Deutscher Luft- und Raumfahrtkongress 2012
DocumentID: 281263

MODELLING APPROACH WITH VERSATILE FILTERING CAPABILITIES

S. Ziemer

Bauhaus Luftfahrt e.V., Lyonel-Feininger-Str. 28, 80807 München, Deutschland

Abstract

The data elements created throughout a conceptual design project reflect the wide diversity of stakeholders that are involved in an aircraft design project. Data elements are contributed and accessed from different engineering disciplines, software tools, and the workflow process defined for a project. For the different users involved in such a design project, it is of high importance to have efficient access to the data relevant to their work independent of this diversity. This paper advocates an integrated data management approach and proposes a filtering approach to support the users in having efficient access to their design data by taking advantage of the overall design knowledge represented in such a large data set. This approach is described in the context of OpenCDT, a framework for integrated data management for conceptual aircraft designs developed at Bauhaus Luftfahrt.

1 INTRODUCTION AND MOTIVATION

The data considered during an aircraft design project is diverse and reflects the different stakeholders involved in aircraft design.An aircraft design project sees the cooperation of several discipline experts, the use of several software tools, and a project management to coordinated the joint effort. The resulting data set from an aircraft design project contains both a parameterized aircraft model, information regarding the software tools used in the project and process data that describe the process deriving the aircraft model. The overall data set from an aircraft design project can, therefore, become quite large and be of some substantial complexity. While such an data set reflects all the mentioned aspects of an aircraft design project, a single user is, usually, interested in a *subset* of the overall data set, containing only those data elements relevant for his tasks.

Having access to the *"right"* subset from a data set will yield a more efficient design process and a higher quality of an aircraft design model. An aircraft engineer is interested in having the flexibility to change the subset of an aircraft model he is working on in accordance with his task or with the progress of the project.

A common solution for providing users with subsets of the overall data set is to organize the data in separate data sets. Typically, each separate data set addresses a single discipline or is managed by a single tool. This solution results in comprehensible data sets, but does not offer the flexibility to efficiently adjust the content of a data set. Also, this solution is hampered with the additional burden of keeping all separated data sets consistent.

Other solutions addresses the challenge of preserving the consistency of a data set by providing a single framework that addresses several disciplines [1] or by supporting the efficient exchange of data between different separate subsets in order to ensure the consistency among them [2]. Both approaches have their merits, but will most likely have limitations with respect to providing the user with the flexibility of integrating software tools for special tasks and of defining and adopting data sets during an aircraft design project.

A different approach is the use of an integrated data set, where all data elements are stored, This approach yields a more complete and accurate description of an aircraft design, and does facilitates the application of multi-disciplinary analysis and optimization (MDAO) processes. This approach enables also the use of filtering as a flexible and efficient solution of defining subsets of the integrated data set.

This paper advocates the use of an integrated data management approach and addresses the issue of a flexible filtering mechanism that provides users with the means to efficiently define subsets of an integrated data set. In addition, a prototypical implementation of a filtering mechanism in OpenCDT is presented. OpenCDT is a framework for building an integrated data set developed at Bauhaus Luftfahrt. The remainder of this paper is organized as follows: In Section 2 a brief overview of two data modeling approaches – segregated data management and integrated data management – is given, followed by the elicitation of some requirements that are required to adapt the integrated data management approach more easily. This is followed by presenting some related work on filtering mechanisms in Section 3. A solution for a filtering mechanism using the OpenCDT software developed at Bauhaus Luftfahrt is then introduced in Section 4 together with a brief example in Section 5. Some direction of future work are pointed out in Section 6. Finally, conclusions are presented in Section 7.

2 MODELING APPROACH AND RESULTING REQUIREMENTS

Any software implementation of a conceptual design process faces a number of challenges, including issues such as performance, data management, concurrency, etc. [3] An important issue is for example the fragmentation of knowledge and design data [4]. A core issue for discussing data management approaches is whether the resulting data set from an aircraft design process is integrated or segregated. In an integrated approach to data management all design disciplines, sub processes, tasks and software tools are using a single *integrated* data set. In the opposite approach, each design discipline, sub process, task and software tool uses their own separate data sets; in this approach the entire aircraft design is kept in a set of *segregated* data sets [5]. An integrated data set reflects the logical structure of a data set; the physical structure of an integrated data set can be realized as a stand-alone database or as a distributed database.

A segregated data management approach can avoid some of the complexity involved with an integrated data set, but introduces some serious drawbacks to design projects. First, in a segregated data management approach many couplings and dependencies between parameters manipulated by different sub processes, tasks or software tools are not represented explicitly and need to be handled implicitly through expert knowledge of engineers. Handling such implicit dependencies does result in additional tasks. Second, in a segregated approach to data management, different data sets will evolve more individually, thus introducing different levels of maturation, precision and fidelity among the involved data sets. While this may be a logical consequence given the different requirements to each sub process, different levels of maturation and fidelity can become serious stumble blocks to the coordination and consistency of the overall aircraft data set. In case the segregation of data sets is also driven by the use of separate software tools, the issue of data format compatibility becomes an issue. Finally, keeping a segregated data set consistent is a potential hard operation that has to solve conflicts that may exist between the individual data sets, the bridging of the different levels of fidelity and the detection of implicit (i.e. not explicitly represented) dependencies.

