


Article

# Towards Hierarchical Workflows in SysML to Support Virtual Validation of Technical Systems

Yizhe Zhang , Georg Jacobs , Gregor Hoepfner  and Joerg Berroth 

Institute for Machine Elements and Systems Engineering, RWTH Aachen University, 52062 Aachen, Germany; georg.jacobs@imse.rwth-aachen.de (G.J.); gregor.hoepfner@imse.rwth-aachen.de (G.H.); joerg.berroth@imse.rwth-aachen.de (J.B.)

\* Correspondence: yizhe.zhang@imse.rwth-aachen.de

**Abstract:** Innovative Model-Based Systems Engineering (MBSE) applies function-oriented hierarchical system architecture and utilizes Systems Modeling Language (SysML) for virtual testing. However, for complex systems, the relevant virtual tests are scattered at different hierarchy levels. Manually performing these virtual tests requires a lot of effort and leads to the potential risk of errors due to the overlooking of some tests and functions. In order to solve these problems, it is necessary to develop automated virtual validation workflows for the function-oriented system architecture. This contribution proposes a standardized virtual validation workflow design framework corresponding to the hierarchical functional architecture to organize virtual tests. The virtual tests are also modeled in workflows consisting of a set of simulation activities that can execute domain models to simulate system behaviors. The simulation activities are developed modularly, corresponding to the classification of the domain models. The resulting workflows are implemented in a wind turbine (WT) system. It demonstrates that the workflows enable automated validation at all hierarchy levels and early detection of technical system failure risks. The design framework allows for the synchronous creation of validation workflows as functions are added or decomposed. The standardized design ensures easy redesign and function reuse across different systems. Modular design, based on model classification, enhances design agility and adaptability in various system contexts. The proposed virtual testing workflows automatically execute corresponding simulation activities sequentially.

**Keywords:** model-based systems engineering; wind turbine; function-oriented system architecture; virtual validation workflow



**Citation:** Zhang, Y.; Jacobs, G.; Hoepfner, G.; Berroth, J. Towards Hierarchical Workflows in SysML to Support Virtual Validation of Technical Systems. *Appl. Sci.* **2023**, *13*, 5122. <https://doi.org/10.3390/app13085122>

Academic Editors: Arkadiusz Gola, R.M. Chandima Ratnayake and Martin Krajcovic

Received: 17 March 2023

Revised: 16 April 2023

Accepted: 18 April 2023

Published: 20 April 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

As multidisciplinary technical systems become increasingly in scale and complexity, the task of developing these systems to meet the constantly evolving and unique market demands grows more and more difficult [1]. Taking the wind turbine (WT) system as an example, in 2021, wind power generation increased by a record 273 TWh (17% growth), which is the highest of all renewable energy technologies [2]. In addition, for WT manufacturers to remain competitive, the development of WTs requires lower wind energy utilization costs and shorter development cycles, while coping with engineering requirements to ensure the reliability of the system [3].

To address these challenges mentioned above, an advanced system engineering approach must be adopted. Systems Engineering (SE) is a promising and interdisciplinary systems development approach that is widely recognized as a preferred solution to meet the challenge associated with the development of new systems or modifications of complex systems [4]. For many years, engineers have used domain models which virtually describe the behavior of real-world systems across multiple disciplines for cost-effective development. These standalone model islands have been formalized into the SE approach, using the so-called Model-Based Systems Engineering (MBSE) approach. The MBSE approach

focuses on models rather than documents as the primary mean of information exchange for the management of complexity, synchronizing, and maintaining consistency [5]. In contrast to documents generated based on natural language, digital domain models have more rigorous semantics, which provides the possibility for standardized modeling. Based on standardized modeling, the MBSE approach can support the consistent design and bridge the gaps between multiple disciplines, reducing information exchange time while ensuring complex systems meet the requirements and handling trade-offs between requirements. One of the practical MBSE methods is proposed by Jacobs [6] for function-oriented system development. The method adapts the Motego as a modeling approach to support the function-oriented development of technical engineering systems (e.g., WT systems). Although Motego is on its way to combining system models with domain models to simulate the system behaviors, engineers still cannot easily use SysML system models for system virtual validation. A comprehensive virtual validation process often relates to different functions at all system hierarchy levels and requires the participation of multiple virtual tests that performs the domain models in sequence. As the system becomes complex, it becomes inconvenient and unreliable to conduct these processes manually. A virtual validation workflow consists of orchestrated and repeatable activity modules that enable the virtual validation process by the systematic organization of virtual tests. Therefore, the objective of this work is to design a standardized framework based on Motego to model virtual validation workflows. The created workflows can organize various virtual tests and conduct the virtual validation of each function of the technical system. In addition, this work further standardizes the virtual tests in workflows based on simulation activities in the hierarchical system architecture. The contributions of this work are to provide an automated workflow that can be used for the virtual validation of functional requirements for multidisciplinary technical systems. The application of the proposed workflow enables engineers to reduce the manual work of virtual validation, and to achieve the reuse or redesign of the virtual validation workflows in different system contexts easily.

