentioned trends, thanks to the benefits that can be obtained by using semantic technologies applied to MDO and workflows systems.

2.2.1 Process Integration and Design Optimization

Process Integration and Design Optimization (PIDO) systems are used to integrate software tools in a multi-disciplinary design optimization problem. These systems include software to control the various software tools, allow communication between them and keep track of design variables, state variables, objective values and constraint information. PIDO tools incorporate SWFM and provide the optimization and design exploration toolboxes to perform actual design optimization.

2.3 Semantic Technologies

Semantic technology can be used to encode the meaning of data separately from the data and content files and separately from application code, thereby enabling human and machine understanding and supporting machine reasoning. Altering and adding relationships between concepts or programs can be done by simply changing the external model that the programs share, providing flexible integration of tools, processes and data. Semantic technologies add an abstraction layer that enables the interconnection of data, content, and processes. Additionally, it allows data to be shared and reused across application, enterprise, and community boundaries. Semantic technology can be split into three main components: an *ontology*, defining concepts and relations; a *reasoner*, providing inferencing capabilities; and a *graph database* (or triple store) with querying functionalities, to store instantiation of classes (i.e. the data).

An ontology is a formal representation of knowledge; it represents domain knowledge as a set of concepts. Ontologies provide a shared vocabulary that can be used to model types of objects or concepts, their properties, and the relationships between them. Using an ontology, information within a domain can be modelled in a format suitable for reasoning. Within a computer system, ontologies can also be used as a means to make information understandable for a machine. Formal MDO or optimization ontologies do not exist, however the conceptualization of the MDO domain by Sobieszczanski-Sobieski [21], the categories derived by Giesing and Barthelemy [17], the specification language for MDO by Tosserams et al. [22] and the classification by Martins and Lambe [5] can form the basis for an MDO ontology and for the modelling of MDO architectures.

Ontologies can be used to structure a knowledge base. To populate this knowledge base with actual data, individuals (instantiations of the classes) are added. Because of the relations between classes that are defined in the ontology, it is possible to derive extra information about an individual. This can be done using a semantic reasoner or reasoning engine. A reasoner is a piece of software that infers so-called logical consequences. Using reasoning, one can, for example, infer that an instance of one class is also an instance of another more specific class, if it satisfies the conditions of the latter class.

The Web-Ontology Language version 2.0, known as OWL 2.0 [23], is used to model the ontology for the MDO advisory system's knowledge base. In OWL 2.0 [23], the properties that relate classes are called *object-properties*. Other properties are available, such as the *datatype-properties*, which represent, for instance, integer values or Boolean values, and the *annotation-properties*, which can be used, for instance, to comment on classes or to describe units. OWL ontologies and OWL reasoners use the Open World Assumption (OWA). [24] This means that something that cannot be deduced

from the knowledge in the knowledge base is unknown, whereas the Closed World Assumption (CWA) implies that anything that is unknown is false (not true).
In addition to relations between classes in OWL, the Semantic Web Rule Language (SWRL) can be used to define rules, which can be modelled on top of the OWL ontology. SWRL can use OWL class expressions to model rules that cannot be modelled through restrictions in OWL. However, SWRL, being modelled in OWL ontologies, is also restricted by the same open world assumption and does not support negation as failure (i.e. that p is assumed not to hold, from failure to derive p) is not supported. Only when something is explicitly stated as *not* being an instance of a class, it can be inferred that it is *not* an instance of that class. Additionally, the SWRL rules, OWL ontology and triple store database can be used in combination with the SPARQL Protocol and RDF Query Language (SPARQL). SPARQL is a semantic query language for graph databases that can retrieve and manipulate data stored in Resource Description Framework (RDF) format. The RDF format is used to store semantic facts as triples (subject – predicate – object) inside the graph database. The OWL format for ontologies is in fact built on top of RDF.

2.4 Limitations of the current application of optimization

Since the introduction of computers into (technical) product calculations and modelling, great progress was made in the fields of computational analysis and design. Analysis algorithms can act on realistic models of product geometry and properties. Product developers have a wide range of proven off-the-shelf tools at their disposal at all levels of fidelity and for all design disciplines, rendering them useful from conceptual design up to the detailed design phase. When such a tool is coupled to an optimization algorithm, product geometry and properties analysed in that particular tool can be optimized for a given objective function, while satisfying numerical constraints. The most suitable way to structure an optimization problem and properly couple disciplines within this problem is, however, unknown and can be different from problem to problem. Commonly, certain MDO architectures or algorithms are only used because they have been successfully used in the past. The algorithms and optimization architecture are not selected to fit the problem at hand, neither is there knowledge available within the companies to determine the suitability of a certain architecture for the problem at hand.
A business may utilize tools from a different vendor having different (geometry) definitions, input and output formats. To perform a complete analysis of the product performance and characteristics, these tools often need to be run one after another in a simulation workflow. In the entire product development process (PDP), multiple simulation workflows may be present. In present-day PDPs, simulation tasks are lacking explicit context on their position in the PDP and the influence their results will have on the remainder of the process is often unknown. Simulation workflows are either not explicitly defined, or they are separately defined and documented outside the overall PDP workflow itself. The analysis tools themselves exist as separate entities within the business IT landscape and lack a formal and explicit connection with related tools and with the overall PDP.
Each tool is associated with a number of experts holding tool-specific knowledge such as favoured settings, preparation of input data and post-processing and interpretation of results. Individual tools have reached maturity at an individual level with respect to computational efficiency, fidelity and functionality but the interaction between multiple tools, covering the entire spectrum of simulating design behaviour has never been a point of focus. Computational resources are called for each tool separately, without regard for the position of that tool within the workflow and for an optimal run sequence that minimizes waiting time**.** This lack of integration between business processes and workflow management systems constitutes a significant bottleneck in the formalization of real multidisciplinary problems, since a lot of information and knowledge is kept logically separate between the two systems while there is, in reality, a clear dependency between the two. The use of logical relations and ontologies to link the two systems to each other follows the most natural way that engineers use to formalize relations. If these relations can be exploited properly, then the multidisciplinary advisor and optimizer can also deliver a more coherent design system to the user that can link the business processes (process, requirements, constraints, etc.) related to the product

design level, to the numerical tools that analyse the system and iteratively optimize it. The MDO advisory system proposed in this paper aims to assist design engineers to properly set up an MDO architecture for their specific problem at hand as well as to build, execute and control dynamic workflows in an agile and collaborative way, including human oriented activities as well as automated simulation and optimization processes.

