CEAS Space Journal
https://doi.org/10.1007/s12567-025-00597-9

ORIGINAL PAPER

Uncertainty quantification data model for the probabilistic design of the thermal protection system of a reusable launch vehicle stage

Aaron D. Koch[1] · **Jascha Wilken**[1] · **Marko Alder**[2]

Received: 31 July 2024 / Revised: 20 December 2024 / Accepted: 27 January 2025

Abstract
Early design phases significantly impact a system's life-cycle costs, yet they are fraught with large uncertainties. Hence, it is important to incorporate uncertainties in preliminary design activities. However, sampling the uncertain design space instead of analyzing a single system entails a considerable computational cost. It also demands a high degree of automation. Standardized data models can significantly aid this effort by offering a structured way of storing and handling large amounts of data. Therefore, this paper proposes an uncertainty quantification (UQ) data model for dealing with probabilistic data. Its hierarchical structure has been specified using Extensible Markup Language Schema Definition (XSD). It is complemented by a library written in C++ and provides additional Python bindings. The proposed approach is used in a case study in which the design of a thermal protection system (TPS) of the wing profile of a reusable launch vehicle stage is assessed. The UQ data model was used to store the UQ data of the TPS design study.

Keywords Probabilistic design · Uncertainty quantification · Data model · XML schema definition (XSD) · Reusable launch vehicle (RLV) · Thermal protection system (TPS)

1 Introduction

Blair et. al. [1] estimate that at least 80% of a launch vehicle's life-cycle cost is determined in the conceptual design phase. At the same time, the highest level of uncertainty is present in the earliest stages, i.e., during the conceptual and the preliminary design phase. Essential methodological tools during these phases are trade and sensitivity studies. Incorporating the effects of uncertainties into conceptual design studies can improve the robustness of the results and thus support the decision-making process when comparing different designs. Instead of just evaluating the nominal (deterministic) performance of different vehicles, a system designer can also consider the level of uncertainty surrounding a particular performance metric. During preliminary design, incorporating uncertainty can help reduce the required margins of safety. Rather than relying on worst-case assumptions, a probabilistic approach allows for a more realistic combination of various factors. This could be especially beneficial for launch vehicles and in particular for reusable versions. Reusable launch vehicles (RLV) have high-performance requirements and are therefore very sensitive to additional dry mass caused by excessive margins.

The state of the art in uncertainty quantification (UQ) in the conceptual design of (reusable) launch vehicles encompasses a variety of methodologies and applications aimed at addressing the inherent uncertainties in the design process. The fundamental task in UQ is to propagate the uncertainties in the system's input variables and the uncertainties imposed by the disciplinary models to the system's outputs (i.e., results). While input uncertainties can be classified into epistemic (i.e., lack of knowledge which can be reduced by additional knowledge) and aleatory uncertainty (i.e., cannot be reduced), most of the studies in the context of conceptual and preliminary design focus on the first category [2, 3]. This is because this phase of the design process is dominated

✉ Aaron D. Koch
aaron.koch@dlr.de

Jascha Wilken
jascha.wilken@dlr.de

Marko Alder
marko.alder@dlr.de

[1] Institute of Space Systems, German Aerospace Center (DLR), Robert-Hooke-Str. 7, 28359 Bremen, Germany

[2] Institute of System Architectures in Aeronautics, German Aerospace Center (DLR), Hein-Saß-Weg 22, 21129 Hamburg, Germany

Published online: 15 February 2025

by rather large assumptions on performance characteristics and technology assumptions for a future entry into service of the vehicle. Furthermore, decision making (e.g., in which discipline to invest resources) requires ranking the importance of inputs with respect to output uncertainties through sensitivity analysis, where Sobol indices [4] have proven to be a valuable indication of the sensitivity of the global design space [5]. Within the probability formalism, techniques based on the Monte Carlo method are typically used in the literature because they are easy to implement in multidisciplinary design tasks (disciplinary models can be considered as black boxes) and the propagated output uncertainty can be characterized by statistical moments, such as mean, variance, skewness, or kurtosis. However, nearly all studies recently found in the literature aim at reducing the computational cost of classical Monte Carlo approaches. Thunnissen et al. [6] advance the latter through subset simulation, a modified version of an efficient probabilistic method. It is applied to conceptual design problems such as estimating fuel mass for attitude control, providing computational efficiency especially for risk-averse decision makers. Brevault et al. [7] apply surrogate models such as reduced-order models and spectral methods to the trajectory optimization of two-stage-to-orbit launch vehicles. Smith and Mahadevan [8] employ reliability-based design optimization on a reusable launch vehicle using sequential optimization and reliability assessment as proposed by Du and Chen [9].

Although mathematical methods exist to address these issues [2], UQ is not yet standard practice for launch vehicle design and analysis. The broader adoption is hindered by the additional effort it requires. This effort is twofold. Firstly, there is a substantial computational burden because a single simulation is replaced by a sampling of the uncertain design space, often involving thousands of system evaluations. Secondly, designers need to acquire additional knowledge regarding methods and tools for defining, executing, and evaluating probabilistic studies. While research on UQ methods has resulted in a large number of software products, most of which have been made available as open source, each product usually has a high level of complexity in terms of freely selectable meta-parameters, a large variety of methods for solving complex problems, and individual visualization techniques. In cases where different software solutions need to be combined or where design experts need to communicate and share knowledge, problems arise with efficient storage and robust communication of UQ data (i.e. misinterpretation of parameter meanings, value units, etc.). This is due to the fact that there is little scientific research on the standardization of interfaces, which is a fundamental prerequisite for digitization.

To our best knowledge, the most comprehensive progress in developing standards and guidelines on UQ, including verification and validaton (V&V) of methods, has been made by the American Society of Mechanical Engineers (ASME), which has published a series of documents providing a "common language and definitions, conceptual framework for V&V and UQ, methods for V&V and UQ, guidance for implementation, and best practices" [10]. However, it strongly focuses on the development and use of disciplinary tools, while the integration of a holistic design process tends to involve black-box tools and is dominated by the propagation of uncertainties rather than the V&V of individual tools. Furthermore, the introduction of a "common language" for UQ evolves from the documents and guidelines, but they do not explicitly provide a parameter dictionary or naming standard, which could directly be applied for defining data interface models.

In this regard, it is helpful to look beyond the topic of UQ to other areas of research that are dealing with the issue of data management and robust interface standards. In the early design of aerospace transportation systems, domain-specific data models such as CLAVA [11] and CPACS [12] provide schema-based hierarchical parameterizations of space launchers and aircraft, respectively. Similar solutions are found in other domains, for example, Holispec [13] for maritime transport systems. Based on such domain-specific languages, the overall design process integrates disciplinary tools (typically provided by experts as black-box tools) into automated design workflows. Therefore, the domain-independent data model CMDOWS [14] proposes a standard on how such workflows are composed and which of the domain-specific exchange data is transferred between the involved tools. The combination of data models for multidisciplinary design and optimization workflows (CMDOWS) with domain-specific data (CPACS, CLAVA, etc.) paves the way for efficient and robust communication of engineering design data. While a similar approach for the standardized specification of uncertainty could expand this approach towards UQ, no solution has yet been found in the literature.

The German Aerospace Center's (DLR) project for the probabilistic technology assessment of complex transportation systems (PROTEKT) aims at the development of a framework for the analysis and propagation of uncertainties in multidisciplinary design processes, which essentially consists of three pillars: (1) modeling uncertainties and managing uncertainty data, (2) proposing consistent methods, tailored for specific application cases, to propagate uncertainties within the multidisciplinary design process, and (3) supporting the assessment, analysis, and visualization of the results. Together, these three pillars form the basis of a framework that aims to make UQ more accessible in the early design stages. The present study investigates to what extent this can be achieved by technologies for efficient and robust exchange of UQ data.

Therefore, this paper presents the development of a UQ data model, complemented by a software library for

high-performance data storage and queuing. The focus is on probabilistic UQ via sampling-based Monte Carlo approaches. These are typically challenging because they require high computational resources, and also produce large amounts of data, for which a structured and standardized approach to data management is desirable. Sec. 2 presents the UQ data model in detail along with the development of an associated software library for reading and writing the data. The benefits of such an approach are demonstrated in Sec. 3 by performing a preliminary sizing of a thermal protection system (TPS) of an RLV stage. Then, the advantages and disadvantages as well as future extensions of the UQ data model are discussed in Sec. 4. Finally, Sec. 5 provides a conclusion and a short outlook.

2 Data management in uncertainty quantification

The strategy for managing UQ data in probabilistic design studies presented herein is based on the following basic assumptions: (1) Monte Carlo-based design processes generate a large number of samples, which require efficient processing and memory management, (2) disciplinary analysis modules are regarded as black boxes, which require non-intrinsic propagation methods, and (3) a higher-level discipline (space, aviation, transportation, etc.) employs a domain-specific data model, which should not be modified in the context of a UQ study and which can be referenced to supplement probabilistic information. The following sections first discuss the use of domain-specific data models in the context of multidisciplinary design processes. Then, a dedicated domain-independent data model for uncertainty data is introduced. Further, the UQ data model's implementation as a C++ software library with bindings to the Python programming language for efficient and robust data handling is addressed. Finally, the usage of the data model within a larger framework is outlined.

2.1 Data modeling in a deterministic design process

As a prerequisite for the results presented in this study, classical deterministic design processes are first introduced. The task of designing complex transportation systems, such as an RLV, is to find a consistent (and ideally optimal) solution to a multidisciplinary system of equations. This is called multidisciplinary analysis (and optimization) (MDA(O)). Experts from different disciplines contribute a variety of analysis modules, which are usually considered as black box modules due to the use of individual (and often proprietary) solution algorithms and software architectures. Previous studies have shown that the resulting complexity of MDA(O) workflows can be addressed by standardized interface definitions. These include domain-independent data models such as CMDOWS [14], which describes the general composition of the design process, and domain-specific data models such as CLAVA [11] or CPACS [12], which serve as a central source of truth for the actual product under consideration. Typically, such data models are structured hierarchically. A suitable technique for implementing data models is the Extensible Markup Language (XML) Schema Definition (XSD) [15], which can be used to specify and validate data in the XML data format. Alternative data modeling approaches include the Unified Modeling Language (UML) (s. e.g. [16]) or schema-free techniques derived from Semantic Web Technologies, such as the Resource Description Framework (RDF) or the Web Ontology Language (OWL). Since any data given in XML and XSD can be converted to RDF, it can be used to semantically link data from different domains and infer new knowledge from such [17, 18].

Since this study focuses on modeling UQ data, a central schema-based approach is desired, which is why XSD is chosen to implement the actual UQ data model. In addition, an object-oriented implementation using C++ is investigated to provide high-performance data queuing methods even for large amounts of data. The conversion to RDF or OWL offers the potential to connect additional information such as provenance data, but this is beyond the scope of this paper.