This paper is organized as follows: Section 2 reviews state-of-the-art and related work. Section 3 presents a current virtual validation process for a redesigned technical engineering system. Section 4 introduces the design methods of the virtual validation workflows based on the system model architecture. Section 5 evaluates the feasibility of the created workflow structure with a demonstration of virtual validation workflows of partial functions in the studied WT system. Section 6 discusses the findings, the superiority of this work, and prospects for future research. Finally, Section 7 concludes the work.

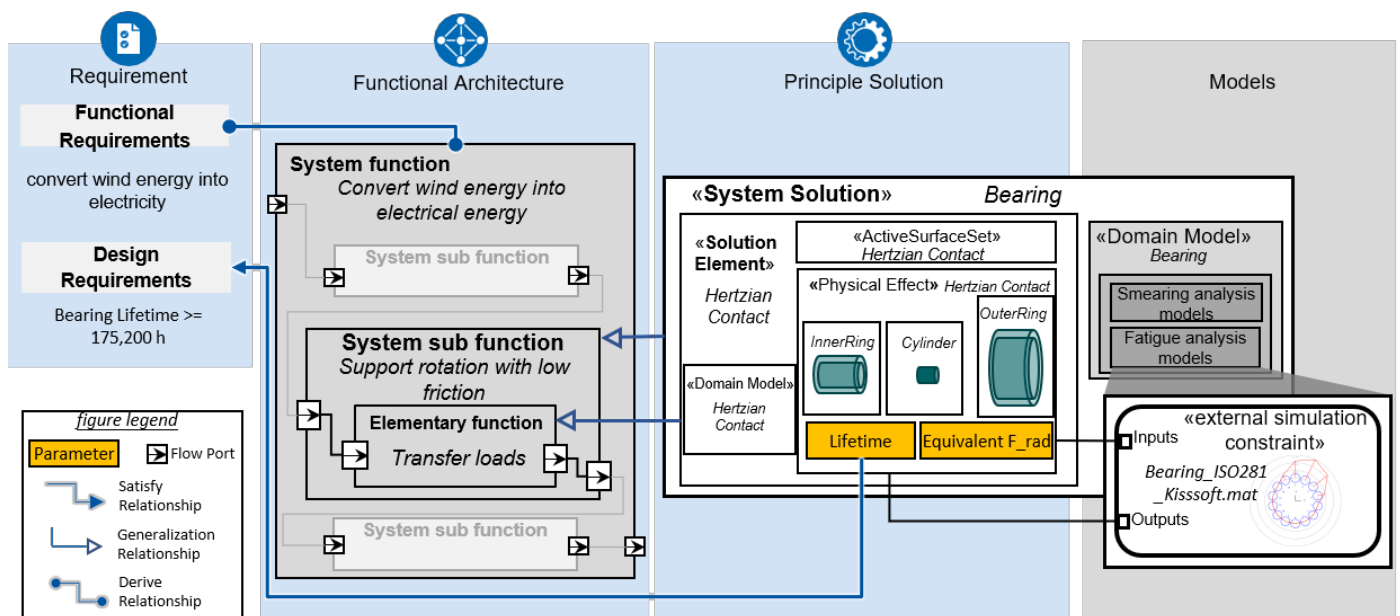
## 2. State of the Art

### 2.1. Function-Oriented System Architecture

More and more practical MBSE methods are proposed [6–8] and applied to the design and validation process of real-world technical systems, such as the telescope system [9], the aero-engine system [10], the space system [11], and the wind turbine system [12]. For this research, a more detailed and comprehensive MBSE method proposed by Jacobs et al. [6] for function-oriented system development was chosen. On the one hand, the merit of this method lies in its ability to offer transparent traceability by constructing clear relationships within the system architecture, connecting requirements, functions, and the solution layer responsible for functional fulfillment during development processes. On the other hand, this method is able to integrate domain models from diverse engineering disciplines, going beyond merely remaining at the conceptual design stage. This method lays the foundation to significantly improve the practicality of MBSE in developing technical engineering systems [6]. Motego, as a function-oriented modeling approach that provides a specific SysML profile for technical engineering systems, is introduced in the following paragraphs.