Using an approach with an *integrated* data set has several important benefits: First, it is a more accurate representation of an aircraft design since it can explicitly model all couplings and dependencies that exist between the parameters used to describe an aircraft. These couplings include both physical dependencies as well as dependencies introduced by calculations using parameters as input to determine other parameters. Second, an integrated data set avoids the use of redundant data elements and helps, therefor, to preserve the consistency throughout the data set. Third, getting all sub processes and software tools to use the same data set implies that a common understanding of how each attribute of an aircraft is to be parameterized has to be reached. Using this approach does, however, also come with some challenges: An integrated data set containing all information about an aircraft design can become quite large and complex, making it hard to read and comprehend all information. Keeping the consistency of an integrated data set requires the processing of all couplings and dependencies between parameters, introducing performance issues. When updating a single parameter in an integrated data set, coordination and conflict resolution with other sub processes may be required. These challenges need to be addressed in order to facilitate the effective application of an integrated data management approach. The analysis of both approaches to data management focused on the consistency, accurateness and comprehensive understanding of the involved data sets. From this point of view, the integrated data set is the preferred option. The benefits of using an integrated data management approach outweigh the described challenges that come with this approach for the intended application in preliminary design. Still, there is the opportunity to improve the tool support for this approach with new functionalities. This paper will elicit some requirements for better support of an integrated data management approach, and discuss a solution to one of these requirements in the following sections.

A new issue introduced with integrated data management is the one of decomposition. While each data set in a segregated data management approach can have its own decomposition, an integrated data set needs a single decomposition. A rather heuristic approach would be to follow the physical breakdown of an aircraft into its component (airframe, propulsion system, cabin, etc.). But other ways of decomposing an integrated data set exists, such as to rely on formal relationships of parameters in an MDAO environment [6].

To make the work with an integrated data set as efficient as possible, the following requirements to support a workflow for a design project are elicited:

- **Consistency check** – The user shall be able to define and run consistency checks on the integrated data set. This includes the declaration of triggers (i.e. code that is performed in response to an event such as the update of a parameter) to check that the dependencies between parameters in the data set are kept consistent, and user defined operations to check quality aspects of the data set.

- **Collaboration support** – Provide support for the concurrent work of several teams, using the same integrated data set. This includes support for conflict resolution when two or more users attempt to manipulate the same parameter, or long term design explorations to allow the performance of design space explorations in a private data space that can be committed once the operation is completed.
- **Filtering of data sets** – Allow the filtering of the integrated data set into user defined subsets of the overall data set. Each subset focuses on data elements that are relevant for a given sub process, task or software tool. In case a user right management is implemented, it must be possible to set the user rights for a parameter for a given subset (i.e. a parameter is read only in a subset A, and read/write in a subset B).
- **Personalized views of filtered data sets** – Users shall be able to personalize the view on filtered data sets by being able to introduce different decomposition structures for the parameters in a given filtered data set or to use an alternative nomenclature that is more “familiar” to the user.

The elicited requirements presented here describe high level requirements that need to be refined further. In the remainder of this paper, we will focus on the requirement “Filtering of data sets”, and propose a solution for filtering integrated data sets using the OpenCDT framework developed at Bauhaus Luftfahrt.

3 RELATED WORK

Techniques to reduce the size of a data space or to simplify the structure of a data space are in use by most software systems that handle larger amounts of data. Many users of software systems are familiar with some kind of filtering functionality: In information retrieval systems, filtering algorithms are used to reduce the size of a text [7]; email systems use spam filters “to identify spam for the purpose of preventing its delivery” [8]; and collaborative filtering techniques are applied when preprocessing the number of hits in web searches [9]. The Oxford Dictionary defines a filter in computing as “a piece of software that processes data before passing it to another application . . . ” [10].

In the context of this paper, a filter is used as a technique to process a data structure in order to create a new data structure that contains only those elements of the original data structure that satisfy a defined selection criterion.

An important part of a filter is the definition of a criterion of what is to be accepted and what is to be rejected by the filter. There exist several basic approaches for how to implement such a criterion. Common approaches are content-based filters and semantic filters. In content-based filters, the actual value of an attribute is compared against the search criterion, using some kind of comparison operator such as equality, inequality, or is-contained-in. Advanced forms of textual filters use pattern recognition techniques as their criterion [11], and are applied on large texts. Many are familiar with the more simpler forms of content-based filters from, for example, statements in query languages, such as the SELECT statements in SQL [12]. A semantic filter takes advantage of semantic information that has been modeled into a data set. Semantic modeling has been applied to database systems since the earlies 1970s, and focus on the explicit representation of objects, attributes and relationships among objects [13]. Several approaches for representing the semantic of a model exists. One approach is the use of tags for semantic annotation of a data model. Tags are identified by their object identity and represent thus their meaning not by the content of their attributes but by an inherent quality; tags can thus be used as primitive for a query language [14]. Similar to this approach is the use of tags for the semantic web [15]. Other approaches use the available modeling constructs of a modeling language or a metamodel.

A related field in the context of this paper is Knowledge Based Engineering (KBE). The aim of KBE is to improve the efficiency of a design process by extending a data model with design and process knowledge [16]. Extending the data model with additional knowledge is similar to the semantic annotation of a data model. The knowledge that is modeled into a data model should thus be available both for a knowledge based engineering methodology and the construction of filters.