2.2 Uncertainty quantification data model in a probabilistic design process

This section will first introduce the probabilistic design process. Then, the UQ data model is presented in detail. Next, the actual usage of the data model within the described probabilistic design process is illustrated.

2.2.1 The probabilistic design process

Figure 1 shows a probabilistic design and analysis process in the form of an extended design structure matrix (XDSM) [19], which is comprised of three distinct steps: (1) a sampling algorithm is used to generate the probabilistic input factor sets $\mathbf{X}$ for the actual design process, which is composed of (2) at least one disciplinary black-box model (usually more) solving for the solutions $\mathbf{y}_i$ of deterministic equations (in this study this step is substituted with a complex design process as shown in Sec. 3). In step (3), the results are collected and analyzed with respect to their statistical properties or sensitivity measures. A detailed description of this process is given in [5].

2.2.2 The uncertainty quantification data model

A data management strategy should be able to map the three steps shown in Fig. 1, considering that the

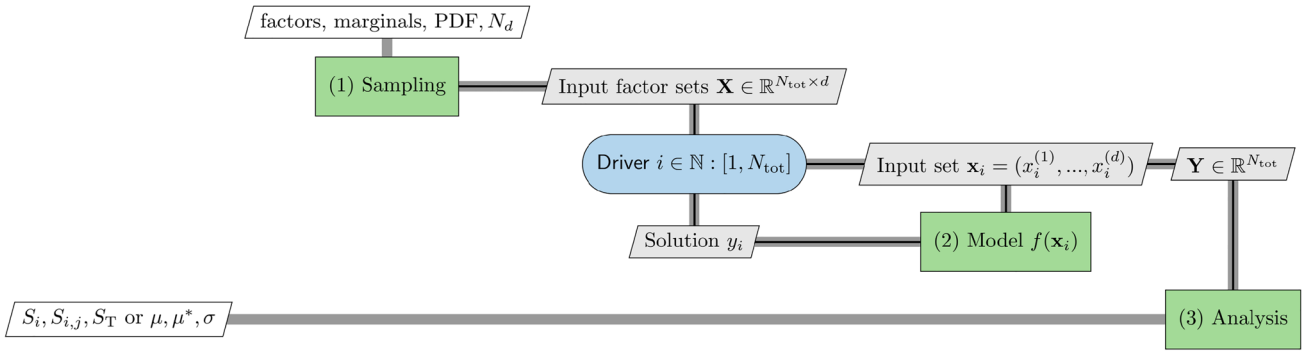

Fig. 1 XDSM representation of the data exchange in a probabilistic design process

disciplinary modules should be treated non-intrusively and the domain-specific data model, s. Sec. 2.1, should be left unchanged so as not to interfere with its use in deterministic analysis. For this purpose, we have designed a UQ data model, which utilizes XSD to define its schema, s. Online Resource 1. In this context, the schema describes how the data is structured and which parameters are used to store the data. It also defines the data types of the parameters. The actual data from a probabilistic study are stored in an XML file that adheres to the structure defined in the schema.

The hierarchical structure of the proposed UQ data model, version 0.23, is shown in Fig. 2 in terms of an XSD diagram. This type of representation visualizes the hierarchical dependency of the corresponding elements, with solid and dashed frames indicating mandatory and optional elements, respectively; while deviating occurrences are represented by a stacked block symbol with an indication of the minimum and maximum number of occurrence (e.g. from 1 to infinity for `study`). The XSD elements, `sequence` and `choice`, are represented by an eight-sided shape containing three horizontal or vertical dots, respectively. A `sequence` contains several other elements in a given order, while a `choice` allows a choice between different elements. When the last element in the presented tree-structure contains a plus on the right side, then it contains a `sequence` with additional parameters. Their contents are separately depicted in Figs. 3 to 6. An important detail that cannot be inferred from the diagram is the usage of the XSD data types `ID` and `IDREF`. These are here exclusively used as attributes of elements and their purpose will be explained when the respective element is described in the following paragraphs.

The root element of the UQ data model is called `uq`. It first contains a `header` element with the mandatory elements, `name` and `version`. Like everywhere in the model, the `description` element is optional. It can be used to provide any kind of detail, e.g., the purpose, the rational for making specific choices, or literature references. In addition, the `header` contains an optional `domainFile` element. This can be used to reference a domain-specific data file as explained in Sec. 2.1.

The first step of the probabilistic design process (s. Fig. 1) is represented by the `variables` element, which allows to predefine the `inputs` and `outputs` of the process in terms of a `name`, `description`, and a reference (`xPath`) to the corresponding domain-specific variable inside the domain file. The latter is currently implemented via XPath [20] but could in the future also be extended by other means of references, such as indices for cells and rows in a table, stored as a comma-separated values file. This approach allows enriching the domain-specific data with uncertainty information, without modifying it (i.e., keeping it independent from the research in UQ). Input variables (referred to as factors in Fig. 1) are furthermore specified by a probability density function (PDF, including marginals), which is currently covered by a parametrization of beta, (truncated) normal, triangular, and uniform distributions. This list can easily be extended by further PDFs. The composition of the implemented PDFs is shown in Fig. 3. All the distributions have the elements `lowerBound` and `upperBound`, which are optional in the case of the `normalDistribution` element. In addition, the `betaDistribution` is characterized by the parameters, `alpha` and `beta`, the `normalDistribution` by `mean` and `stdDev`, which is the standard deviation, and the `triangularDistribution` by its `mode`. Each `input` and `ouput` has an `id` attribute of type `InputIdType` and `OutputIdType`, respectively. An identifier (ID) uses the XSD type `ID` but is further restricted by the type of element it denotes. The ID of an input uses the prefix `in` and the ID of an output the prefix `out`. This means that a valid reference to a variable can be limited to either input or output elements. The UQ data model uses camelCase for IDs. Digits are allowed anywhere within the ID. So, `study1` is an acceptable ID. Also, valid examples for the main stage diameter of a rocket are `inStageDiameter` and `outStageDiameter`. As this example shows, it is possible to use the same base name, here `StageDiameter`, when

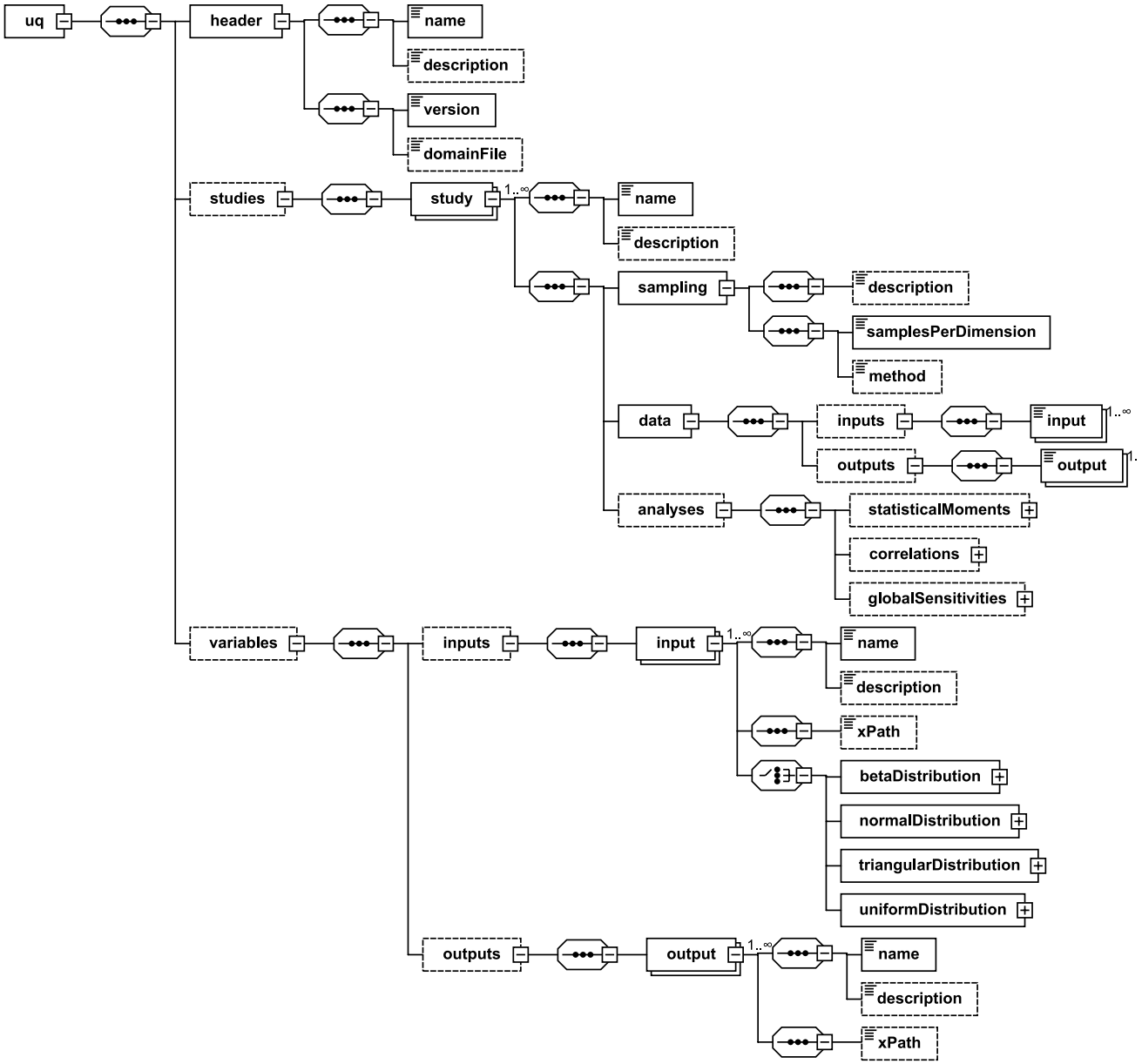

Fig. 2 Hierarchical structure of the uncertainty quantification (UQ) data model

a variable is used as an input in one study and as an output in another. The base name is, however, not a reliable way to determine if two variables are the same, as the variable naming is entirely decided by the user.