A successful system is developed to fulfill the requirements of stakeholders. Therefore, the significant task of system modeling is to collect and model these requirements before and during the development process of the system. In the Motego modeling approach [13], the requirements are mainly classified into two types: functional requirements and design

requirements. Functional requirements outline the necessary functions and behavioral aspects for a system to operate effectively [14]. In Motego, requirements are linked to other SysML elements (e.g., the value properties) through different relationships, such as satisfy and validate relationships. Functional requirements depict the desired functional behavior, which can be virtually confirmed through functional testing, drawing on corresponding solutions. Design requirements, on the other hand, are directly met by the property values present in the solutions. Functions can be derived from functional requirements. They are linked to corresponding requirements of systems through satisfy relationships. The functional architecture is in the form of a hierarchy. As shown in Figure 1, the primary function can be broken down into sub-functions, which act as components of a single higher-level function. A system function, or a portion of it, can be defined by a boundary that allows physical quantities to pass in and out as functional flows. These flows may consist of energy, materials, or signals. The function of the delimited system converts the incoming flow quantities into different outgoing flow quantities. If the transformation of the flows they represent cannot be further decomposed, functions are termed as elementary functions.



**Figure 1.** Interconnected function, solution, and domain model architecture within a system model. Adapted from [6].

Solution describes a general effect with geometry or a set of general effects that fulfills functions physically. The functions and the solutions are linked by generalization relationships. Therefore, the solutions will inherit the functional flows from the functions they fulfill. Moreover, system solutions consist of hierarchically structured sub-system solutions, with their parameters interconnected across various hierarchy levels. A system solution is referred to as a solution element if it does not physically decompose further. A solution element consists of the physical effects and more parts, such as active surface pairs. Domain models are incorporated into the system model to further describe the related solution's physical behavior. Each domain model is enclosed within a SysML constraint, featuring input and output ports that connect with external domain simulation models. The constraints are connected to the parameters of the corresponding solution. High-fidelity domain models are seamlessly linked with the structured solution to ensure changes can be propagated into the domain models. By connecting the domain model to the solution, the modeling approach establishes traceability from requirements, over functions, and solutions, to the domain models.

In order to provide engineers with the confidence that the right system solutions were built or selected, it is necessary to propose a standardized method to design executable

workflows for virtually validating whether the system solutions fulfill their intended functionalities in a specific operating environment.

## 2.2. Virtual Validation Process

The virtual validation process can be viewed as an orchestrated and repeatable set of activities. This process is able to systematically organize the simulation activities of the domain models and update critical parameters during the simulation activities. The common approaches to designing a workflow in the current state of the art in both the literature and in practice are either to design a descriptive workflow or to apply an executable workflow based on one or more independent simulation calculations to satisfy specific design requirements.

Regarding management workflow designs, such as those presented by Bretz et al. [15] and Zou [16], these workflow designs focus on describing the workflow of entire system development processes at the conceptual level. Although these workflows clearly define the developer's tasks and specific operations at each step, they do not dive into the parameter level for simulations of the physical behavior of a system and do not consider the qualitative analysis of the system. Therefore, it is still challenging for the designer to directly obtain a reliable virtual validation conclusion regarding engineering issues by executing these management workflows.

Regarding executable workflow designs, the commercial workflow builders (e.g., ModelCenter) provide a way to design virtual testing workflows at the parameter level [17]. Although these workflow designs are based on quantitative analysis, they are still not enough to judge whether the system solution fulfills the function. This is because the workflows and the function-oriented system architecture are built independently in different tools, which means the changes in the system architecture need to be propagated to the workflow design by using specific interfaces. Due to the lack of a direct link between system functions and workflows, these individual quantitative analyses are not sufficient to support virtual validation against functional requirements. To solve this problem, a set of virtual testing workflows was created within the system model based on Motego [12,18]. In Motego, there is a clear mapping between solutions and functions. Thus, the system engineer can state that the function can be satisfied by the corresponding designed solutions by testing all virtual tests in the relevant solutions. Ref. [19] proposed the structured virtual testing workflows based on the classification of domain models to increase the reusability of the workflows, so that the engineer can easily use the workflow to meet different test purposes. However, the current workflows proposed in [19] still have limitations in achieving the virtual validation process.