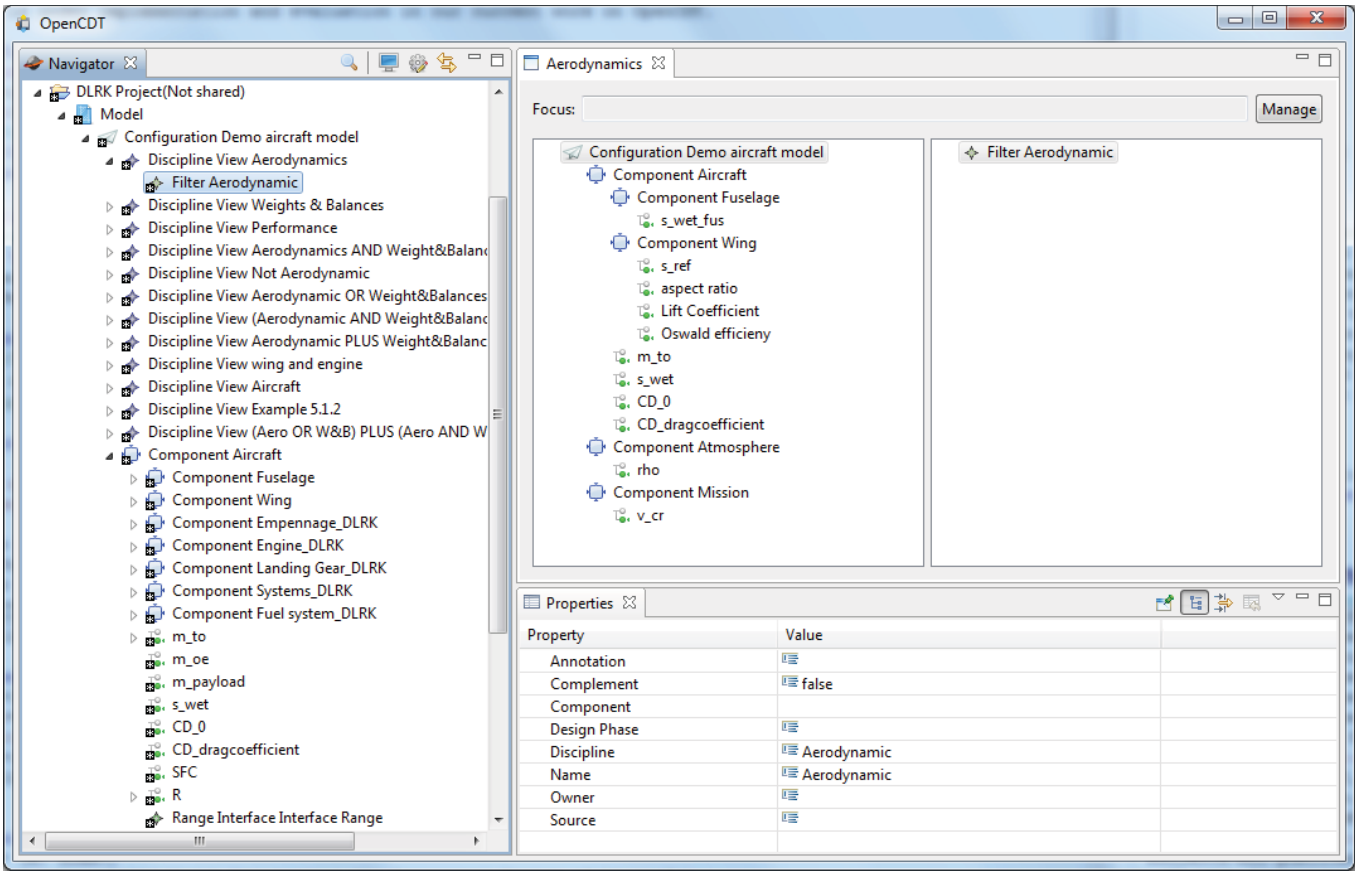
4 PROPOSED SOLUTION: FILTERING IN OpenCDT

Among the requirements for supporting the use of an integrated data set identified in Section 2 is filtering. This section proposes a filtering mechanism that allows the flexible combination of filters and that is under implementation and evaluation in our current work on OpenCDT.

4.1 OpenCDT

OpenCDT [17, 18] is an open source framework for conceptual aircraft design under development at Bauhaus Luftfahrt. The focus of OpenCDT is on providing support for the integration of aircraft design data from different disciplines and specialist software. This is clearly stated in our vision statement:

> *The vision of OpenCDT is to provide a framework for conceptual aircraft design that enables the* integration *of design data and functionality from existing software tools, that supports the* collaboration *between discipline teams and that is* flexible *enough to be adapted for designing unconventional aircraft concepts.*



OpenCDT

Navigator

- DLRK Project(Not shared)
  - Model
    - Configuration Demo aircraft model
      - Discipline View Aerodynamics
        - Filter Aerodynamic
      - Discipline View Weights & Balances
      - Discipline View Performance
      - Discipline View Aerodynamics AND Weight&Balanc
      - Discipline View Not Aerodynamic
      - Discipline View Aerodynamic OR Weight&Balances
      - Discipline View (Aerodynamic AND Weight&Balanc
      - Discipline View Aerodynamic PLUS Weight&Balanc
      - Discipline View wing and engine
      - Discipline View Aircraft
      - Discipline View Example 5.1.2
      - Discipline View (Aero OR W&B) PLUS (Aero AND W
      - Component Aircraft
        - Component Fuselage
        - Component Wing
        - Component Empennage_DLRK
        - Component Engine_DLRK
        - Component Landing Gear_DLRK
        - Component Systems_DLRK
        - Component Fuel system_DLRK
        - m_to
        - m_oe
        - m_payload
        - s_wet
        - CD_0
        - CD_dragcoefficient
        - SFC
        - R
        - Range Interface Interface Range

Aerodynamics

Focus: Manage

- Configuration Demo aircraft model
  - Component Aircraft
    - Component Fuselage
      - s_wet_fus
    - Component Wing
      - s_ref
      - aspect ratio
      - Lift Coefficient
      - Oswald efficieny
    - m_to
    - s_wet
    - CD_0
    - CD_dragcoefficient
  - Component Atmosphere
    - rho
  - Component Mission
    - v_cr