The actual probabilistic `studies` are aggregated under a single element. Each `study` inside it has a `name` and a `description`. In addition, a `study` carries specific `sampling` information. These are currently a `description`, the number of `samplesPerDimension`, and the sampling `method`. The variables that are used in a specific study are chosen from the `variables` segment and referenced in the study `data` segment. Each study `input` and `output` contains an `idref` attribute of XSD type `IDREF`. These references are, however, restricted to the variables type with the correct prefix, in this case, either `in` or `out`. This is achieved by defining the custom types, `InputIdrefType` and `OutputIdrefType`. The input data represent the values of the design variables $\mathbf{x}_i$ of the domain-specific model, shown as green component (2) in Fig. 1. The output data, on the other hand, match the values of the coupling variables $\mathbf{y}_i$, once the MDA(O) step is finalized. Figure 1 shows a single domain-specific model but this block could consist of several interconnected modules and within that workflow the $\mathbf{y}_i$ can function both as inputs and outputs. Relevant to the probabilistic process are only the final values that these variables assume. Further,

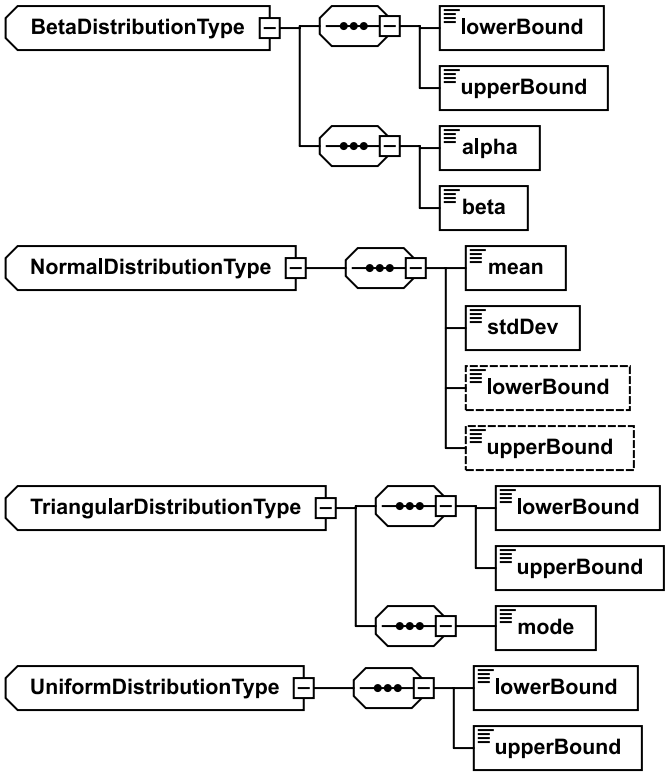

Fig. 3 Distribution types of the UQ data model

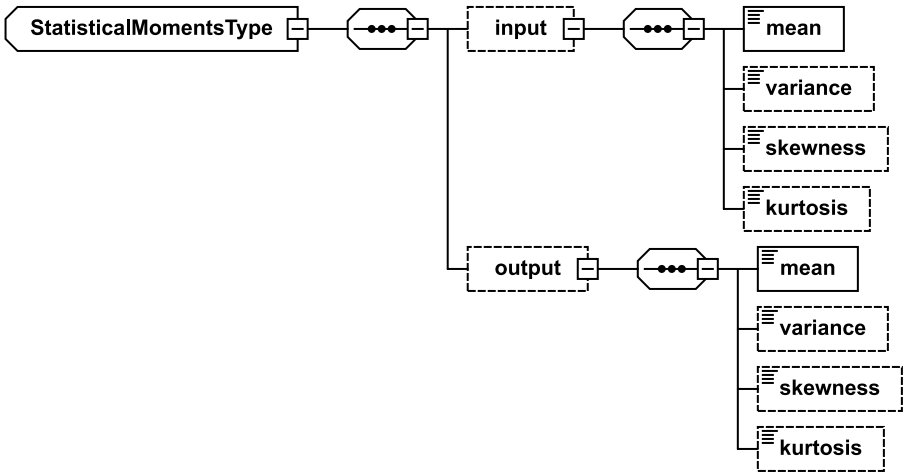

Fig. 4 `StatisticalMomentsType` of the UQ data model

each `study` has an `id` attribute with the prefix `study`. As before, this ID uses `camelCase` with digits allowed at any position. It is utilized when accessing the studies contained in the data model from outside. It is safer to access them via an ID than by their numerical position in the XML file.

Finally, the third component (3) in Fig. 1 comprises the analysis of the uncertainty data Y. This is accomplished by a dedicated `analyses` element containing information on `statisticalMoments` (Fig. 4), `correlations` of variables (Fig. 5), and quantification of `globalSensitivities` (Fig. 6), such as `FAST`, `Morris`, and `Sobol` sensitivity indices. The first four moments, `mean`, `variance`, `skewness`, and `kurtosis`, can be specified for the `input` and `output` data. These entries utilize the XSD type `list`, and the length of the list equals the number

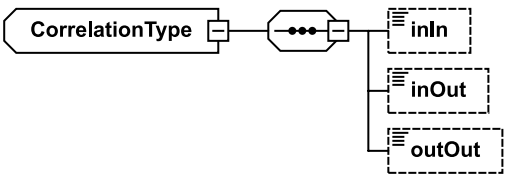

Fig. 5 `CorrelationType` of the UQ data model

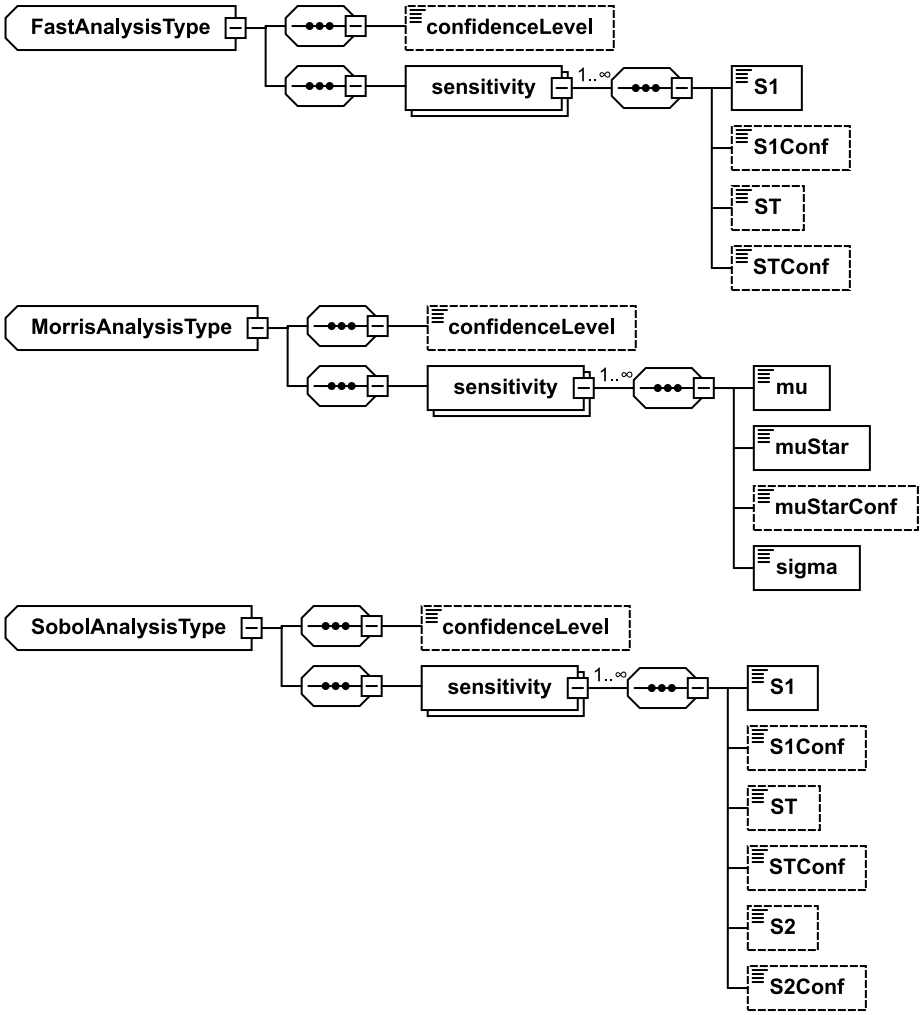

Fig. 6 Global sensitivity analysis types of the UQ data model

of study inputs or outputs, respectively. Next, `Pearson` as well as `Spearman` correlations can be defined in the data model. They are both of type `CorrelationType`, which contains three elements, `inIn`, `inOut`, and `outOut`. These store the correlation coefficients between inputs and inputs, inputs and outputs, as well as outputs and outputs. There is no inherent matrix data type in XSD but arrays can be saved using the data type `list`. So, a straightforward way of storing multi-dimensional data is to flatten the matrix into a one-dimensional array. This data model utilizes row-major order. In the case of the correlations, only the data above the diagonal is stored. The data below the diagonal is redundant and the values on the diagonal are always one. Lastly, each type of global sensitivities element, like `Sobol`, holds a `confidenceLevel` parameter. This is the confidence interval level used to calculate the confidence intervals belonging to the respective sensitivity indices. The indices themselves are contained in the `sensitivity` element, one for each output of interest. Each `sensitivity` contains an `idref` attribute. It has the XSD type `IDREF` but is restricted to reference IDs that use the prefix `out`. For further details on the three available methods and their indices s. [5, 21]. For `FAST`, the parameters are called `S1` and `ST`. Both are arrays with a length that equals the number of study inputs. `Sobol` has the same parameters plus the parameter `S2`. This parameter is a two-dimensional matrix and for the same reasons as for the correlations, only the values above the diagonal are stored as a flattened array. The length of that array equals $\frac{n(n-1)}{2}$, where n is the number of study inputs. Finally, `Morris` contains the parameters `mu`, `muStar`, and `sigma`. They each have a length equal to the number of study inputs. Each of the parameters `S1`, `ST`, `S2`, and `mu` have an additional confidence interval parameter defined. They are denoted with the parameter name plus the suffix `Conf` and have the same length as the parameter they belong to.

An additional design decision when creating the schema was to define abstract base types and then extend them to show that several types belong together. This was applied for the probability distributions (`BaseDistributionType`), for the variables (`BaseQuantityType`), the global sensitivity analyses (`BaseSensitivityAnalysisType`)

and the individual sensitivities (`BaseSensitivityType`). The base types, `BaseDescriptionType` and `BaseNameDescriptionType`, cover the often-occurring case that a type contains an optional description element or in addition a mandatory name element. The usage of base types also explains why the XSD element `sequence` is used instead of `all`. Using `all` means that unlike in a `sequence`, the order of the elements can be freely chosen. This can be the preferred behavior in cases where zero or one of each element occurs. Its usage is, however, not allowed when extending types. Therefore, `sequence` is used everywhere in the schema to stay consistent.

Another configuration choice is the definition of container nodes. This concerns the `studies` node, as well as the `inputs` and `outputs` nodes that both appear twice in the UQ data model. A container only holds elements of a single data type. To denote this relationship, the container is named by forming the plural of the name of the element it contains. As an example, the `inputs` container stores `input` elements. It is not strictly necessary to use containers. The `data` element could directly include an `input` and an `output` element with an occurrence of zero (`minOccurs="0"`) to infinity (`maxOccurs="unbounded"`). The advantage of containers is that it is easy to handle all elements of a single type together. The study inputs and outputs, e.g., are often read or written as an entire block. It also simplifies accessing elements of a specific type that are identified via their ID, like a `study`. Both behaviors are matched by data structures commonly found in programming languages, and this is further addressed in Sec. 2.3.