First of all, the current workflows cannot be easily used for virtual validation across the system function level. Specifically, functions often contain sub-functions. The failure of any sub-function may lead to failures of the entire function. Therefore, a comprehensive virtual validation of a system at different hierarchical levels is required, which means engineers do not only need to virtually validate a system solution, but also each of its sub-solutions. As system complexity and the number of functions increase, where the dependencies between system designs increase exponentially, manually managing virtual testing workflows for the virtual validation of the specific functions increases manual efforts. Secondly, virtual testing workflows employ a predetermined simulation sequence. When a solution is reused in another system, the workflows must be manually reorganized and reordered according to the system context to guarantee accurate testing. Thus, it is crucial to structure semi-automated workflows to facilitate the efficiency of the virtual verification process across the entire functional hierarchy. To achieve this goal, the virtual testing workflow should be easily adjusted and redesigned when related solutions are reused or the test purpose and accuracy demands have changed. In order to further elucidate the corresponding issues, the following section (i.e., Section 3) will demonstrate an example of a wind turbine virtual validation process to provide a more comprehensive and clear understanding.

### 3. Wind Turbine System Virtual Validation

Examination of energy scenarios reveals that wind energy is set to become increasingly vital in the future energy supply system [20]. The production of energy from wind is typically achieved by the WT system [20]. For wind turbine manufacturers to remain competitive and meet the increasing demands of customers, WT systems need to be constantly redesigned and developed to capture more wind energy. However, as complexity grows in the later stages of system development, the number of interdependencies between designs also increases significantly. Therefore, ensuring that the redesigned WT system and its sub-systems meet functional requirements is still a challenge today.

Before introducing the method of this work, this section describes a classical virtual validation process of a technical system (i.e., WT) to identify the research object. In classical virtual validation processes, developers often focus on their domains. In the case described below, it is assumed that “Engineer A” considers WT as a whole system and focuses on the annual energy production (AEP) testing of the WT, while “Engineer B” considers the development of WT component, i.e., a bearing system, and focuses on fatigue analysis related to bearing lifetime.

Figure 2 illustrates a joint development of a WT system and its components by the two engineers. It was assumed that the customer’s requirement for AEP of the WT increased from 8 GWh/year to 12 GWh/year (Step 1). In order to meet the higher power generation requirement, the WT was redesigned with larger blades, increasing length from 61.5 to 80 m, and a higher power (from 3.2 MW to 6 MW) generator (Step 2). After the WT AEP testing, the AEP of the redesigned WT increased from 8.25 GWh/year to 13.38 GWh/year. Therefore, as the relevant requirement was satisfied, it was a successful redesign by “Engineer A”. However, in highly coupled complex systems, a design change often leads to unsatisfied design requirements in other fields due to physical interdependencies. In the case of these redesigns, larger blades and higher power generators will lead to increased loads on the bearing system and shaft system, which then leads to potential failure risks (Step 3). Based on the experience of engineers, a foreseeable failure risk is that the lifetime of the bearings will be reduced and the rate of utilization of the shaft will be improved. For the bearing system, the bearing lifetime requirements are set to be greater than 20 years (i.e., 175,200 h) and the requirement of the shaft rate of utilization should be less than 1. In order to ensure the reliability of the virtual validation of the bearing, “Engineer B” needs to obtain the latest data of loads on the bearing system from “Engineer A” (Step 4). The executable workflow proposed in [19] can save these updated simulation results and transmit them to the following related tests.

After the bearing fatigue analysis, it was found that the bearing lifetime was reduced to 82,014 h, which does not meet the requirements anymore (Step 5). From the perspective of “Engineer B”, this was a failed redesign, which may have conflicted with the conclusion of “Engineer A”. As a result, “Engineer B” needed to inform “Engineer A” that there was a risk of failure in the new design; this is a process that would normally rely on manual effort rather than being seamless. In complex multi-level systems, the failure of components may lead to the failure of the overall system. Therefore, the first problem is that the current virtual validation process still lacks a systematic organization that can handle the virtual testing and verification results at different hierarchy levels to achieve comprehensive and automated virtual validation.