3 DESIGN OF AN MDO ADVISORY SYSTEM

This chapter provides a general description of the MDO advisory system, highlighting the functionalities, subcomponents and the process of the system.

3.1 General description

The intrinsic complexity of MDO itself and the problems to be solved by MDO, combined with a lack of understanding of MDO, yield that MDO is not fully exploited in industry. An advisory system could support non-MDO-experts in the application of MDO technology. The MDO advisory system that is presented in this paper has three main functionalities:

1. *Advise*.
   Following a description of the problem at hand, together with additional criteria (e.g. speed, accuracy, required feasibility at every iteration, involved software tools), the advisor should provide the user with a ranked list of suitable MDO architectures (e.g. 1. IDF; 2. MDF; 3. SAND). The advisory system can provide links to relevant documentation for these architectures.

2. *Formalize*.
   The formalization of the selected architecture into a complete problem definition should be supported by the advisory system. From the formal definition of the MDO architectures in the ontology, a template can be provided that has to be filled by the user with the support of the advisor. The advisory system can provide links to relevant documentation to assist the user with this task.

3. *Integrate*.
   The formalized MDO architecture must be integrated into an executable workflow by means of a PIDO system that incorporates SWFM, containing all relevant information about licenses and external tools, input and output files and workstations to effectively solve the optimization problem.

The integration of these three functionalities is illustrated in Figure 1. On the top left, the MDO problem is specified: the objective could be, for example, to design an aircraft with minimum fuel weight for a certain range and payload. Next to that the user can specify some preference concerning, for example, speed, accuracy, required feasibility at every iteration and involved software tools.
On the basis of the specification of the MDO problem and the provided user's preferences, a reasoning mechanism will suggest the most suitable MDO architecture(s) and optimization algorithm(s) to use, among those formalized in a central knowledge base (KB) (structured by ontologies) (Figure 1, bottom left). The selection is performed on the basis of criteria stored in the KB, to discriminate between the applicability of certain architectures. Examples of these selection criteria can be, for example, the number and type of design variables or the need for intermediate results to be feasible. Once a given MDO architecture has been selected, e.g. an MDF (Multi-Disciplinary Feasible) or an IDF (Individual Discipline Feasible) architecture, the MDO advisory system will help the user to assemble the given MDO system according to the selected MDO architecture (Figure 1, bottom right).
At this point, dedicated translators could be used to translate the obtained MDO architecture (stored in the KB) into specific implementations in one (or more) commercial PIDO systems such as Optimus [25] and its associated workflow management module, where the actual optimization will eventually

take place (Figure 1 top right). Results (in terms of performance of the MDO advisor and execution of the workflow) are communicated back to the knowledge base, for re-use.

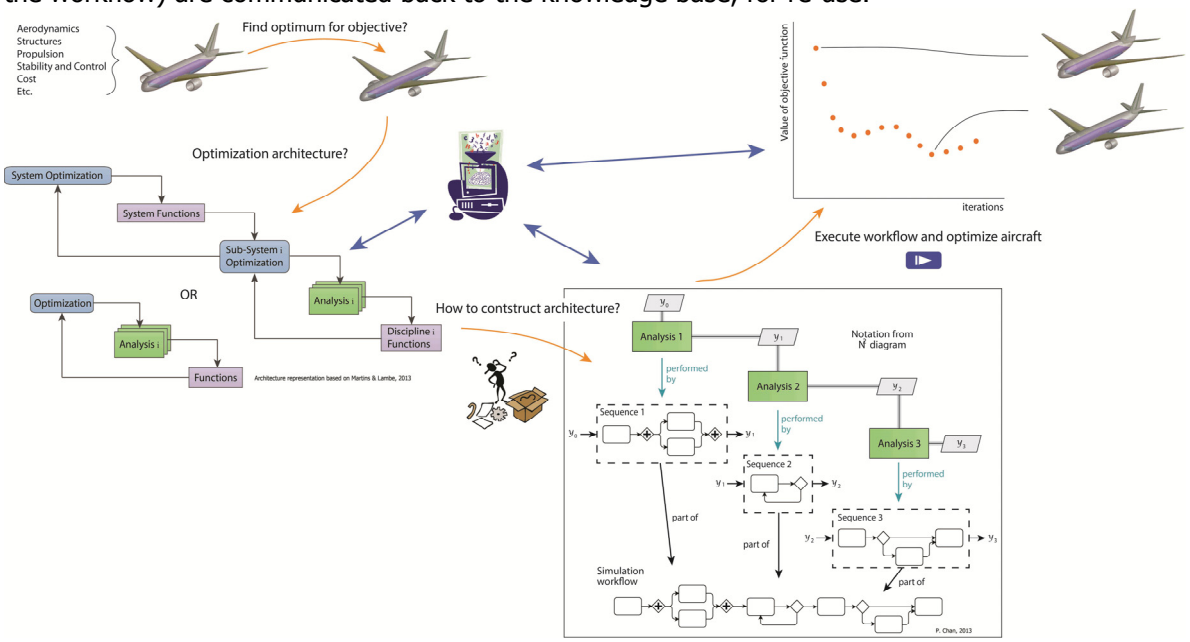

Figure 1. Graphical representation of the proposed MDO advisory system.