One of the most important architectural decisions was to separate the variables definitions from the actual data storage inside the study elements. This has two advantages. First, the variables can be reused in different studies, e.g., to perform convergence studies, where the sampling size is varied. Secondly, the custom data types used to model the input and output data are very lean. `StudyInputType` and `StudyOutputType` consist only of a list of double values and an attribute each. This means that the `data` segment inside the XML file is very compact. It also facilitates alternative implementations of data storage in binary formats (s. Sec. 4) or as data structures in programming languages (Sec. 2.3). It is still possible to add variables when the UQ data model already contains previous studies. The new variables can then be included in subsequent studies. It is even thinkable to add outputs to a completed study when the values for those outputs are still available. In this case, they should be added after the existing outputs. This means that one-dimensional analyses results, like the mean, can be easily extended by appending the results to the new outputs. In the case of multi-dimensional results, however, it is easier to rewrite those result items instead of inserting the additional values at the correct locations. By contrast, adding input variables to a study always requires rerunning the entire study because all inputs need to be included in the sampling step.

Overall, the structure of the UQ data model is primarily hierarchical because it divides the data into several levels of significance and defines parent–child relationships. A `study`, e.g., contains a `sampling`, `data`, and `analyses`. An advantage of this architecture is that information that belongs contextually together can be easily read and written as a whole. For example, all header information is contained in the `header` element. The information contained in the `study` element is sorted according to the steps of the probabilistic design process in Fig. 1. As the steps of the process itself are considered fixed, the model focuses on the data generated by this process. It only stores some settings needed by the probabilistic design process, like the specification of the `sampling`. An exception to the tree structure is brought in by the cross-references implemented via the XSD types `ID` and `IDREF`. In this case, the model more resembles a graph structure. A study `input` or `output` references an `input` or `output` definition in the `variables` segment. References are a feature that helps to avoid redundancies in data models.

2.2.3 Usage of the UQ data model during the probabilistic design process

The UQ data model is used throughout the probabilistic design process shown in Fig. 1. Its usage can be divided into three distinct steps, which are preprocessing, sample handling, and postprocessing. The following paragraphs provide instructions for a typical use case, where a UQ data model is set up from scratch, a domain-specific file is used, and a single probabilistic study is run, including postprocessing.

The preprocessing step occurs before the design process is even started. First, create a `uq` element and add a `header` element underneath it. Then, at least specify the `name` and `version` element of the `header` to describe the contents of the file. To make use of a domain-specific data model, as explained in Sec. 2.1, set `domainFile` to the path where the respective file is located. Next, define the input and output variables that will be used in this instance of the UQ data model. For this purpose, insert the `variables` element into the `uq` element, along with its children `inputs` and `outputs`. Add each `input` to the `inputs` element and at the minimum define for each a `name` and a distribution with its respective parameters. Insert each `output` to the `outputs` element and provide at least a `name` for each. In addition, set the `id` attribute of each `input` and `output`, using the prefixes `in` and `out`, respectively. Also, as a domain-specific file is being used, provide the location of each variable inside that file by populating the `xPath` element with an appropriate value. Finally, insert the

studies element into the uq element and below it add a single study element for the probabilistic design process that will be run next. Set its id attribute using the study prefix. Add the data element to the study, along with its children inputs and outputs. Chose the desired input quantities among the inputs defined under variables. For each input, create a reference to its definition in the variables section by setting its idref attribute to the corresponding value. Otherwise, leave each input element empty. Perform the equivalent steps for the outputs. Insert the sampling element into the study. Below it, specify the sampling method element and the samplesPerDimension element.

Once the outlined data has been defined, the probabilistic design process, s. Fig. 1, is started. Its step (1) is the sampling of the inputs, defined in the current study, using the chosen method with its samplesPerDimension parameter. The sampling step is complete when the values for all inputs have been produced. For each input, add the generated values as a list. Now, the input elements will be filled, while the output elements will still be empty. Their values are generated by running step (2), the MDA(O) model, for each sample. The domain-specific file acts as an input to this model and a file with the correct values needs to be created for each sample. First, aggregate the values of the inputs into a matrix. Assuming row-major order, the simplest way to handle this is to iterate over the inputs and store the value list contained in each input as a row in a matrix. After finishing the data extraction, transpose the matrix so that each row of the matrix contains the values for a single sample. Now, generate the input files by iterating over the rows and writing the sample values into a copy of the original domain-specific file. The correct locations for those values are designated by the corresponding XPath expressions of the input variables. For every sample that has finished its MDA(O) run, the domain-specific file will now contain the computed outputs and a reverse procedure has to be applied. The XPath expressions of the output variables indicate where in the file the output values are found. Extract those values and gather them in another matrix. Once the entire step (2) has finished, transpose the output matrix so that is sorted by outputs rather than samples. Iterate over its rows and transfer the values to the study outputs of the UQ data model.

The last step is the postprocessing of the generated data set. It is denoted as step (3) Analysis in Fig. 1. Add the analyses element to the current study element to store all analysis results. Assume that we are interested in the statistical moments of the output variables and the results of a Sobol analysis. Below the analyses element, add the statisticalMoments element, and then underneath that an output element. After calculating the mean and variance of all outputs, add elements with the same name to the analysis output element. For each, insert the calculated values as an XSD list entry. Next, create a globalSensitivities element under the analyses element, and subsequently add a Sobol element to the globalSensitivities element. If you are interested in the confidence intervals of the Sobol indices, then add the confidenceLevel element to the Sobol element. Provide its value. SALib [21], e.g., uses a default value of 0.95. For each output for which you would like to store the results of a Sobol analysis, add a sensitivity element with an attribute idref referencing the respective output variable. Add the correct elements (S1, ST, S2) for storing the Sobol indices and their respective confidence intervals (S1Conf, STConf, S2Conf) to each Sobol sensitivity element. For each, insert the calculated Sobol sensitivity analysis values as an XSD list.

The UQ data model for this single study is now complete. Optionally, provide additional information by adding description elements. It is especially useful to add a short summary to the study and the header element. When the file contains more than one study, then it is helpful to briefly describe its contents.

2.3 Implementation of the UQ data model as a C++/Python software library

Once a standardized, hierarchically structured schema, like in Fig. 2, has been created, there are at least four ways one could work with it. (1) Manually accessing the XML file, which is tedious and error prone, and therefore only suitable for handling a small number of parameters. It also does not fit into an otherwise automated workflow process. (2) Directly working with the XML data using, e.g., the Python package, xml.etree.ElementTree [22]. This approach is still tedious and error prone as the data in the XML file is accessed step by step and the code has to be adapted to each new use case. It also does not make usage of the schema, which provides type definitions that are akin to classes in object-oriented programming. (3) Auto generating a hierarchical class structure using an existing XSD code generator. The disadvantage here is that the generated classes merely provide read and write access and lack additional functionality. (4) Creating a common software library specifically for the schema. The schema plus the library constitutes the full data model. It comprises of the hierarchical structure, defined through the XSD schema, and the added functionality provided by the library. We previously used this approach for the CLAVA data model [11], which features a library using C++ and provides Python bindings via Boost.Python [23]. This hybrid approach combines the speed of C++ with the user-friendliness of Python [24]. As this is an established approach, that code base was reused within PROTEKT to develop the UQ library for the UQ data model.

In addition to using C++, utilizing a fast XML parser, as well as minimizing the number of read and write operations, ensures a good performance of the data model library. When a user wants to work with an existing data model that is stored as an XML file, then the content of that file is parsed using an XML Document Object Model (DOM) parser. Kapoulkin's C++ library, pugixml 1.13 [25], was selected for this purpose because his benchmarks showed that pugixml was the fastest C++ XML parser on an x64 architecture at the time of testing. After the DOM tree is constructed, it is recursively translated into a class-based representation of the schema. While working with the data model, the entire model is kept in memory. The data is only converted back into a DOM tree and then written into an XML file when the user explicitly executes the model's write function. While this has the disadvantage that progress is lost should a crash occur, the data can usually be recreated by rerunning the respective Python script.

The custom, complex XSD types like `study` are implemented in C++ as classes. All classes are derived from a common Base class that provides the basic functionality for reading data from XML and writing data to XML. In contrast, native XSD types like `double`, used e.g. for `mean`, are implemented by matching them with native C++ types, in this case double. The reference mechanism provided with the XSD data types `ID` and `IDREF` are implemented in C++ using shared pointers. The containers, mentioned at the end of Sec. 2.2.2, are matched by data structures found in C++. When the order of elements in a container matters, then a vector type is used. This applies to the `inputs` and `outputs` in the study `data` element. When an ID is used to identify the elements inside a container, then a map type is employed. This is, e.g., the case for the `studies` node. The key that is used in the map is formed by removing the ID's prefix and lowercasing the first letter of the base name to adhere to camelCase. For example, the ID `studyRocketSobol` would yield the key `rocketSobol`. If the base name starts with a digit, then no change occurs. So `study1` would result in `1`. The Python bindings translate vector and map into list and dictionary, respectively.

In addition to implementing the schema as defined in XSD, the UQ library offers additional functionality for often occurring tasks. For example, Sec. 2.2.3 describes two tasks as part of step (2). The first is creating a domain-specific file that contains the sampled input data. The second task is the opposite, i.e., reading the output data from a domain-specific file that results from the MDA(O) process. These jobs are covered by the C++ functions, `Study::realize_sample` and `Study::load_sample`, respectively.

Moreover, using a software library creates another level of abstraction. The usage of XML/XSD could be replaced by another format without changing the functionality of the library itself, s. Sec. 2.1 for alternatives. XSD offers a relatively easy way to define a hierarchical structure, where additionally elements can reference each other. The overall functionality is, however, limited. Switching to its latest implementation, version 1.1 [26], would add useful features, like conditional type assignment. Unfortunately, to our knowledge, only commercially available editors fully support XSD 1.1. This substantially limits its potential for wider adoption. One of the main ideas behind a standardized format is data exchange between users. So, if a specialized software with additional costs is required, then not everyone will be able to access the data with all its properties.

2.4 Usage of the UQ data model in the UQ framework

Within DLR's PROTEKT project, the UQ data model serves as the central source of truth of the UQ framework, mentioned in Sec. 1. The framework consists of a backend, which integrates the UQ data model, and a frontend, which provides a graphical user interface. The framework utilizes the software library described in Sec. 2.3 to implement the steps outlined in Sec. 2.2.3. For the most part, the backend keeps the UQ data in memory but at specific points during the probabilistic design process the data is saved as an XML file. These save points occur after the generation of samples and before the workflow execution, after retrieving all processed results from the workflow execution, and after an analysis step, like a Sobol analysis, has been performed.

3 Case study

The purpose of the following case study is to show a practical application of the UQ data model, as introduced before. The next three subsections focus on the case study itself, while the final Sec. 3.4 addresses the application of the data model.