To fulfill all requirements, “Engineer B” needed to redesign the bearing system as well. In order to adapt to the higher loads, “Engineer B” chose bearings with larger diameters, which allow for a relatively long service life. Similarly, it was necessary for “Engineer B” to provide design information for new bearings to “Engineer A” and perform the relevant analysis (i.e., AEP analysis) again to ensure that the requirement was still satisfied. Step 6 repeats the steps from 2 to 5 until all the tests meet the design requirements. Usually, the designer has to manually and repeatedly design and execute the workflows for the virtual validation, which takes a lot of time and effort, especially if the system context changes.



Thus, the second problem is that there is a lack of a method for easy reuse when the systems are redesigned or reused in different system contexts.

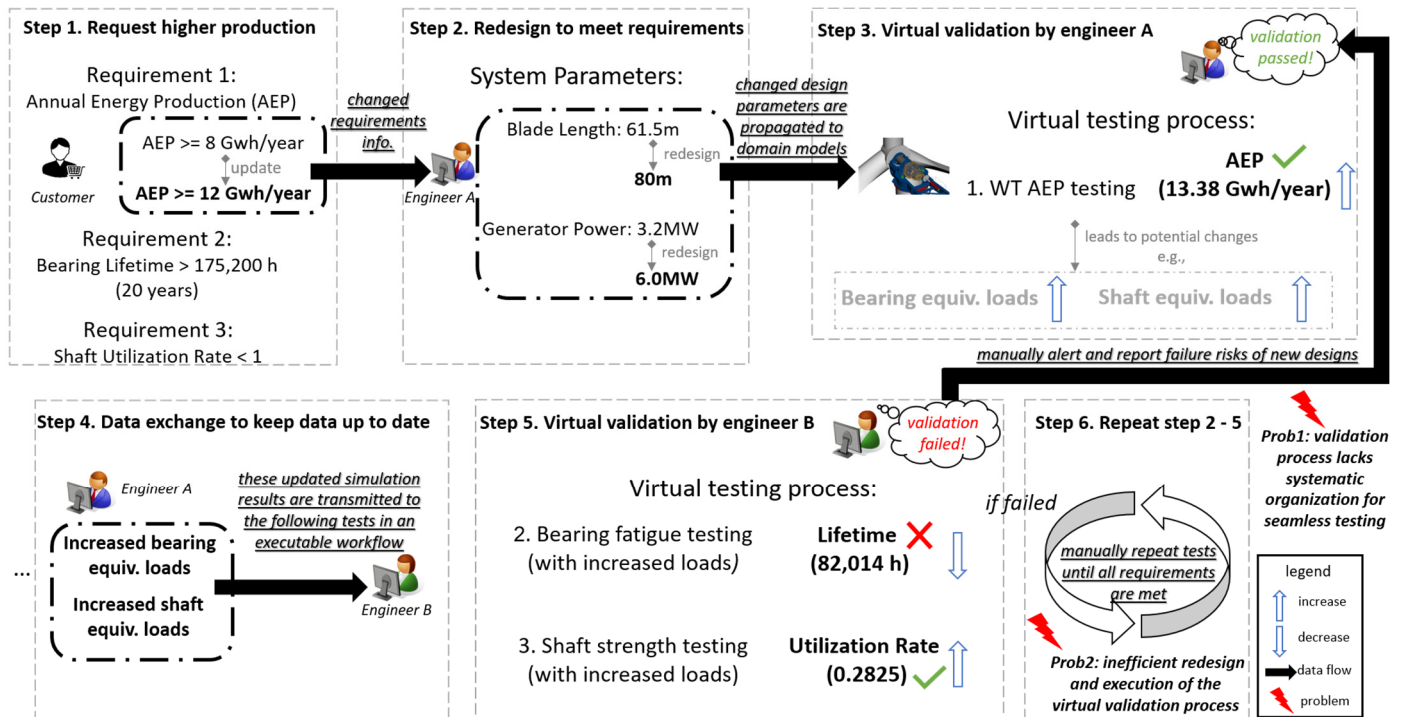


Figure 2. The classical virtual validation process of the development of a WT system and its components by the two design engineers. Adapted from [19].