Re-use of information can be considered in future MDO problems and/or the same optimization problem, e.g. when reconfiguring the workflow or selecting a different architecture. The selection of a different architecture can, for example, be suggested based on a preliminary assessment of the MDO problem by Optimus. The preliminary assessment can identify ineffective constraints or design variables, or weak discipline interactions and feed this information back to the user via the MDO advisor, to suggest a different problem configuration.

3.2 Components

To provide the three functionalities (advise, formalize and integrate) several components have to be developed and combined into an MDO advisory system. These components are necessary to interface with a user, store and retrieve data in a structured manner, respond to user inputs according to certain rules, present results, and solve the actual optimization problem. These components are described in this chapter and their interactions are illustrated in Figure 2.

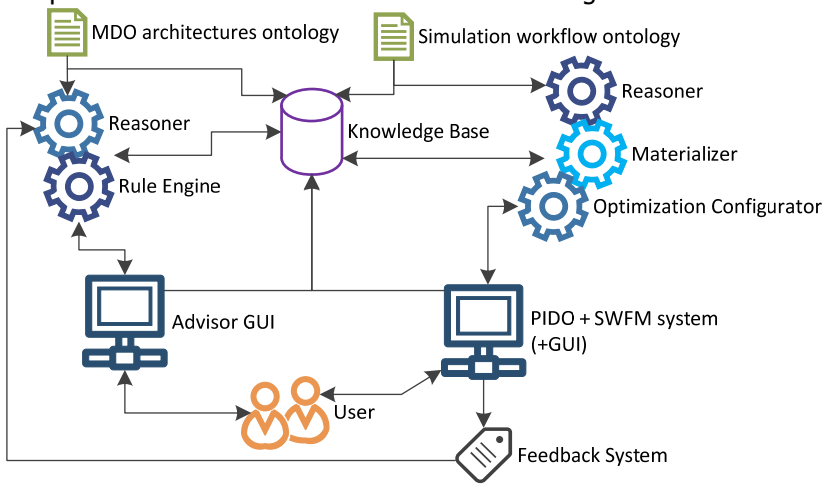

Figure 2. Graphical illustration of the advisory system's components and their interaction.

Advisor GUI
The advisor's graphical user interface is the main component that the user sees and interacts with. It consists of the GUI and the code required to integrate the different components.

Ontologies
The ontologies form the structure of the knowledge base. They are required to store and retrieve data according to a properly specified scheme. Additionally, these ontologies are required to support reasoning on input information according to rules. In the advisory system, separate ontologies are present: one for MDO architectures, and one for simulation workflows.

Knowledge Base
The knowledge base is a triple-store (graph) database that contains the actual data, structured according to the ontologies. Separate repositories are created to separate data of different projects (or components, when required). The data, in the form of triples, are the instantiations of the classes defined in the ontologies.

Reasoner and Rule Engine
The reasoner and rule engine together allow the MDO advisor to compare the problem definition to the formalized MDO architectures in the ontology and check whether there is a match, or check up to what percentage a match with a predefined architecture, formalized in the ontology, exists. Additionally, these tools allow the use of rules that represent the selection criteria between MDO architectures. These rules can be defined inside the ontology, as restrictions, using a rule language or they can be added by including database queries to manipulate data. The reasoner and rule engine form the core of the functional part of the MDO advisor.

Materializer
The automatic generation of the overall optimization work flow is used to automatically translate a high-level description of a workflow into a multi-level system of simulation work flows that can be used for optimization and are ready to be solved. The materializer uses the information obtained from the user and derived by the reasoner and rule engine to write the problem definition to a workflow in Optimus.

PIDO system with SWFM
The PIDO system is used to execute the simulation workflows generated by the materializer. It uses an optimization algorithm suggested by the advisor to solve the problem specified in the workflow. In this case Optimus (which has its own GUI) with its innovative HAROS-HD strategy is the PIDO system used to solve the workflow. The PIDO system will have to be configured to solve the optimization problem.

Optimization Configurator
The optimization configurator is used to translate a selected MDO architecture, instantiated for the problem definition at hand to the PIDO system, using the Materializer. The configurator adds the required technical details, in terms of software tools, licences, exchange formats etc., to actually configure an executable workflow. This is supported by the use "High-Level Activities" (HLA) and "High-Level Engineering Services" (HES) concepts, as presented by Chan [26].

Feedback system
Based on initial runs of the solver, the workflow can also be analysed. This can result in valuable information about the influence of design variables, constraints or disciplines on the optimization process. This information can be used to fine-tune the workflow and perhaps even provide different advice for the optimization architecture to use, e.g. to improve the convergence time or the accuracy.

3.3 Process description

The flow through the components can be visualized as a process. Different processes could be followed, depending on which components of the advisory system are used. The user may decide to skip part of the flow and only use a few components. The different processes that can be followed when using the MDO advisory system, or components of the system, are visualized in Figure 3.