Filter Aerodynamic

Properties

Property	Value
Annotation	
Complement	false
Component	
Design Phase	
Discipline	Aerodynamic
Name	Aerodynamic
Owner	
Source	

Figure 1: Example of an aerodynamic view in OpenCDT

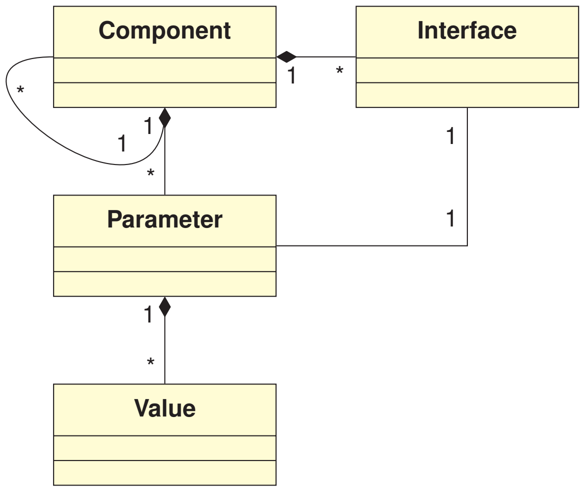


Figure 2: UML diagram of a part from OpenCDT's metamodel

The core functionality of OpenCDT is the generic support for building parameterized aircraft models. Users shall have the freedom to build aircraft models according to their needs. This includes the way an aircraft model is decomposed and the nomenclature used for the aircraft model. Aircraft models can also be build by importing data sets from external tools. The meta model for OpenCDT's own data model consists of the following constructs. An UML diagram is shown in Figure 2; note that the pairs of (1..*) shown along the arcs connecting the classes does express the cardinality of the model.

- **Component** – Components describe entities that can be parameterized such as systems or physical components. Components can be further decomposed by subcomponents.
- **Parameter** – Parameters are used to describe the characteristics of a component. A parameter can be quantified; in addition the user can qualify parameters further by specifying a quantity for the parameter, assigning it to one or more design disciplines and design phase, as well as by additional annotations.
- **Interface** – An Interface gives *global* access to a component and makes it possible to access a component when the scope of work comprises several components. An interface is linked to a single parameter and can be qualified in the same way as a parameter.

In OpenCDT, different types (or *classes*) of interfaces are defined, carrying some additional information. These interface classes include mass interface, wetted area interface and reference area interface. They correspond partly to a parameters quantity, but can be used to make even more coarse grained distinctions between interfaces, as shown here with two different interfaces for area.

- **Value** – Values are used to quantify parameters. A value has a unit that must correspond to the parameters quantity. A parameter can have several values, which makes it possible to have values representing different states of a parameter, including values specifying a requirement for that parameter or a constraint.

As a framework for integrated data management, OpenCDT will typically store data sets from different disciplines and from different software tools. To aid the work of discipline experts, OpenCDT provides filters to hide data elements that are not of interest to them.

4.2 Discipline views and filtering in OpenCDT

OpenCDT uses an approach to filtering that combines both semantic filtering and content based filtering. Semantic filtering is used for making references to elements in the aircraft model, such as references to components and interfaces. Content based filtering is used for matching the textual content of defined fields in the data model. These field are called Labels. The filtering mechanism in OpenCDT is facilitated by the following two constructs:

- **Filter** – A filter defines the search criteria applied when filtering the integrated data set. The current implementation of a filter in OpenCDT provide the following possible criteria:
  1. Interfaces – None, one or more interface classes can be specified for a filter
  2. Component – None, one or more components can be specified for a filter
  3. Disciplines – None, one or more strings containing the name of a discipline can be specified for a filter
  4. Source – None or one string containing the name of a source can be specified for a filter
  5. Annotation – None or one string containing an annotation can be specified for a filter
  6. Owner – None or one string containing the name of a owner can be specified for a filter
  7. Design Phase – None or one string containing the name of a design phase can be specified for a filter

  Only data elements that satisfy all search criteria of a filter will be selected for the resulting subset of the integrated data set. For example, when an interface class is specified in a filter, only parameters that are accessible by this interface class will be included in the result set from a filter.

- **Discipline View** – A discipline view in OpenCDT is a subset of the integrated data set. It is defined by a collection of filters and will display only those data elements that satisfy the search criteria defined in the filters. In the current implementation, a discipline view in OpenCDT preserves the default hierarchy of the integrated data set. An example of a discipline view is shown in Figure 1 containing only a single filter `Aerodynamic`, which specifies only a single discipline (as shown in the properties view).

  It is possible to combine filters with **set operators**. To build a discipline view that combines two filters, a **set expression** must be defined. A set expression defines the set operator to be used and specifies the two filters that the operator is used upon. OpenCDT supports a single unary set operator and four binary set operators:

  – The operator **OR** takes two subsets of elements from the aircraft model, each defined by a filter, and returns the **union** of them

  – The operator **AND** takes two subsets of elements from the aircraft model, each defined by a filter, and returns the **intersection** of them

  – The operator **MINUS** takes two subsets of elements from the aircraft model, each defined by a filter, and returns the **relative complement** of them (i.e. the elements that are contained in the first subset but not in the second subset)

  – The operator **PLUS** takes two subsets of elements from the aircraft model, each defined by a filter, and returns the **Cartesian product** of them (i.e. the elements that are contained in either for the two subsets, but not in the intersection of both subsets)

  – In addition the unary operator **NOT** is implemented. The operator is defined as an attribute of a filter and returns the **complement** of the subset of elements from the aircraft model that is defined by the filter (i.e. all elements from the aircraft model that are not in the resulting subset from a filter)

An example of a discipline view combining several filters is shown in Figure 5, Part B.