3.1 Thermal protection system design

A full design process for a launch vehicle necessarily includes the assessment of many subsystems, which are tightly coupled. To keep the number of degrees of freedom to a manageable level and avoid time-consuming iterations, in this case study the portion of the workflow was singled out that directly affects the design of the TPS system.

Herein, the case study contains the following analysis steps:

- Assessment of trajectory performance
- Derivation of an aerothermal database
- TPS sizing for a given wing profile

Given the computational cost of repeatedly sampling the toolchain, rapid methods are required to assess the effect of the uncertainties on the individual subsystems. Instead of a full optimization the descent trajectory is controlled with a proportional-derivative controller that ensures that the maximum permissible lateral acceleration of 2g is not exceeded. After the initial deceleration a constant flight path angle of $-1°$ is targeted. The aerodynamic and aerothermodynamic properties along the trajectory are assessed with the DLR tool HOTSOSE [27], which can rapidly estimate the values for hypersonic flow with surface inclination methods [28]. Finally, the local TPS thickness is sized with the DLR tool TOP3, which integrates the thermal response of the TPS in one dimension and subsequently varies the insulation thickness until the temperature limits of the underlying structure are respected. The different tools used to perform these steps are combined via a Python script and evaluated for each sample.

Within these three steps, the following uncertainties are considered in this case study

- Aerodynamic reference area: In trajectory integration, the reference area is utilized to compute the forces acting on the vehicle based on the aerodynamic coefficients. Variations in this typically fixed value, linked to the aerodynamic dataset, are used to represent the uncertainty in aerodynamic forces during reentry.
- Reentry mass: This is the initial mass at the start of the reentry trajectory integration. This uncertainty can stem from the assumptions made for the vehicle's dry mass during conceptual design but can also appear during the actual flight, when it is, e.g., uncertain how much of the residual and reserve propellants still remain in the stage. Other possible sources are variations in the production of each individual vehicle.
- Initial conditions for reentry: Depending on external factors during ascent and the performance of the guidance, navigation, and control systems, the conditions at stage separation can vary from mission to mission. Herein, uncertainties in the initial flight path angle (FPA), altitude, and velocity are considered. As can be expected, the initial conditions have a major impact on the trajectory, including the thermal loads experienced during the reentry.
- Transition Reynolds number: The transition from laminar to turbulent flow has a significant impact on the local heat flux and its prediction is notoriously uncertain. Local flow perturbations can trigger the transition early and lead to local hot spots.
- TPS panel emissivity: The majority of the heat flux that enters the TPS is radiated outward. The thermal insulation assures that only a small fraction arrives at the vehicle's main structure. Therefore, the emissivity of the TPS surface panels has an impact on the necessary insulation thickness.
- Heat flux uncertainty: This parameter accounts for the general uncertainty in deriving the local heat loads numerically.
- Heat transfer coefficient and ambient temperature during non-hypersonic flight: For the estimation of the heat loads, the HOTSOSE tool, which estimates the aerodynamic and aerothermodynamic properties with surface inclination methods, is used. This approach works well for hypersonic speeds but is not applicable for lower supersonic and subsonic speeds. While the highest heat loads are present during the hypersonic phases, the slower phases also have to be modelled to correctly account for the cooling of the structures. Within this study a fixed heat transfer coefficient and ambient temperature are assumed and uncertainties are applied to them.

In this use case, all uncertainties were considered as uniform variations around the reference value. The specific relative variation and reference values are given in Table 1. The values are chosen arbitrarily for this case study and do not necessarily reflect relevant cases. The variation for the

Table 1 Considered uncertainties and their variation and reference value, all sampled uniformly

Uncertainty	Relative variation	Reference value
Aerodynamic reference area	[0.9,1.1]	461 m^2
Reentry mass	[0.9,1.1]	229657 kg
Initial velocity	[0.99,1.01]	3772 m/s
Initial flight path angle	[0.9,1.1]	5.06°
Initial altitude	[0.9,1.1]	62.26 km
Transition Reynolds number	[0.5,5.0]	1×10^7
TPS Panel emissivity	[0.9,1.1]	0.81
Heat flux	[0.9,1.1]	Varies for each samples
Heat transfer coefficient for non-hypersonic flight	[0.9,1.1]	20 W/(Km^2)
Ambient temperature for non-hypersonic flight	[0.9,1.1]	200 K

velocity is set very small, as its impact on the heat flux is known to be large and follows a cubic trend.

Within HOTSOSE, methods for laminar and turbulent boundary layers are implemented. However, no models are implemented for the transition phase. Instead, every point on the mesh with a Reynolds number above the transition Reynolds number is treated as fully turbulent and otherwise as fully laminar. In contrast to the transition usually encountered in subsonic flows, recent results indicate that for the hypersonic boundary layer transition two heat flux peaks are formed. The first local heat flux peak is found at the maximum amplitude of the second mode instability and the second heat flux peak is caused by the full transition to turbulent flow [29–31]. These new insights could be implemented in a future more detailed analysis. This, more complex, hypersonic transition would likely also include additional uncertainties in the modelling.

The transition Reynolds number variance used herein is based on the values given in [32] for a cone at 0° angle of attack and is chosen to be large enough to include both the values found in conventional, noisy wind tunnels as well as in flight tests. Possible surface deformations that might trigger the boundary transition are not considered.

3.2 Results

For this case study, the UQ analysis was performed for the wing profile at the root of the wing of the SpaceLiner 8 Booster stage, as described in [33]. In the following, only the bottom side of the wing is considered, as it sees the larger thermal loads.

In total, the uncertainty space was sampled 11264 times for this exemplary case study. Figure 7 shows the resulting reentry trajectories. The therein presented estimation for the heat flux at the stagnation point is derived for a reference nose radius of 0.5 m with a modified Chapman equation:

$$\dot{q} = 20254.4\mathrm{W/cm^2} \cdot \sqrt{\frac{q}{\rho_R}\frac{R_{\mathrm{nose,ref}}}{R_{\mathrm{nose}}}} \left(\frac{v}{v_{\mathrm{ref}}}\right)^{3.05} \tag{1}$$

$R_{\mathrm{nose,ref}}$ is the reference nose radius (1 m), R_{nose} the vehicle nose radius, v the vehicle's velocity, and v_{ref} a reference velocity of 10000 m/s. It can be seen that the selected uncertainty distribution leads to a fairly broad range of initial conditions and, subsequently, also heat fluxes encountered throughout the trajectory. For each of the trajectory samples shown in Fig. 7, an aerothermal database is derived. As the local heat flux depends on the local wall temperature, the calculations are repeated for four different wall temperatures. This makes it possible to compute the current heat flux by interpolation, using the current wall temperature, during the thermal simulation for the TPS design.

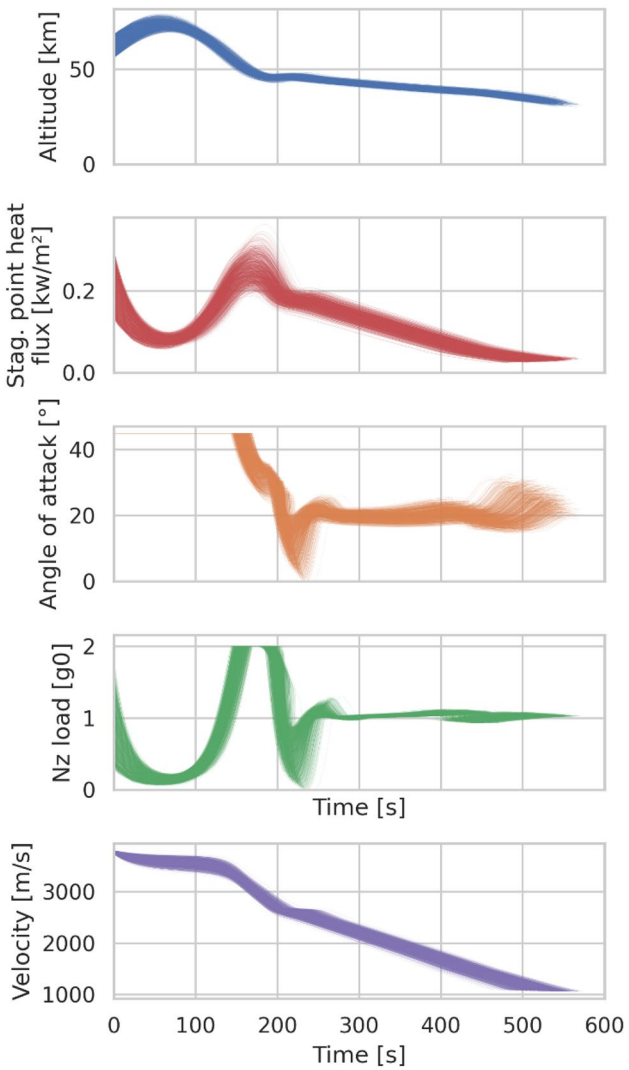

Fig. 7 Reentry trajectory variation resulting from uncertainties in the initial conditions. A subset consisting of 10% of the evaluated samples is shown

Figure 8 shows the wing profile and Fig. 9 shows the heat flux along the x-axis of this profile resulting from the trajectory variations. The transition from laminar to turbulent flow can be seen in the sudden jumps in heat flux between two trajectory points. They are not evenly distributed, since the aerothermodynamic database is not evaluated at every instance of the trajectory due to the associated computational cost. Instead, the evaluation times are evenly distributed along the trajectory and expanded at critical time points such as the time of maximum stagnation point heat flux. Therefore, the transition from laminar to turbulent flow at any location can only happen between two timesteps at which the aerothermal database is evaluated. In this case, the maximal timestep size was set to 10 s.