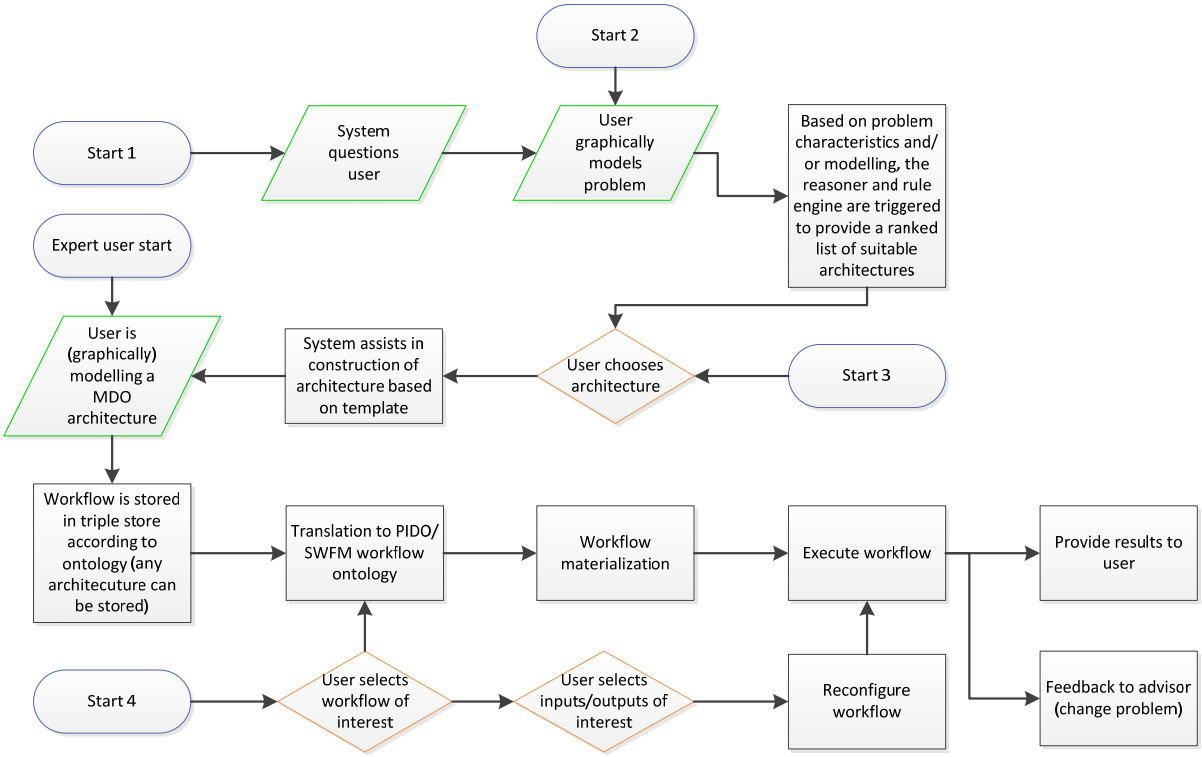

Figure 3. General overview of the MDO advisor process and the possible entry points.

In this figure, five different starting points can be identified, resulting in different flows through the advisory system. Starting point 1 is the entry point for the non-expert user that wishes to use the entire process of the advisory system. Starting point 2 is the entry point when the user wishes to use the advisory functionalities, only based on the problem he described graphically. This graphical modelling can be done in the form of design structure matrix (DSM). Starting point 3 lets the user select the desired MDO architecture (among those stored in the knowledge base) and assists in the modelling. An expert user can decide to model a different MDO architecture than those available in the knowledge base and then use only the translation, execution and feedback functionalities of the MDO advisory system. A last entry point, starting point 4, is considered for users who wish to execute a preconfigured MDO workflow, from a database or an external Optimus workflow. Alternatively, the user can decide to let the system select only the part of the workflow that is of interest, depending on the input and/or output variables of interest selected by the user. The system will reconstruct the partial workflow automatically and configure it for execution.

The entire process through the system will now be described step by step, starting from the first entry point. The system will start by questioning the user about the characteristics of the problem at hand, for example the type of software involved and the number and type of design variables. These selection criteria are automatically triggered by the rule engine to help excluding certain MDO architectures, based on their suitability. After this step, the user will be asked to model the problem graphically, with the support of the advisor. This graphical modelling can also be used to assess the representation of the problem with respect to certain MDO architectures, by comparing the structure via the reasoning engine to the MDO representations formalized in the ontology. Based on the results

of the rule engine and reasoner, the user is presented with a list of suitable MDO architectures, from the ones that are formalized in the ontology and available in the knowledge base. These suitable architectures are presented in a list, sorted on their suitability. The user can subsequently select one of these architectures to proceed to the modelling phase. In this phase, the DSM, which is constructed based on the previously modelled graphical representation, is extended to the point where sufficient information is present to compile the problem into a workflow that can be transferred to a PIDO tool for execution. The workflow modelling makes use of "High-Level Activities" (HLA) and "High-Level Engineering Services" (HES) concepts, as presented by Chan [26]. These represent standard parts of the workflow, such that only the specific details need to be added, e.g. where a tool is located, on which server, pre- and post-processors that are required, etc. These can be loaded from a KB and even entire analysis parts, such as a the aerodynamic analysis of an aircraft wing with some standard settings, can be retrieved and integrated in the workflow, without the need of going through all software intensive operations to configure this part. The workflow is then loaded from the KB into the PIDO tool (Optimus in this case), by means of the workflow materializer. This also allows the user to select only part of a preconfigured workflow, e.g. in case he is only interested in the aerodynamic analysis of a full aircraft conceptual design workflow or when he is only interested in particular outputs. In the latter case, only that part of the workflow that is relevant to this output is automatically reconstructed by the materializer. The workflow can then be executed and the results can be stored inside the KB. Additionally, a feedback connection can be made to the advisory functionalities to communicate, for example, that a certain design variable has little relevance. This means that the advisor could propose to make this variable a fixed parameter, reducing the complexity and number of design variables and possibly changing the suitability of certain MDO architectures. The advisor may then propose to select a different architecture and supports the user in the switch.