5 EXAMPLE

This section presents an example of a workflow for building a discipline view and filters. The example uses a

Parameter	Description	Component	Aerodynamics	Weights & Balance	Performance
AR	Wing aspect ratio	Wing	●		●
C_D	Drag coefficient	Aircraft	●	●	
C_L	Lift coefficient	Aircraft	●	●	
e	Oswald efficiency		●		●
m_{TO}	Take-off mass (initial)	Aircraft	●	●	
$m_{Propulsion}$	Propulsion mass	Propulsion		●	
m_{Wing}	Wing mass	Wing		●	
$m_{Fuselage}$	Fuselage mass	Fuselage		●	
m_{OE}	Operating empty mass	Aircraft		●	
$m_{Payload}$	Payload mass	Aircraft		●	
SFC	Specific fuel consumption	Aircraft			●
$S_{Wet_{fus}}$	Wetted area fuselage	Fuselage	●		
S_{ref}	Wing reference area	Wing	●		
S_{wet}	Aircraft wetted area	Aircraft	●		
v_{cs}	Cruise speed	Mission	●		●
ρ	Air density	Atmosphere	●		

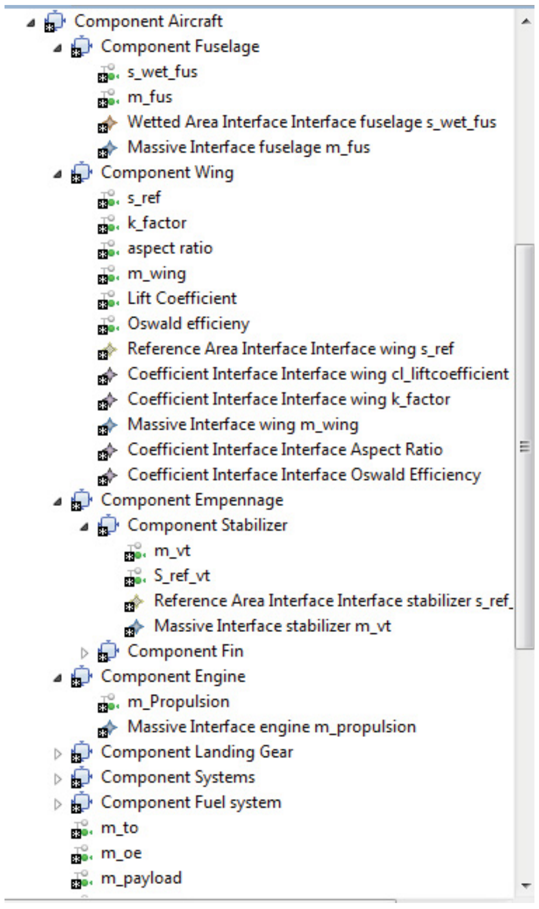
Table 1: Example data set. The dots indicate to which discipline a parameter has been assigned

small sample data set, containing three top-level components for `aircraft`, `Mission` and `Atmosphere`. Each contains further sub-components and parameters. An excerpt of the data set and the assignment of parameters to their respective disciplines is given in Table 1.

5.1 Workflow

The workflow for building a discipline view contains the following steps:

1. **Build up or extend a comprehensive aircraft model** – There a several ways of building up an aircraft model, as this can be done from scratch, by importing data sets from external tools, or by merging an existing baseline model with additional data elements. In OpenCDT, the user can rearrange the decomposition of an aircraft model by creating new subcomponents and by moving parameters and interfaces between components. The example aircraft model in OpenCDT is shown in Figure 3 and contains several components and subcomponents to describe an aircraft. The component `Aircraft` contains subcomponents for its main physical components, including `Fuselage`, `Wing` and `Systems`. Parameters and interfaces are defined for all components. The component `wing` is for example described by the two parameter `s_ref` and `m_wing`. For both parameter, the component does also offer the interfaces `Interface wing s_ref` and `Interface wing m_wing`. When defining interfaces for a parameter, the user has to choose among different types of interfaces, as described in Section 4.1.

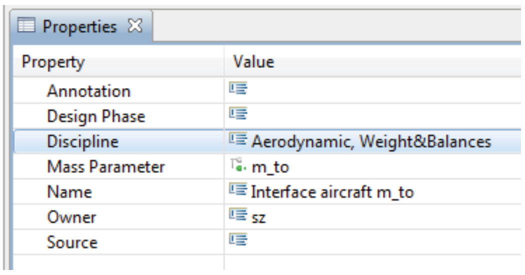


- Component Aircraft
  - Component Fuselage
    - s_wet_fus
    - m_fus
    - Wetted Area Interface Interface fuselage s_wet_fus
    - Massive Interface fuselage m_fus
  - Component Wing
    - s_ref
    - k_factor
    - aspect ratio
    - m_wing
    - Lift Coefficient
    - Oswald efficieny
    - Reference Area Interface Interface wing s_ref
    - Coefficient Interface Interface wing cl_liftcoefficient
    - Coefficient Interface Interface wing k_factor
    - Massive Interface wing m_wing
    - Coefficient Interface Interface Aspect Ratio
    - Coefficient Interface Interface Oswald Efficiency
  - Component Empennage
    - Component Stabilizer
      - m_vt
      - S_ref_vt
      - Reference Area Interface Interface stabilizer s_ref_
      - Massive Interface stabilizer m_vt
    - Component Fin
  - Component Engine
    - m_Propulsion
    - Massive Interface engine m_propulsion
  - Component Landing Gear
  - Component Systems
  - Component Fuel system
  - m_to
  - m_oe
  - m_payload

Figure 3: An example aircraft's data model in OpenCDT

2. **Annotate the aircraft model with labels** – Creating interfaces is already a first step for annotating the aircraft model with additional knowledge, due to the different types of interfaces. A further step to extend the aircraft model with design knowledge is the annotation of the model, using the predefined labels. Relevant design knowledge to be represented in the aircraft model are `Discipline`, `Source`, `Design phase` and `Owner`. The labels `Discipline` and `Owner` are self explaining, while the label `Source` refers to the external tool that a parameter has been imported from. The label `Design Phase` can mean several things, but in this example it refers to the Design Phase that introduced a parameter into the aircraft model. Table 1 shows to which disciplines the parameters of the example aircraft model have been assigned to.