The dips in heat flux seen at approx. 220 s are likely caused by a temporary reduction in the angle of attack,

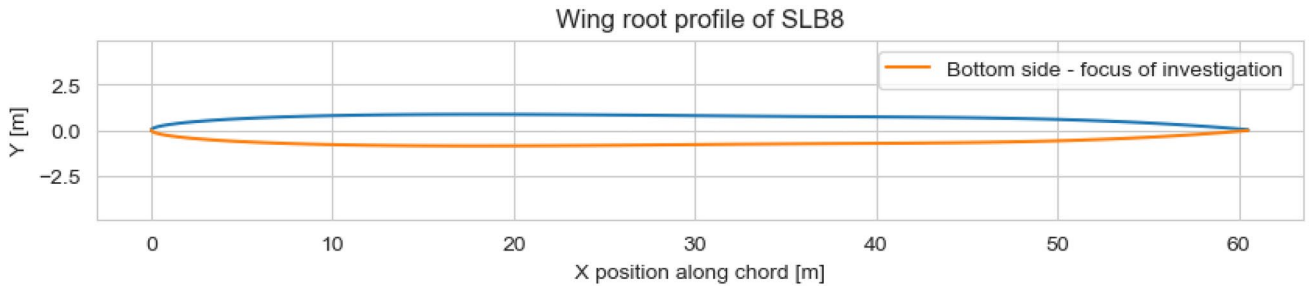

Fig. 8 Wing root profile of the Space Liner 8 vehicle (SLB8). The bottom side, the focus of this case study, is highlighted

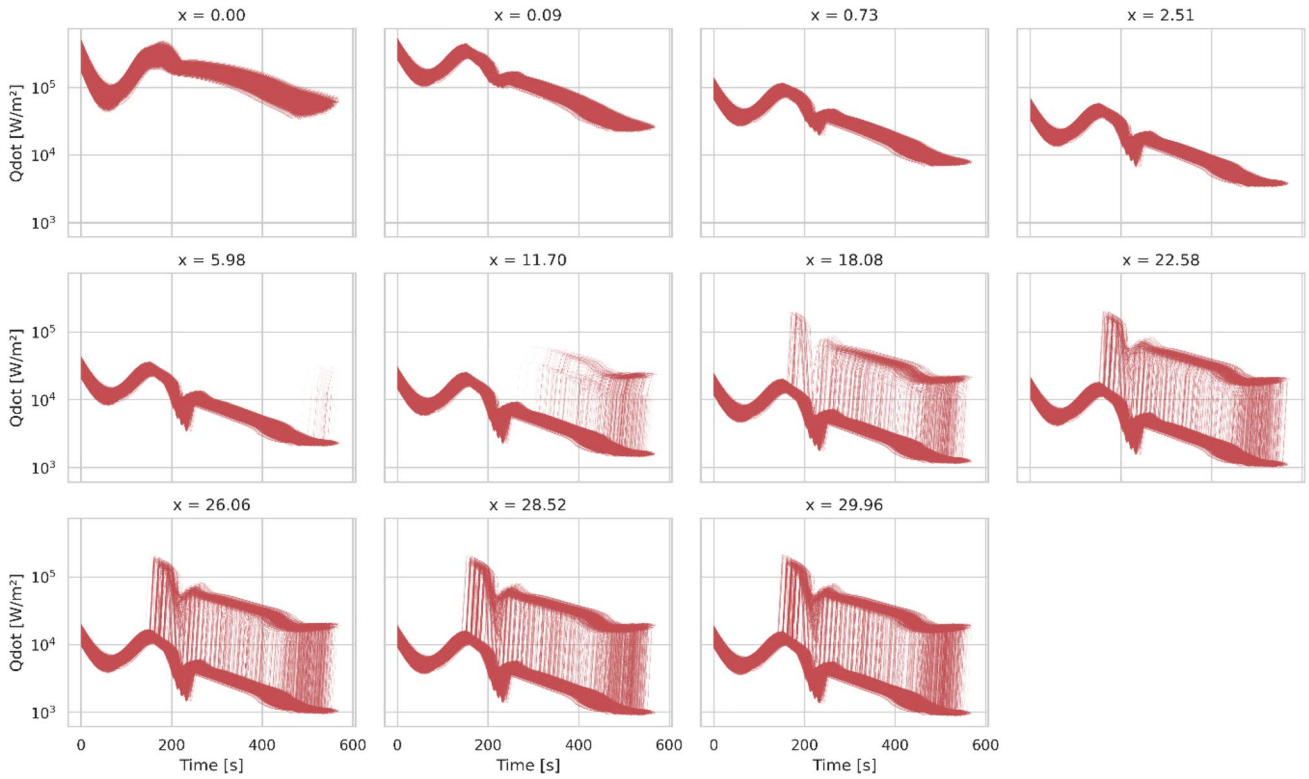

Fig. 9 Variability of heat flux over time and position along wing profile. Values assuming a wall temperature of 100 K are shown

which can also be seen in Fig. 7. This leads to temporary lower heat flux on the rearward portions of the wing, as it is less exposed to the incoming flow.

The final objective of this case study was the determination of the required TPS thickness to keep the temperature of the structure below 400 K. In this case, an idealized minimal insulation thickness was assessed along the wing chord. In reality, only a limited number of different thicknesses would be used to reduce production and maintenance effort. Figure 10 shows the resulting minimal thickness of the insulation layer of the chord of the wing root.

As expected, the TPS needs to be thickest at the stagnation point, which is in line with the heat load distribution. While the average values follow a clear trend, a number of outliers, including the worst-case scenario, exhibit significantly increased TPS thickness in the rear half of the wing profile.

For the vehicle design, the local TPS thickness is of less importance than the total TPS mass. To assess that, the total insulation cross-section along the wing profile was calculated and the resulting values are shown in Fig. 11. The aforementioned outliers are also visible here. While most samples result in a TPS cross-section of approx. 0.2 m^2, the distribution is strongly skewed by the outliers, which result in significantly higher cross-sections.

To underline the motivation for assessing the system performance within an uncertainty framework, Fig. 10 and Fig. 11 also show the theoretically possible worst case. In

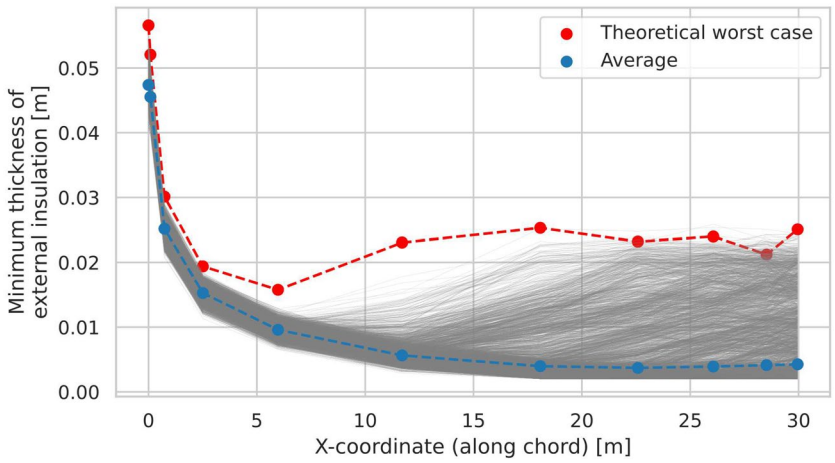

Fig. 10 Minimal TPS insulation thickness of wing chord. Average values and range of all 11264 samples are shown alongside the theoretical worst case

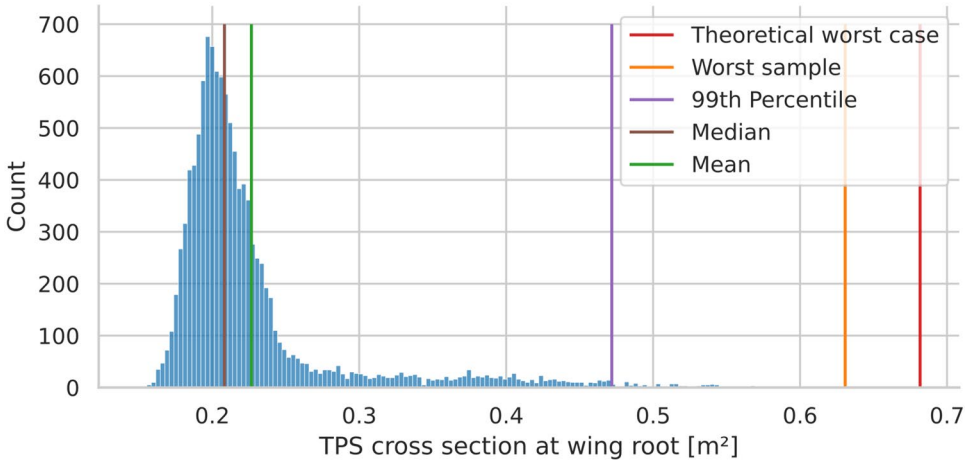

Fig. 11 TPS cross-section data for all 11264 samples and theoretical worst case. Result distribution is highly skewed by a group of outliers with very high values

that case, every uncertainty parameter was chosen to result in the highest TPS cross-section possible. This value corresponds to a conservative design approach. A significant difference can be seen between the results of this worst case (0.68 m^2), the worst evaluated sample (0.63 m^2), as well as the 99th percentile (0.47 m^2).

The cause behind the outliers can be identified when examining the correlation between the input variables and the resulting TPS cross-section, as shown in Fig. 12. It can be seen that if the transition Reynolds number is assumed to be underneath a critical value, which appears to be less than 2e7, it has a major impact on the thickness of the TPS. Above this value, the trajectory results in a fully laminar boundary layer in the hypersonic portion of the flight, consequently the value has no further impact.

3.3 Sensitivity analysis

In total ten uncertain parameters were modelled in the case study given above. To quantify the effect on the outputs (both the local TPS thickness as well as the total cross-section area), a Sobol Analysis was conducted using the implementation of the SALib library [21].

Figure 13 shows the total effects indices with regard to the TPS cross-section along the wing profile. It is clearly apparent that the uncertainty in the transition Reynolds number accounts for most of the variance in the output values. As mentioned above, this affects only a portion of the dataset, but in these cases the effect is very significant. The uncertainties in the heat flux, initial FPA, reentry mass, and aerodynamic reference area also have noticeable effects, although the magnitudes are smaller.

Figure 14 also shows the total effect indices but this time for the local TPS thickness over the wing profile. Herein some interesting contrasts to the sensitivity of the global results appear. The transition Reynolds number has no effect on the local TPS thickness for the first few meters of the wing profile. Even at the lower boundary, the boundary layer remains laminar in all cases. Instead, the variance in the heat flux, reference area, initial FPA, and initial altitude affect the TPS thickness. Even though the variation of the initial velocity is an order of magnitude smaller than