4 TECHNICAL IMPLEMENTATION

The previous sections have presented some insight into the main functionalities of the envisioned MDO advisory system and described its operation from a user perspective. The following sub-sections discuss the technical integration of the various software components of which the MDO advisory system is built. At date, the development of the advisory system is still in progress and not all the components have been implemented at the targeted level of maturity yet. The section below provides some detail of the main components, the knowledge base, MDO ontologies, rule engine and reasoner for the MDO architecture selection (these components are explained in [27]) and the optimization configurator, specifically the workflow materializer, that are now in operation. The section is concluded by an illustration of the capabilities of the workflow materializer on a small aircraft design example.

4.1 Technical architecture and enabling technologies

Figure 4 illustrates the process through the advisory system, composed of the architecture selection and configuration, and the optimization configurator. The process starts when a user makes a request for the creation of an MDO architecture, suitable for the specific MDO problem at hand. As explained in the previous section, there are different entry points a user can take. However, all will result in the advisor proposing an architecture that would work for the disciplines and criteria that the user selected. This is portrayed in Figure 3. This section describes the technologies per component that should enable the process illustrated in this figure.

One of the main technologies used in the advisory system is OWL [23], which is a semantic web technology as explained in Section 2.3. There are several standard OWL terms that can be used to provide structure to data and, next to this, custom terms can be added to build the structure that is desired. These OWL ontologies provide the structure and meaning to the knowledge base and the data that is used by the advisory system. In addition to this, they are used by the reasoning engines.

Central to the advisory system is the knowledge base. A graph database, or triple store, is selected as the database for the advisory system. The graph database that is used has built in OWL capabilities and it has many APIs, including a Java API and a Python API. These APIs are what the other parts of the advisory system use to communicate with the triple store. The architecture selection communicates through the Java API while the configurator does so through the Python API.

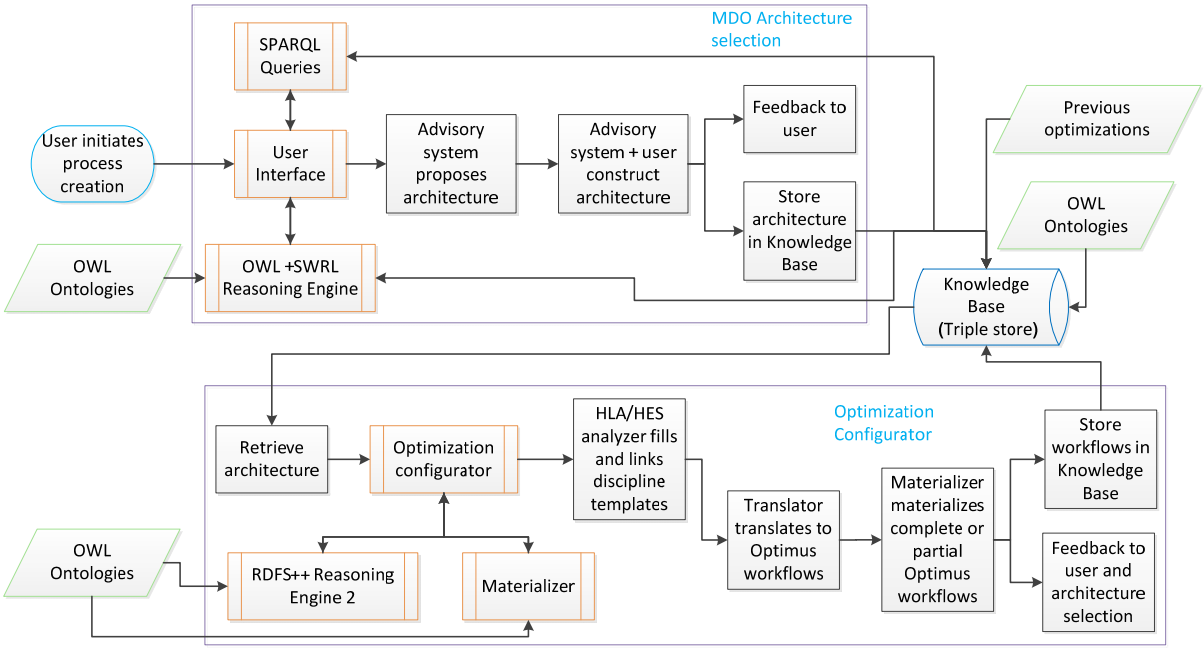

Figure 4. Overview of the technical architecture of the MDO advisory system where the architecture selection is coupled to the optimization configurator.

All information is stored inside the triple store as triples, with a subject, property and an object. To provide structure to the data, the MDO architecture and workflow OWL ontologies are added to the database. The materializer uses the workflow ontology to describe the standard structure that Optimus elements should have, this ontology is illustrated in Figure 5 with some basic examples. Figure 5a shows a part of the class hierarchy in the Optimus Ontology. Figure 5b shows an example of a property, *hasInputVar*, which has an inverse property, *isInputVarOf*. It has domain and range *InputArray* and *InputVar*, which means that only input arrays can contain this property and only input variables can be linked using this property. The inverse functional relation means that the inverse property (*isInputVarOf*) can only be used once by an element. Figure 5c shows a simple example of restrictions used to define what structure an element of a certain class must have. A simple restriction is added to *TextAnnotation*, stating that next to being a *WorkflowElement*, it has to have the property *hasText* added at least once. Also, subclasses inherit from their superclasses.
The advisor is written in Java, because of the availability of semantic technology for this language, such as reasoners, rule engines and other pieces of software related to OWL. An open source OWL reasoner is used, with some additions added to this code to achieve the desired reasoning functionalities, as is described in [27]. The open source reasoning engine can also use the Semantic Web Rule Language (SWRL) [28], allowing modelling rules inside an ontology.
The optimization configurator is completely written in Python to enable easy integration with the Optimus Python API. Part of the configurator is a workflow materializer, which extracts information from the triple store, takes all of the information extracted from the triple store and creates the Optimus workflow and stores the results back in the triple store. It can retrieve a partial (sequence of) workflow(s), in case the user is interested only in the flow from a certain input, the result of a certain output or both. This will be illustrated with an example in Section 4.2.