   The user has to perform the annotation of the aircraft model manually, using the Properties view of an Interface. An example is shown in Figure 4. As the filter mechanism in OpenCDT does only filter interface, only interfaces have to be annotated.


Properties

Property	Value
Annotation	
Design Phase	
Discipline	Aerodynamic, Weight&Balances
Mass Parameter	m_to
Name	Interface aircraft m_to
Owner	sz
Source	

Figure 4: An annotated interface for the interface `Interface m_to`

   Annotating an aircraft model in the described way serves also the preservation of design knowledge throughout a design process.

3. **Creating discipline views and defining filters** – With the preparations in the two foregoing steps, the user is now able to create discipline views. A discipline view is created to reduce the complexity of an aircraft model by extracting only a subset of all model elements contained in that model. This is achieved by adding one or more filters to the discipline view. A filter defines the actual search criteria used for building a discipline view (see Section 4.2).

   A simple example of a discipline view is the Aerodynamics view which is shown in Figure 1. This view contains a single filter. The configuration of the filter is shown in Figure 1, in the tab called `Properties`. The filter specifies only three of the predefined criteria: The `Complement` field is set to `false`, and both the `Name` and the `Discipline` field are set to `Aerodynamics`. The corresponding Aerodynamics discipline view in OpenCDT is shown in the tab called *Aerodynamic* shown in Figure 1, and consists of two parts: to the left, the filtered parameters and the components that does contain these parameters, are shown. To the right, the filters that specify the open discipline view are listed (which in this case is only the Aerodynamic filter).

   Several other search criteria can be expressed by using a single filter. It is thus possible to perform the following searches:

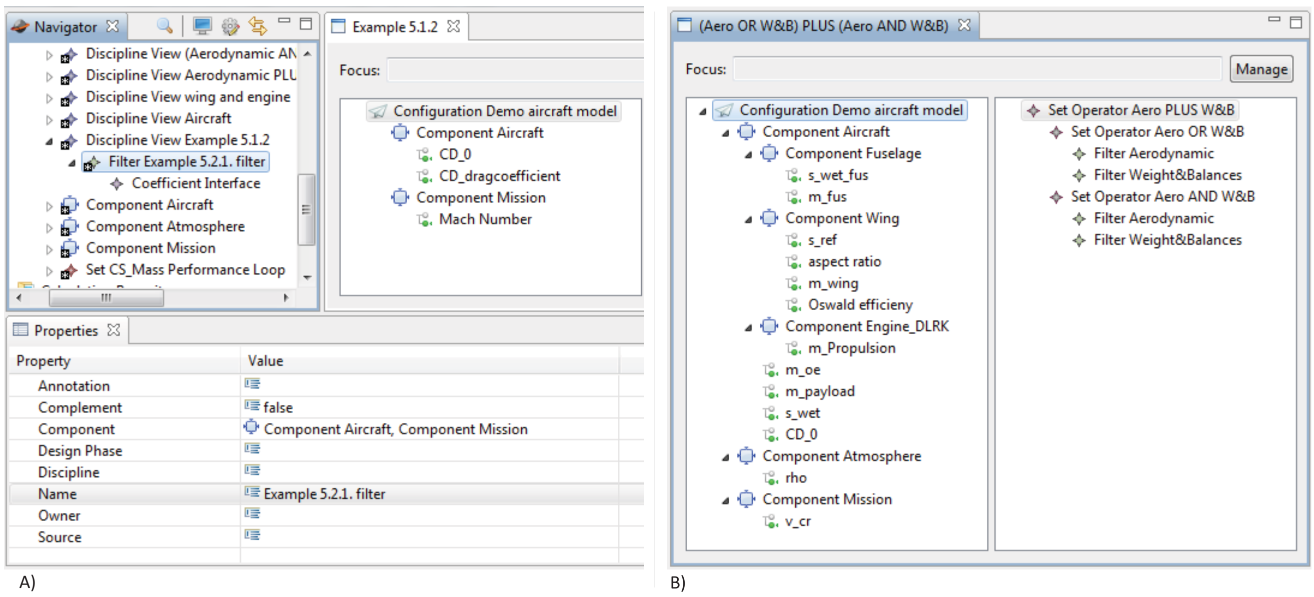
   - Select all parameters that are contained in the discipline views `Aerodynamics` and `Weights & Balances` and that are owned by the author. The following fields of a filter needs to be set as follows:
     a) Complement: `false`
     b) Discipline: `Aerodynamics, Weight & Balances`
     c) Owner: `sz`
   - Select all parameters that are coefficients and that are defined for components `Aircraft` and `Mission`. The following fields of a filter needs to be set as follows:
     a) Complement: `false`
     b) Components: `Aircraft, Mission`
     c) In addition, an Coefficient interface is added to the filter.

     A screen shot of this discipline view is shown in Figure 5, part A.