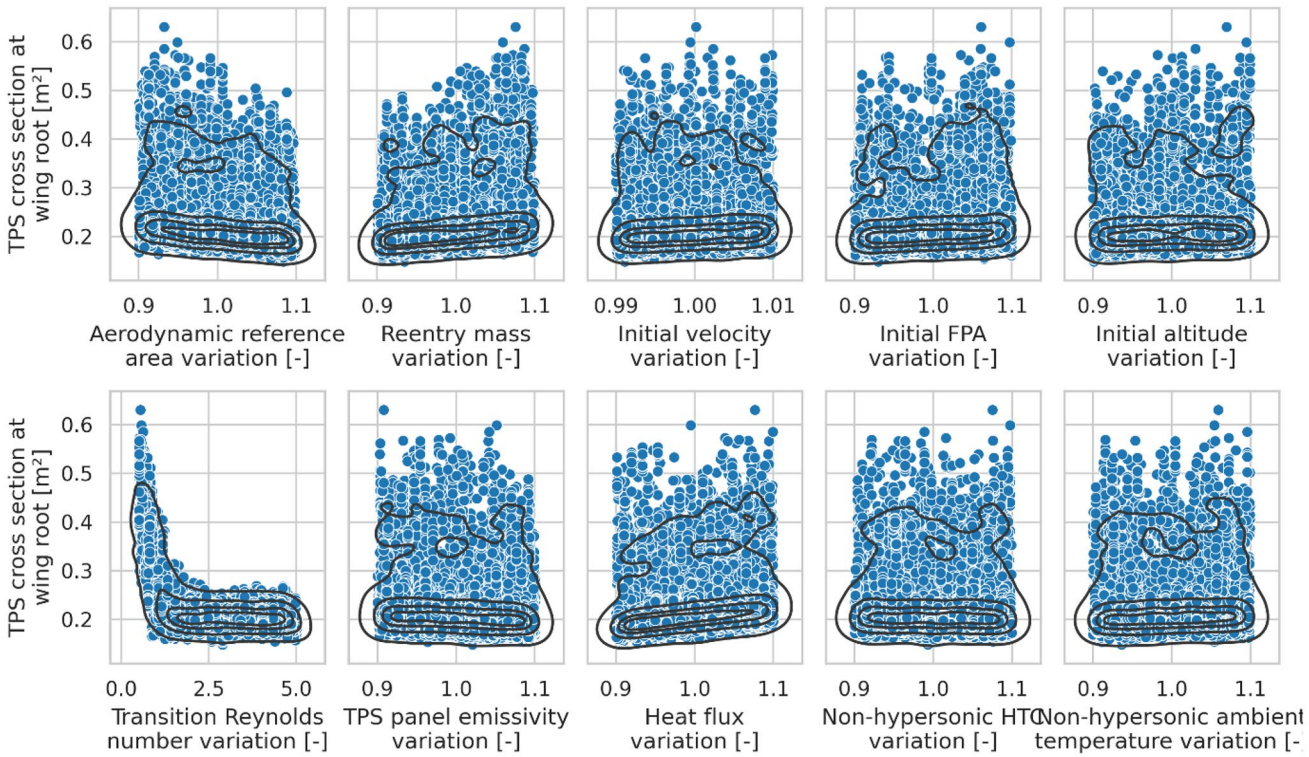

Fig. 12 Results for TPS cross-section plotted over the ten considered input uncertainties as defined in Table 1

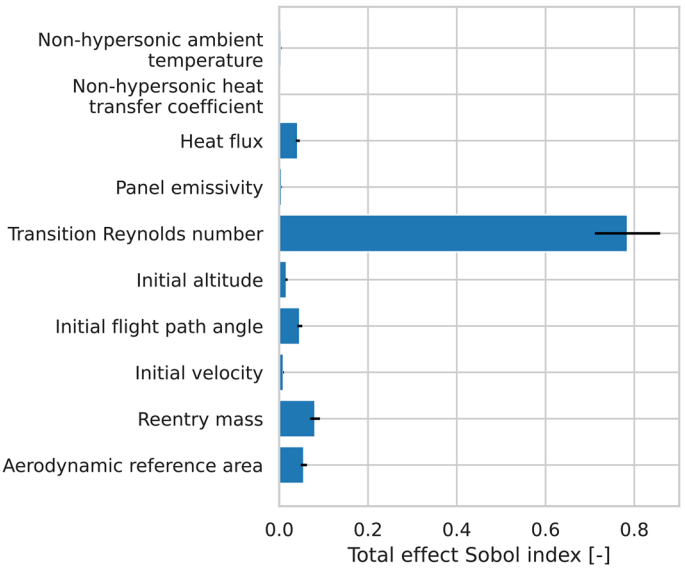

Fig. 13 Total effect sensitivity indices with regard to TPS cross-section. Error bars indicate confidence interval

that of the other initial conditions, it still has a roughly equal effect on the outcome. At the stagnation point, the hottest point, the initial FPA has the largest impact. The front of the wing is also the only portion where the panel emissivity plays a role. Here, the temperatures of the TPS surface are highest and therefore the radiation heat transfer plays the largest role.

3.4 Application of UQ data model

The entire size of the input and output data, generated by this study and stored as a Python pickle file, is 7.8 GB. From this data set a UQ data model was generated. The respective XML file is provided as Online Resource 2. The UQ data model contains a single study with ID `studySobol`. All

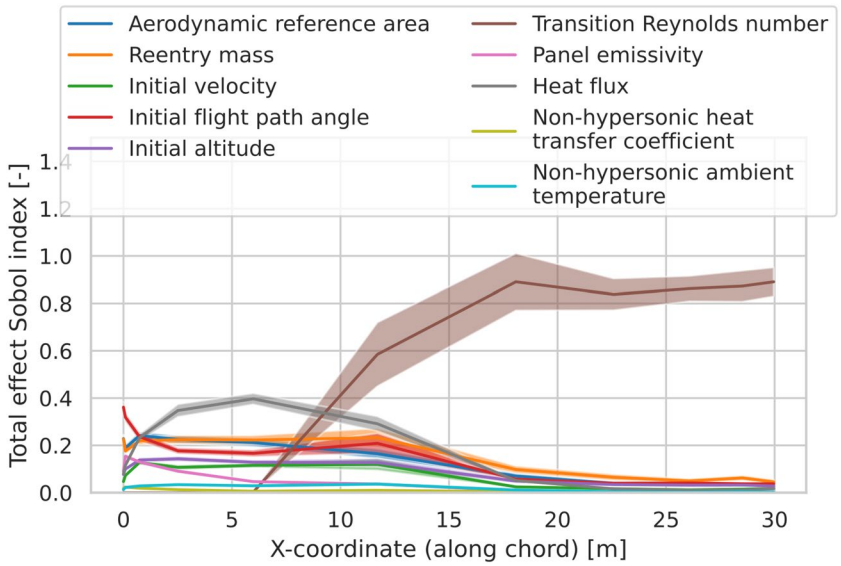

Fig. 14 Total effect sensitivity indices for local TPS thickness for different points along the wing profile including confidence intervals

ten uncertain inputs, listed in Table 1, along with the TPS cross-section as an output, were extracted and stored in the model. The cross-section was selected because it serves as the main performance metric of the TPS system design. It is correlated with the mass of the TPS, and thus, has a direct influence on the RLV's system performance. The study contains a total of 11264 samples. In addition, the mean, variance, and Sobol indices for the output `outCrossSection` are included. In sum, about 124000 double-precision floating-point values are stored in the XML file. The resulting file size is about 2.2 MB.

Other outputs that are important when looking specifically at the design of the TPS subsystem are the ones evaluated in Figs. 7, 9, and 10. These encompass altitude, stagnation point heat flux, load factor in z-direction, angle of attack, and velocity as functions of time, plus TPS insulation thickness as a function of the x-coordinate along the chord, and finally heat flux as a function of both time and x-coordinate. However, the presented version of the UQ data model only supports scalar variables and its future extension to multidimensional variables is addressed in the last paragraph of Sec. 4.

4 Discussion of UQ data model and future extensions

The previous section provides an example for the application of the UQ data model to the probabilistic study of a single TPS system. This use case serves as a proof-of-concept, and despite its small size, helps to highlight certain benefits and limitations. First, the UQ data model functions as a single source of truth. It simplifies the storage of all data related to the probabilistic design study by consolidating it into a single file. This file can then be used for postprocessing steps such as visualization, as well as for data exchange and the archiving of study results. The example also showcased that even a small probabilistic study generates a large volume of data and that the purposeful selection of the relevant probabilistic data significantly reduces that amount. Employing the UQ data model for this purpose also provides valuable guidance to the user on selecting the appropriate pieces of information because it offers a predefined structure. In addition, by using a domain-specific data file, as described in Sec. 2.2.2, the input files for specific samples can be restored and rerun. This is a functionality supported by the software library described in Sec. 2.3. This feature was not utilized here because the launch vehicle data model, CLAVA, does not yet cover RLV-specific design tasks, like the TPS subsystem design.

The benefits of a standardized data model become even more apparent when considering the adoption of probabilistic methods on a larger scale than presented herein. This could mean more complex use cases, trade studies, or the participation of several institutions in a study. In this context, a complex use case could mean the probabilistic design of an entire RLV, with the TPS just being a subsystem. Such an approach would necessitate the deliberate selection of essential data for the probabilistic assessment, due to the sheer amount of data. Further, recent European RLV trade studies, like ENTRAIN [34] and SYST-RLV [35], compare different vehicle staging and stage return concepts. As the analyzed variants can differ substantially from each other, quantifying the uncertainty of the performance metrics, like the vehicle's gross-lift-off mass, adds a layer of information that is valuable to decision makers. A decrease in performance might be acceptable when paired with a reduction in performance uncertainty. Analyzing and visualizing the

results for the different vehicles is facilitated by standardized data sets, as the same tools can be used for processing the results of several configurations.

Further, applying a standardized approach for handling probabilistic data is particularly advantageous in a collaborative setting, where experts from different disciplines are involved. The UQ data model is domain independent, and therefore tools developed within that context can be used by everyone. The data model helps to ensure data consistency and enables automation of the entire probabilistic design process, (s. Fig. 1), because it was devised with this particular process in mind. It also facilitates data exchange between partners. A disadvantage of such an approach is that collaboration is needed so that all involved parties keep adhering to the same standard. Sharing a software implementation of the data model helps to ensure this, as ideally the software library will enforce correct usage. In addition, a library makes consistency checks possible that go beyond simple compliance with the schema. Both, a common schema and a related software library, entail an additional effort but the wide adoption of CPACS in aeronautics demonstrates that the advantages of this approach substantially outweigh the effort [36].

As described in Sec. 2.4, the UQ data model is the central part of the UQ framework under development at DLR. Within this context, it is used to store and handle the probabilistic data. In addition, the proposed UQ data model can be used independently from such a framework. The minimal requirement for effectively using an XSD-based format is the installation of an XML editor, like the freely available Eclipse XML Editors and Tools [37].

Within the PROTEKT project, the first tests were performed using much larger studies, and the data model approach presented herein worked as intended. Still, at some point the total amount of data will become an issue for two reasons. First, the size of the XML file will become unreasonably large, as the data is stored using the American Standard Code for Information Interchange (ASCII). Secondly, when working with the data model, it is fully kept in memory to ensure good runtime performance; however, Random Access Memory (RAM) is a much more limited resource compared to hard drive space. Therefore, trials were performed where everything contained in the `data` element belonging to a `study` was stored using the Hierarchical Data Format version 5 (HDF5) instead. HDF5 has two advantages. First, storing double-precision floating-point values requires less storage in a binary than in a text format. Secondly, it natively supports matrix data types. As the combination of two data sources was handled by the UQ library, the user experience stayed the same. However, maintaining consistency when employing two files adds additional challenges. The UQ library has to ensure that both files stay in sync and the `data` element exists either in the XML or the HDF5 file and not in both at the same time. So far, these aspects have not yet been fully addressed in the current implementation of the UQ library.