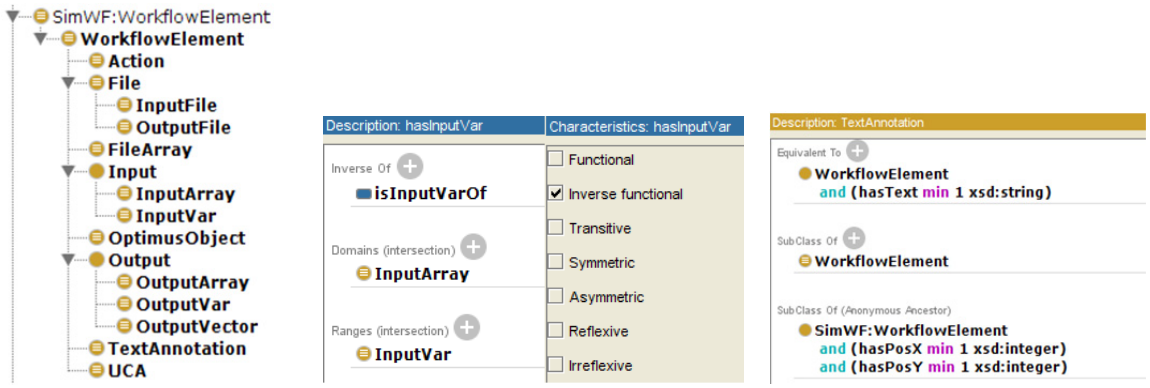

Figure 5. Workflow ontology class hierarchy (a), object property (b) and restrictions (c)

When the user presents these inputs and/or outputs, the script first searches through all available workflows to find the correct inputs and outputs. If the name of inputs and outputs are found multiple times, the script prioritizes found variables in higher level workflows. Then, a search is done to find all variables in between the found variables. The main goal of this partial recreation is to avoid the execution of tools that are not needed for the desired computation. However, if an external tool needs three inputs but only one is recreated, the action will not be able to execute. Thus, a dependency check is done, to make sure that all elements have the elements they need to be able to execute the workflow. The materializer uses a reasoning engine to infer new information represented by the properties provided and the OWL terms connected to it. When a workflow is stored, initially all direct connections present in the workflow are added to the database. Arrays often have multiple incoming and outgoing connections. However, not all of the variables within the arrays follow all connections present. Thus, for each variable within an array a search is performed through all available connections to see which variable has an active data connection with which elements. This connection is stored as a triple, which will be used by the script for partial materialization to ensure that also from input arrays only parts of the workflow are created that are relevant to the search provided by the user.

4.2 Workflow materialization: Aircraft design use case

The use case focusses on the optimization configurator, more specifically on the workflow materialization. The materialization of partial workflows is extremely useful in the reconfiguration of an optimization problem, either based on user specified criteria (e.g. the wish to reconfigure a single level MDO architecture into a multi-level architecture), or based on feedback from the PIDO system. The feedback could indicate that a particular variable has only a limited influence on the optimization (based on initial runs), therefore it could be fixed instead of remaining a design variable. Fixing this particular design variable can have a significant influence on the workflow and the materializer can be used to quickly retrieve only that part of the original optimization problem that is dependent on the remaining design variables. To illustrate these capabilities for a multi-level workflow, a use case is presented that involves two aerodynamic calculations. A vortex lattice calculation using AVL [29] is used to compute the lift (C_L) and the induced drag (C_{Di}) from the Mach number (M) and angle of attack (alpha). A second program is used to compute the drag coefficient at zero angle of attack (C_{D0}) from the Mach number (M) and altitude (h). The original workflows are illustrated in Figure 6a. The analysis is done for a constant geometry and airfoil, stored in a separate file.
Figure 6a shows three workflows that together compose a multi-level workflow. The top level workflow, *Aerodynamics*, is composed of two lower level workflows, *AVL* and *CD_0*, respectively. The *Aerodynamics* workflow contains all inputs needed by both lower level workflows (angle of attack, Mach number and altitude), where the actual calculations are performed. The outputs (C_L, C_{Di} and C_{D0}) are gathered in an output array. The *AVL* workflow uses only the angle of attack and Mach

number in combination with some input settings and airfoil data to run AVL and calculate the C_L and C_{Di}. These values, together with the angle of attack used, the reference wing span and the reference wing surface area are extracted and passed to the top level workflow. Additionally, the aspect ratio is calculated. The *CD_0* workflow uses the Mach number and altitude (converted from meters to feet, puts these in an input file and, using a batch file, and runs the C_{D0} analysis. C_{D0} is extracted and passed to the top level workflow.
All workflows are stored in the knowledge base, according to the ontology. Because of the links between the different levels of the workflows, only storing the top level workflow automatically triggers an action to store all connected workflows. The link between the workflows is also represented inside the ontology.
Consider the case when the user is only interested in a specific output, in this case C_L, influenced by the input variable Mach number. (For example, when it was determined from a preliminary analysis that this variable has a large influence on the result.) This requires only partial retrieval of the multi-level workflow, from inspection of the workflows it can easily be concluded that C_L is not influence by the CD_0 workflow. In this case the materializer can be used to automatically retrieve only the necessary information from the knowledge base and reconstruct the partial workflows as is illustrated in Figure 6b.
The materializer first tries to identify Mach and C_L combinations in the same workflow, in this case finding it in the *Aerodynamics* and *AVL* workflows. In a direct search between these variables it finds 8 elements with 6 connections. Finally, the dependencies are searched and 11 elements with 9 connections are found. The result of the materialized workflows is shown in Figure 6b.