4. **Combining filters** – It is also possible to combine filters to create more advanced discipline views. This is possible by using unary and binary set operators on filters. When mixing search criteria in a filter, all criteria must be satisfied for a model element in order to include it into result of a filter. The use of set operators makes it possible to define a discipline view that contains all elements that are either part of the Aerodynamics discipline and the Weights & Balances discipline, but not part of both disciplines. As shown in Table 1, there are three parameters in the example aircraft model that would be excluded by such a discipline view. The definition of this discipline view is shown in Figure 5, part B.

5.2 Evaluation of discipline views and OpenCDT's filtering mechanism

Discipline views are a mechanism to filter large data sets. As such, they support the user in focusing on those data elements that are of relevance in a given context. They also help to reduce the complexity inherent in an integrated data set and to improve the usability when working with data elements in an integrated data set. The discipline view approach described in this paper is, thus,



Navigator

- Discipline View (Aerodynamic AN
- Discipline View Aerodynamic PLU
- Discipline View wing and engine
- Discipline View Aircraft
- Discipline View Example 5.1.2
  - Filter Example 5.2.1. filter
    - Coefficient Interface
- Component Aircraft
- Component Atmosphere
- Component Mission
- Set CS_Mass Performance Loop

Example 5.1.2

Focus:

- Configuration Demo aircraft model
  - Component Aircraft
    - CD_0
    - CD_dragcoefficient
  - Component Mission
    - Mach Number

Properties

Property	Value
Annotation	
Complement	false
Component	Component Aircraft, Component Mission
Design Phase	
Discipline	
Name	Example 5.2.1. filter
Owner	
Source	

A)

(Aero OR W&B) PLUS (Aero AND W&B)

Focus: Manage

- Configuration Demo aircraft model
  - Component Aircraft
    - Component Fuselage
      - s_wet_fus
      - m_fus
    - Component Wing
      - s_ref
      - aspect ratio
      - m_wing
      - Oswald efficieny
    - Component Engine_DLRK
      - m_Propulsion
    - m_oe
    - m_payload
    - s_wet
    - CD_0
  - Component Atmosphere
    - rho
  - Component Mission
    - v_cr

- Set Operator Aero PLUS W&B
  - Set Operator Aero OR W&B
    - Filter Aerodynamic
    - Filter Weight&Balances
  - Set Operator Aero AND W&B
    - Filter Aerodynamic
    - Filter Weight&Balances

B)

Figure 5: Part A) A discipline view selecting all coefficients for components `Aircraft` and `Mission`; Part B) A discipline view with set expressions

a first step to fulfill the requirement for *Filtering of data sets* defined in Section 2.

The possibility to define discipline views by combining different filters using binary set operators provide the user with flexibility to define individual discipline views. Filter definitions can be reused and be combined for new discipline views.

The implementation of discipline views in OpenCDT described in this paper doesn't include an integration with an user right management system, as this is currently out of scope in our work on OpenCDT. We acknowledge, however, that this is an important functionality. It should also be noted that the use of discipline views integrated with user rights are a potential approach to support another requirement of Section 2: Collaboration support. Using user right privileges is a practical approach to manage the collaboration between different disciplines teams. Discipline views could also be used to support inter-organizational collaboration.

The actual implementation of discipline views in OpenCDT is only prototypical and has, consequently, its limitations that make it unfit for day-to-day operation. These limitations include (1) the missing functionality to reuse actual filters in several discipline views, (2) to turn filters dynamically on or off when working with a discipline view, (3) to define new interface types dynamically, and (4) a real tag mechanism replacing the use of labels for extending an aircraft model with design and process knowledge. Some potential improvements for discipline views in OpenCDT are described in Section 6.

This paper has also described a typical workflow to define and work with discipline views (see Section 5.1). This workflow depends on the user manually labeling the model elements of an aircraft model. This task does require some special care for several reasons: First, as the filtering mechanism depends on the consistent use of labels in the entire model, all users of the the filtering approach must agree on a common nomenclature to be used for all model element and by all disciplines. Second, as the aircraft model developed in a project most likely will grow during the projects lifetime, the initial labeling may become to coarse-grained and may need to refined. The use of an ontology based filtering approach as described in Section 6 could reduce the amount spent on reorganizing an aircraft model.

Annotating an aircraft model with labels expressing design knowledge, such as the disciplines that an element is assigned to, or in which design phase an element has been first introduced into the aircraft model, is an activity studied in Knowledge Based Engineering (KBE). Using discipline views as described in this paper would thus benefit from already existing knowledge models for an aircraft model.

6 FUTURE WORK

The filtering approach described in this paper is only a first step towards exploiting the potential of filtering in order to support the use of an integrated data management approach. The overview of future work on filtering mechanisms in OpenCDT given in this section is divided into two parts. The first part addresses two new modeling approaches that will improve the filtering mechanism described in this paper:

- **Alternative hierarchy structures** – This approach will extend a discipline view with an alternative decomposition hierarchy for a user defined subset of the integrated data set: When displaying the filtered subset, the data elements contained in the subset will be shown using a different hierarchy. The data can be displayed in separate views that may not only present subsets of the complete dataset, but at the same time contain a separate hierarchic order of the data. An example of alternative hierarchies is shown in Figure 6. This approach is of practical use for example in cases where a subset of a large data set is needed as input for an expert software module, which requires its own proprietary arrangement of data.

- **Ontology based filtering** – Ontology based filters provide the opportunity to use an ontology to model domain specific knowledge and to use this knowledge for specifying filters. Among the practical applications of using ontology based filters is the opportunity to specify filters independent of the internal nomenclature of a data set. Current work on using ontologies for conceptual aircraft design at Bauhaus Luftfahrt addresses the semi-automatically integration of models from different sources [19].