At the moment, the UQ data model presented herein provides the basic functionality required to handle the data involved in probabilistic studies, as shown in Fig. 1. There are several areas for future extension of the model. As of now, the information included in the `sampling` element is quite limited. It is therefore planned to replace the current sampling `method` element, which is now a simple string, with a complex type that contains the various parameters of that method. Next, the use case in Sec. 3 produces trajectory results, e.g., altitude over time. At present, the UQ data model only supports single-value quantities. As discussed in Sec. 2.2.2, multidimensional data can be saved in XML by flattening it into a one-dimensional array. If the shape of the original data cannot be inferred from other information stored in the model, then an additional shape attribute needs to be added to the respective element. Another potential extension relates to surrogate models. The execution time of a single MDA(O) run in the presented case study is about two minutes on a single core. When this time increases considerably and is combined with a high number of evaluations, then Monte-Carlo-based UQ studies become only possible with the help of surrogate models. It might therefore be useful to store information on the surrogates used in such studies in the UQ data model to adhere to the idea of a single source of truth.

5 Conclusion and outlook

A UQ data model that facilitates probabilistic design studies was presented. Its structure was developed based on the probabilistic design process and its purpose is to store the data associated with this process. The schema that describes the model's hierarchical structure was implemented using XSD. As a proof-of-concept, the data model was applied to a case study addressing the uncertainty-based TPS sizing of the wing of an RLV stage. This use case demonstrated that the proposed model provides the basic functionality for aggregating the data that is associated with the probabilistic design process into a single data repository. The XML file that stores the data then functions as a single source of truth and can be used as an input for postprocessing, data exchange, and storage of study results. In addition, the structure of the UQ data model guides users in the selection of the appropriate data, and thus significantly reduces the amount of stored data.

The case study also revealed a current limitation of the model when it comes to storing multidimensional data that are, e.g., produced as part of trajectory simulations. This important point will be addressed in the next iteration of

the UQ data model because our long-term goal is to perform an uncertainty-based design of a full launch vehicle. Other future features concern increasing the level of detail of certain elements, employing the binary format HDF5, and adding information on surrogate models.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12567-025-00597-9.

Author contributions A.K. and M.A. designed the UQ data model. A.K. wrote the UQ library. J.W. performed the case study. Section 1 was written by M.A. with contributions from A.K. Section 2 was written by A.K. with contributions from M.A. M.A. prepared Fig. 1, while A.K. prepared Figs. 2, 3, 4, 5, 6. J.W. prepared Sect. 3, including Table 1 and Figs. 7, 8, 9, 10, 11, 12, 13, 14. The exception is Sect. 3.4, which was contributed by A.K. A.K wrote Sects. 4 and 5 and prepared online resources 1 and 2. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability Data is provided within the supplementary information files.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Blair, J.C., Ryan, R.S., Schutzenhofer, L.A., Humphries, W.R.: Launch vehicle design process: characterization, technical integration, and lessons learned. NASA, TP-2001-210992 (2001)
2. Yao, W., Chen, X., Luo, W., Tooren, M., Guo, J.: Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles. Prog. Aerosp. Sci. **47**(6), 450–479 (2011). https://doi.org/10.1016/j.paerosci.2011.05.001
3. King, J.M., Grandhi, R.V., Benanzer, T.W.: Quantification of epistemic uncertainty in re-usable launch vehicle aero-elastic design. Eng. Optim. **44**(4), 489–504 (2012). https://doi.org/10.1080/0305215x.2011.588224
4. Sobol', I.M.: Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Math. Comput. Simul. **55**(1–3), 271–280 (2001). https://doi.org/10.1016/s0378-4754(00)00270-6
5. Alder, M., Ahmed, T., Fröhler, B., Skopnik, A.: Assessment of techniques for global sensitivity analyses in conceptual aircraft design. In: AIAA AVIATION 2023 Forum. AIAA, San Diego, CA and online (2023). https://doi.org/10.2514/6.2023-3696
6. Thunnissen, D.P., Au, S.K., Swenka, E.R.: Uncertainty quantification in conceptual design via an advanced Monte Carlo method. J. Aerosp. Comput. Inf. Commun. **4**(7), 902–917 (2007). https://doi.org/10.2514/1.28307
7. Brevault, L., Balesdent, M., Morio, J.: Aerospace system analysis and optimization in uncertainty. Springer Optimization and Its Applications, Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39126-3
8. Smith, N., Mahadevan, S.: Integrating system-level and component-level designs under uncertainty. J. Spacecr. Rockets **42**(4), 752–760 (2005). https://doi.org/10.2514/1.6662
9. Du, X., Chen, W.: Sequential optimization and reliability assessment method for efficient probabilistic design. In: Volume 2: 28th Design Automation Conference. ASME, Montreal, Quebec (2002). pp. 871–880. https://doi.org/10.1115/detc2002/dac-34127
10. Freitas, C.J.: Standards and methods for verification, validation, and uncertainty assessments in modeling and simulation. J. Verif. Valid. Uncert. (2020). https://doi.org/10.1115/1.4047274
11. Fischer, P.M., Deshmukh, M., Koch, A., Mischke, R., Martelo Gomez, A., Schreiber, A., Gerndt, A.: Enabling a conceptual data model and workflow integration environment for concurrent launch vehicle analysis. In: 69th International Astronautical Congress. IAF, Bremen, Germany (2018)
12. Alder, M., Moerland, E., Jepsen, J., Nagel, B.: Recent advances in establishing a common language for aircraft design with CPACS. In: Aerospace Europe Conference. 3AF, Bordeaux, France (2020)
13. Harries, S., Abt, C.: Integration of tools for application case studies. In: Papanikolaou, A. (ed.) A holistic approach to ship design: application case studies vol 2, pp. 7–45. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-71091-0_2
14. van Gent, I., La Rocca, G., Hoogreef, M.F.M.: CMDOWS: a proposed new standard to store and exchange MDO systems. CEAS Aeronaut. J. **9**(4), 607–627 (2018). https://doi.org/10.1007/s13272-018-0307-2
15. Fallside, D.C., Walmsley, P.: XML Schema Part 0: Primer Second Edition. World Wide Web Consortium (W3C). https://www.w3.org/TR/xmlschema-0/. Accessed: 2024-07-29 (2004)
16. Keller, R.M.: Ontologies for aviation data management. In: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference. IEEE, Sacramento, CA (2016). https://doi.org/10.1109/dasc.2016.7777971
17. Decker, S., Melnik, S., Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., Horrocks, I.: The Semantic Web: the roles of XML and RDF. IEEE Internet Comput. **4**(5), 63–73 (2000). https://doi.org/10.1109/4236.877487
18. Jepsen, J., Zamfir, A., Boden, B., Zamboni, J., Moerland, E.: On the development of a collaborative knowledge platform for engineering sciences. In: Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, pp. 208–215. SciTePress, Rome. (2023). https://doi.org/10.5220/0012189600003598
19. Lambe, A.B., Martins, J.R.R.A.: Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. Struct. Multidisc. Optim. **46**(2), 273–284 (2012). https://doi.org/10.1007/s00158-012-0763-y
20. Clark, J., DeRose, S.: XML Path Language (XPath). Version 1.0. World Wide Web Consortium (W3C). https://www.w3.org/TR/1999/REC-xpath-19991116/. Accessed: 2024-04-05 (1999)
21. Herman, J., Usher, W. (2017) SALib: An open-source Python library for sensitivity analysis. J. Open Source Softw. **2**(9), 97. https://doi.org/10.21105/joss.00097
22. Python Software Foundation: xml.etree.ElementTree — The ElementTree XML API. https://docs.python.org/3.12/library/

xml.etree.elementtree.html. Version 3.12.8, Accessed: 2024-07-30 (2024)
23. Abrahams, D., Seefeld, S.: Boost.Python. https://www.boost.org/doc/libs/1_83_0/libs/python/doc/html/index.html. Version 1.83.0, Accessed: 2023-09-29
24. Abrahams, D., Grosse-Kunstleve, R.W.: Building hybrid systems with Boost. Python. C/C++ Users J. **21**(7), 29-36 (2003)
25. Kapoulkine, A.: pugixml. https://pugixml.org. Version 1.13, Accessed: 2023-05-12
26. Gao, S., Sperberg-McQueen, C.M., Thompson, H.S.: W3C XML Schema Definition Language (XSD) 1.1 Part1: Structures. https://www.w3.org/TR/xmlschema11-1. Accessed: 2024-07-30 (2012)
27. Reisch, U., Streit, T.: Surface inclination and heat transfer methods for reacting hypersonic flow in thermochemical equilibrium. In: Körner, H., Hilbig, R. (eds.) New Results in Numerical and Experimental Fluid Mechanics. Vieweg+Teubner Verlag, Wiesbaden pp. 267–274 (1997). https://doi.org/10.1007/978-3-322-86573-1_34
28. Anderson, J.D., Jr.: Hypersonic and high-temperature gas dynamics. AIAA Education Series (2006). https://doi.org/10.2514/4.861956
29. Zhang, C., Zhu, Y., Chen, X., Yuan, H., Wu, J., Chen, S., Lee, C., Gad-el-Hak, M.: Transition in hypersonic boundary layers. AIP Adv. (2015). https://doi.org/10.1063/1.4935019
30. Lee, C., Chen, S.: Recent progress in the study of transition in the hypersonic boundary layer. Natl. Sci. Rev. **6**(1), 155–170 (2018). https://doi.org/10.1093/nsr/nwy052
31. Zhu, Y., Chen, X., Wu, J., Chen, S., Lee, C., Gad-el-Hak, M.: Aerodynamic heating in transitional hypersonic boundary layers: role of second-mode instability. Phys Fluids. (2018). https://doi.org/10.1063/1.5005529
32. Chen, F.-J., Malik, M.R., Beckwith, I.E.: Boundary-layer transition on a cone and flat plate at Mach 3.5. AIAA J. **27**(6), 687–693 (1989). https://doi.org/10.2514/3.10166
33. Sippel, M., Stappert, S., Bayrak, Y.M., Bussler, L.: Systematic assessment of SpaceLiner passenger cabin emergency separation using multi-body simulations. CEAS Space J. **15**(6), 797–811 (2023). https://doi.org/10.1007/s12567-023-00505-z
34. Dietlein, I., Bussler, L., Stappert, S., Wilken, J., Sippel, M.: Overview of system study on recovery methods for reusable first stages of future European launchers. CEAS Space J. (2024). https://doi.org/10.1007/s12567-024-00557-9
35. Iranzo-Greus, D., Deneu, F., Le-Couls, O., Bonnal, C., Prel, Y., Guedron, S.: Selection and design process of TSTO configurations. In: 54th International Astronautical Congress 2003, Bremen. (2003). https://doi.org/10.2514/6.IAC-03-V.4.05
36. Alder, M., Skopnik, A.: Development of a software library for performant and consistent CPACS data processing. In: Deutscher Luft- und Raumfahrtkongress. DGLR, Dresden. (2022)
37. IBM Corporation: Eclipse XML Editors and Tools. Eclipse Foundation. Accessed: 2024-12-04 (2017). https://marketplace.eclipse.org/content/eclipse-xml-editors-and-tools