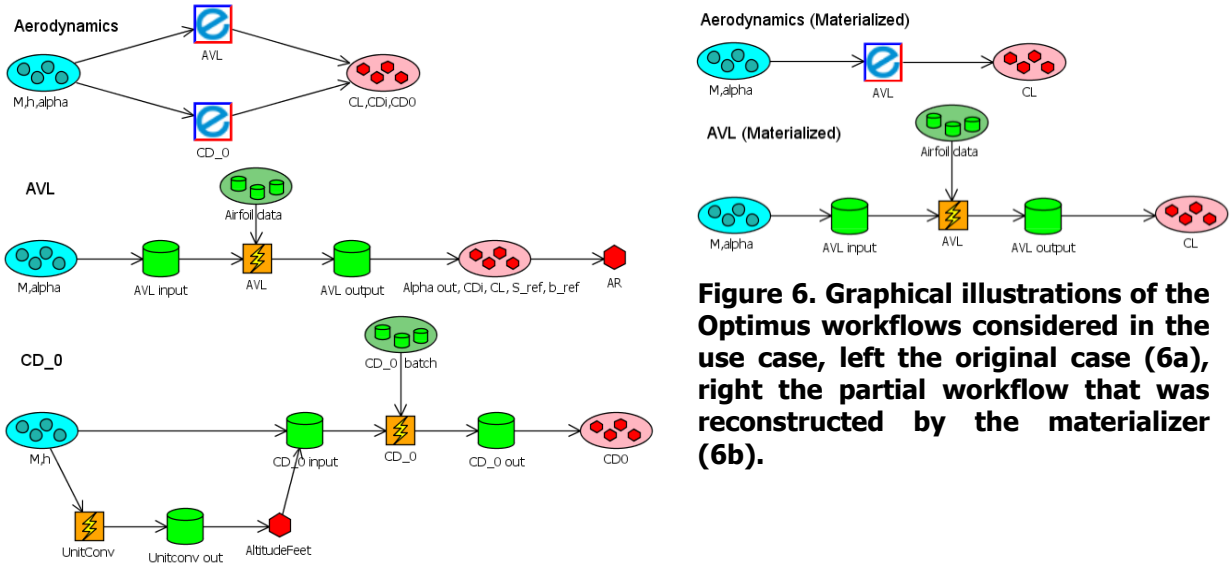

Figure 6. Graphical illustrations of the Optimus workflows considered in the use case, left the original case (6a), right the partial workflow that was reconstructed by the materializer (6b).

As can be seen, the materialized *Aerodynamics* workflow does not include the link to the *CD_0* workflow anymore, since this workflow was not needed to compute the lift coefficient. Therefore, the latter workflow is also not materialized. For the *AVL* workflow, all elements are materialized, because the elements are directly connected from the input to the output because a dependency exists. While 11 elements were found, Figure 6b only shows 9 elements, because the Mach number and altitude are present in an input array that is used twice, similarly for the output array for C_L. The connection to the aspect ratio calculation is not made, because it is not a dependency of the lift coefficient or Mach number.
The approach described for the materializer will, once integrated with the optimization configurator in the advisory system, allow for the quick generation of partial workflows, after modifications have been suggested by the feedback system or desired by the user. Moreover, the materializer also allows for the reuse of information from previous optimizations.

4.3 Impact

The impact of the MDO advisory system presented in this paper is dependent on the type of user, and on the application area. The primary impact is for non-expert users, who will not have the expertise required to select a suitable architecture, so are very likely to choose the wrong one for their problem. Next to this, the large complexity of different MDO architectures can often scare non-experts away from optimization all together, leaving an area of design improvement largely untouched. The MDO advisory system lowers the complexity of implementing different MDO architectures, thus lowers the threshold for using different MDO architectures and for using optimization in general. In addition, suggesting the most suitable architecture for an optimization problem and helping the user to properly implement this architecture according to a formal definition and integrate it inside a PIDO system will save time. The optimization configurator can save time through the partial workflow creation, since only the part of the workflow that is relevant to the calculation the user wants to perform is created. This way, computations that are not relevant for the current calculation are not performed, preventing wasted computation time. Furthermore, a lot of trivial, repetitive, and non-creative tasks in the software intensive operations required for the set-up of the optimization problem and architecture, as well as the creation and use of workflows are automated. This reduces drastically the probability to repeat mistakes and increases the capability to accumulate and reuse design knowledge. Moreover, the use of OWL ontologies and reasoners allows a structured storage and the inference of extra information, about the problem and workflows. These technologies allow re-use of information in the future, for similar problems or e.g. for problems where, an entire discipline (like aerodynamics) can be reused.

5 CONCLUSIONS AND FUTURE DEVELOPMENT

In this paper the design and current research state for a MDO advisory system, supported by knowledge based technologies, was presented to support non-experts in the application of MDO. The system is able to advise on the most suitable architecture for the problem at hand, formalize the problem according to this architecture and support its integration in a PIDO tool as an executable workflow. The knowledge based technologies allow the storage of problems in a knowledge base according to a formal definition inside an ontology, moreover this ontology allows for reasoning. Reasoning can be used to infer new information. Part of the integration of MDO problems inside a PIDO tool is the configuration of the workflow. A use case was presented to illustrate this functionality, focusing on the recreation of a workflow and a partial workflow. The software proved to be capable of selecting only the required part of a workflow, depending on the output values that the user was interested in. The selection of this partial workflow and its recreation is supported by the ontological approach that allows determining the relations between different parts of the workflow without them being exactly specified. Future work will focus on the integration of all aspects of the MDO advisor inside a demonstrator and testing on actual MDO problems for all of the functionalities that were mentioned.