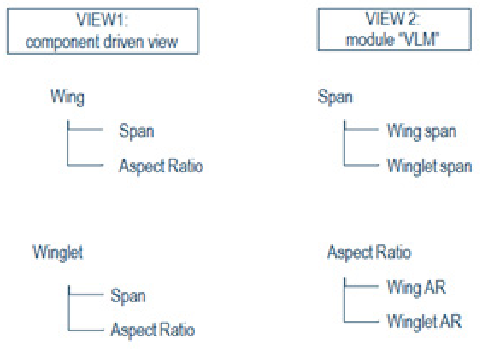


Figure 6: Two views of the same data in different hierarchic arrangements

The second part of this section addresses how discipline views can take advantage of all the data managed in OpenCDT as well as how discipline views can by used to support other functionalities.

- **Rights management** – The current version of OpenCDT does not support the management of user rights. Discipline views can be used as the sole access point to the to integrate data sets managed by OpenCDT. By extending the definition of discipline views with a specification of user rights, it is possible realize a flexible functionality for user rights. It is, for example, possible to have read-only views or to grant write access to a given parameter to the person or group ”owning” this parameter.

- **Calculation dependencies** – The only alternative decomposition structure that OpenCDT already offers are the dependencies between parameters and calculations and processes that are applied to calculate values for other parameters. This structure could be used for searches such as list all parameters that are calculated and depend on a specified parameter as input.

- **Additional filtering capabilities** – Filters can take advantage of other modeling concepts provided by OpenCDT than the one presented in this paper. An example would be the use of the versioning history available for a data set. Another concept for building data sets in OpenCDT is that a parameter can be described by many values (see Section 4.1), supporting thus different types of values such as requirements, constraints, or experimental or different state-dependent values, such as take-off, cruise, or landing. Using filters that define criteria specifically for values allows the construction of more complex discipline views.

7 CONCLUSION

This paper has addressed the software support for conceptual aircraft design and the need for having an integrated data management approach. While an integrated data set will have several benefits, including better consistency and higher accuracy, it will also introduce a more complex data set with a reduced usability due to its size.

A number of requirement for supporting a workflow using an integrated data management approach have been identified, including the requirement for powerful filtering mechanisms. Starting with a brief overview of available filtering mechanisms, this paper has described how such a filtering mechanism has been implemented prototypically in OpenCDT, an open-source tool for conceptual aircraft design.

An example has been used to illustrate how the filtering mechanism can be used in a typical workflow in a conceptual aircraft design project. Finally, the paper has outlined some ideas for future work to improve the filtering mechanism and to extend it with new ideas.

Filtering capabilities are a useful tool to reduce the complexity of an integrated data management approach, and support an aircraft designer only having access to the ”right” subset of the entire data set.

Acknowledgment

We gratefully acknowledge the contribution of Dr. Kerstin Wieczorek to the discussions leading to this paper and the aircraft model used in Section 5.

References

[1] L.A. McCullers. *Aircraft configuration optimization including optimized flight profiles*. 1984.

[2] C. M. Liersch and M. Hepperle. A unified approach for multidisciplinary preliminary aircraft design. In *CEAS European Air and Space Conference, Manchester, UK, 2009*, 2009.

[3] W. F. Tam. Improvement Opportunities for Aerospace Design Process. *AIAA Space Conference & Exhibit*, 2004.

[4] A. Giassi, F. Bennis, and J.-J. Maisonneuve. Multidisciplinary design optimization and robust design approaches applied to concurrent design. *Structural Multidisciplinary Optimization*, 38(5):356–371, 2004.

[5] M. Glas. Towards a metamodel for conceptual aircraft design. In *Proceedings of 27th Congress of the International Council of the Aeronautical Sciences (ICAS)*, pages 20–24, 2010.

[6] J. Sobieszczanski-Sobieski. *Multidisciplinary systems optimization by linear decomposition*. 1984.

[7] W.B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.

[8] Gordon V. Cormack. Email spam filtering: A systematic review. *Foundations and Trends in Information Retrieval*, 1(4):335–455, 2008.

[9] B. Sarwar, G. Karypis, J. Konstand, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW10, May 1–5, 2001, Hong Kong*, 2001.

[10] C. Soanes and A. Stevenson, editors. *Oxford Dictionary of English. 2nd edition, revised*. Oxford University Press, 2005.

[11] C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. Information filtering and query indexing for an information retrieval model. *ACM Transactions on Information Systems*, 27(2):10:1–10:47, 2009.

[12] C.J. Date. *An Introduction to Database Systems, Volume 1, Fourth Edition*. Addison Wesley Publishing Company, 1986.

[13] R. Hull and R. King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[14] S. Abitebou. and P.C. Kanellakis. Object identity as a query language primitive. *Journal of the ACM*, 45(5):798–842, September 1998.

[15] L. Reeve and H. Han. Survey of semantic annotation platforms. In *2005 ACM symposium on Applied computing (SAC '05), March 13–17, 2005, Santa Fe, New Mexico, USA*, 2005.

[16] C.B. Chapman and M. Pinfold. The application of a knowledge based engineering appraoch to the rapid design and analysis of an automotive structure. *Advances in Engineering Software*, 32(12):903–912, December 2011.

[17] Opencdt. Project web site, `www.opencdt.org`, last accessed on 30 July 2012, 2012.

[18] S. Ziemer, M. Glas, and G. Stenz. A conceptual design tool for multi-disciplinary aircraft design. In *2011 IEEE Aerospace Conference, Big Sky, MT, USA*, 2011.

[19] M. Glas. *Ontology-based Model Integration the Conceptual Design of Aircraft*. PhD thesis, Technische Universität München (TUM), Chair of Applied Software Engineering, 2012.