ACKNOWLEDGEMENTS

This research is partly funded by the ITEA-2 call 8 project IDEALISM of the European Union (see https://itea3.org/project/idealism.html for details), in form of the national grant number 13040 of the ministry of science and education in the Netherlands, and by the grant IWT (Agentschap voor Innovatie door Wetenschap en Technologie in Flanders), national grant number 140235.

REFERENCES

[1] J. Sobieszczanski-Sobieski, "Multidisciplinary Optimization for Engineering Systems: Achievements and Potential," 1989.
[2] AIAA Technical Committee on Multidisciplinary Design Optimization, "White Paper on Current State of the

Art," 1991.
[3] J. Sobieszczanski-Sobieski, "Multidisciplinary Design Optimization: an Emerging New Engineering Discipline," 1993.
[4] N. Alexandrov and S. Kodiyalam, "Initial results of an MDO method evaluation study," in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998.
[5] J. R. R. A. Martins and A. B. Lambe, "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal,* vol. 51, no. 9, pp. 2049-2075, July 2013.
[6] N. M. Alexandrov and R. M. Lewis, "Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design," *AIAA Journal,* vol. 40, no. 2, pp. 301-309, February 2002.
[7] N. Alexandrov and R. Lewis, "Analytical and computational properties of distributed approaches to MDO," in *Proceedings of the 8th Symposium on Multidisciplinary Analysis and Optimization*, 2000.
[8] N. F. Brown and J. R. Olds, "Evaluation of Multidisciplinary Optimization Techniques Applied to a Reusable Launch Vehicle," *Journal of Spacecraft and Rockets,* vol. 43, no. 6, pp. 1289-1300, November 2006.
[9] E. Cramer, J. Dennis Jr., P. Frank, R. Lewis and G. Shubin, "Problem Formulation for Multidisciplinary Optimization," *SIAM Journal on Optimization,* vol. 4, no. 4, pp. 754-776, November 1994.
[10] J. E. Dennis Jr., S. F. Arroyo, E. J. Cramer and P. D. Frank, "Problem formulations for systems of systems," in *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, 2005.
[11] Z. Lu and J. Martins, "Graph Partitioning-Based Coordination Methods for Large-Scale Multidisciplinary Design Optimization Problems," in *Proceedings of the 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.*, 2012.
[12] B. Roth and I. Kroo, "Enhanced Collaborative Optimization: Application to an Analytic Test Problem and Aircraft Design," in *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2008.
[13] A. de Wit and F. van Keulen, "Overview of Methods for Multi-Level and/or Multi-Disciplinary Optimization," in *Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2010.
[14] S. Kodiyalam and J. Sobieszczanski-Sobieski, "Multidisciplinary design optimization - some formal methods, framework requirements, and application to vehicle design," *International Journal of Vehicle Design,* vol. 25, no. 1/2 (Special Issue), pp. 3-22, 2001.
[15] T. Long, "The Optimization Assistant - Helping Engineers Explore Designs Through Collaboration," in *Proceedings of the 4th International Conference on Intelligent User Interfaces*, New York, NY, USA, 1999.
[16] J. Agte, O. de Weck, J. Sobieszczanski-Sobieski, P. Arendsen, A. Morris and M. Spieck, "MDO: assessment and direction for advancement -- an opinion of one international group," *Structural and Multidisciplinary Optimization,* vol. 40, no. 1-6, pp. 17-33, 2010.
[17] J. Giesing and J.-F. Barthelemy, "A summary of industry MDO applications and needs," in *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998.
[18] G. Belie, "Non-Technical Barriers to Multidisciplinary Optimization in the Aerospace Industry," in *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.
[19] N. Russell, A. H. M. ter Hostede, W. M. P. van der Aalst and N. Mulyar, "Workflow Control-Flow Patterns : A Revised View," BPM Center Report BPM-06-22, BPMcenter.org, 2006.
[20] A. H. M. ter Hofstede and W. M. P. van der Aalst, "YAWL: Yet Another Workflow Language," *Information Systems,* vol. 4, no. 30, pp. 245-275, 2005.
[21] J. Sobieszczanski-Sobieski, "{Multidisciplinary Design Optimization: an Emerging New Engineering Discipline}," 1993.
[22] S. Tosserams, A. Hofkamp, L. F. P. Etman and J. E. Rooda, "A specification language for problem partitioning in decomposition-based design optimization," *Structural and Multidisciplinary Optimization,* vol. 42, no. 5, pp. 707-723, 2010.
[23] Anonymous, The World Wide Web Consortium (W3C), *OWL 2 Web Ontology Language.*
[24] N. Drummond and R. Shearer, *The Open World Assumption,* 2006.
[25] Noesis Solutions, "Optimus v10.16," June 2015. [Online]. Available: http://www.noesissolutions.com/. [Accessed 12 June 2015].
[26] P. K. M. Chan, "A New Methodology for the Development of Simulation Workflows: Moving Beyond MOKA," 2013.
[27] M. F. M. Hoogreef and G. La Rocca, "An MDO advisory system supported by knowledge-based technologies," in *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Dallas, Texas, 2015.
[28] Anonymous, The World Wide Web Consortium (W3C), *SWRL Semantic Web Rule Language.*
[29] M. Drela and H. Youngren, "AVL," [Online]. Available: http://web.mit.edu/drela/Public/web/avl/. [Accessed 14 June 2015].