In conclusion, engineers from different teams need to communicate with each other frequently for data exchange to find the balance of the WT design among the various requirements. In order to test that the WT design can meet various requirements, in the classical virtual validation process, engineers need to manually perform relevant virtual tests based on experience. However, as systems become complex and design dependencies increase, it becomes difficult to empirically define what failure analysis is required and avoid these potential design failures. At the same time, manually performing repetitive virtual tests leads to a waste of time and effort. Thus, the case demonstrates the necessity of an automated virtual validation process based on an MBSE approach that can help engineers discover the potential failure risk of complex systems after redesigns.

#### 4. Method

To implement an automated virtual validation process, this work introduces in detail how to structure the virtual validation workflows in the system model based on Motego (see Section 4.1). As part of the virtual validation workflow, the virtual testing workflow is further systematically developed in Section 4.2 according to the model classification to ensure that the workflow can be easily adapted to different system contexts.

##### 4.1. The Framework of Virtual Validation Workflows Based on the Functional Architecture

This section proposes a workflow design framework corresponding to the hierarchical functional architecture to organize virtual testing workflows scattered at various functional hierarchy levels, which helps engineers easily conduct a comprehensive virtual validation.

##### 4.1.1. Structure of Virtual Validation Workflows Based on the System Hierarchy

As the behavior of a super-function is a composition of the behavior of its sub-functions in the hierarchical functional architecture, the failure of any sub-function will result in the

failure of the super-function. Therefore, a comprehensive virtual validation of functions at different levels is required. This means that engineers not only need to test whether the system function is fulfilled by the solution, but also that each of its sub-functions is fulfilled by the corresponding sub-solution. Therefore, a nested framework is introduced in this section to formally organize the virtual validation workflows corresponding to the hierarchical functional architecture.

Figure 3 shows a framework diagram for the standardized virtual validation workflow. The virtual validation workflow framework corresponding to a function always includes four elementary actions that invoke the activities as follows:

- “Action 1” involves the activity of checking if the function is fulfilled by the corresponding solution. The specific concept and content will be introduced in detail in the next section (see Section 4.1.2).
- The implemented virtual validation workflow framework should correspond to the hierarchical structure of the functional architecture. Therefore, the upper-level virtual validation workflow should also include sub-virtual validation workflows. The invoked activity can be further decomposed into other activities that perform various sub-tasks with activity hierarchies. Therefore, “Action 2” invokes the activity in which multiple “Virtual Validation Workflow” actions of its sub-functions are organized in parallel. Validation results of sub-functions are exported and passed through pins and object flows. The action called “Check all validation results” in “Action 2” performs logic gate calculations on all validation results of a sub-function. When the validation result of any of the sub-functions is “False”, the overall validation result for its sub-function is considered to be “False”. Otherwise, the overall validation result of its sub-function is considered to be passed.
- “Action 3” invokes the activity, which validates the function by checking if the function and its sub-functions are all fulfilled by the corresponding solutions. The checking process is realized by a logic gate calculation: when the function is fulfilled by the selected solution, and the overall validation result of its sub-function is passed, the validation result of the function is considered to be passed. Otherwise, it failed.
- “Action 4” invokes the activity to save the validation result in the system model architecture. As mentioned in “Action 2”, this result is also used to support the virtual validation of its parent function.

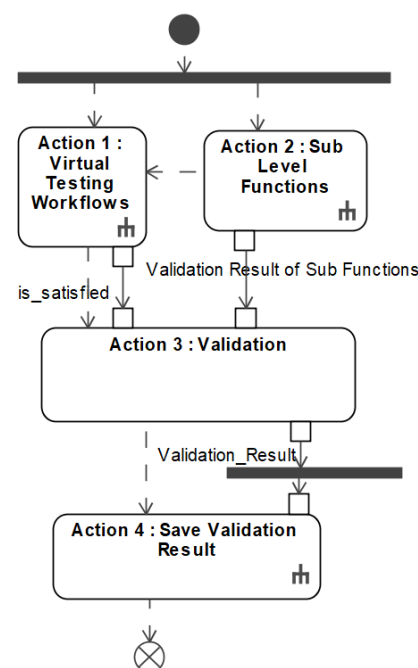


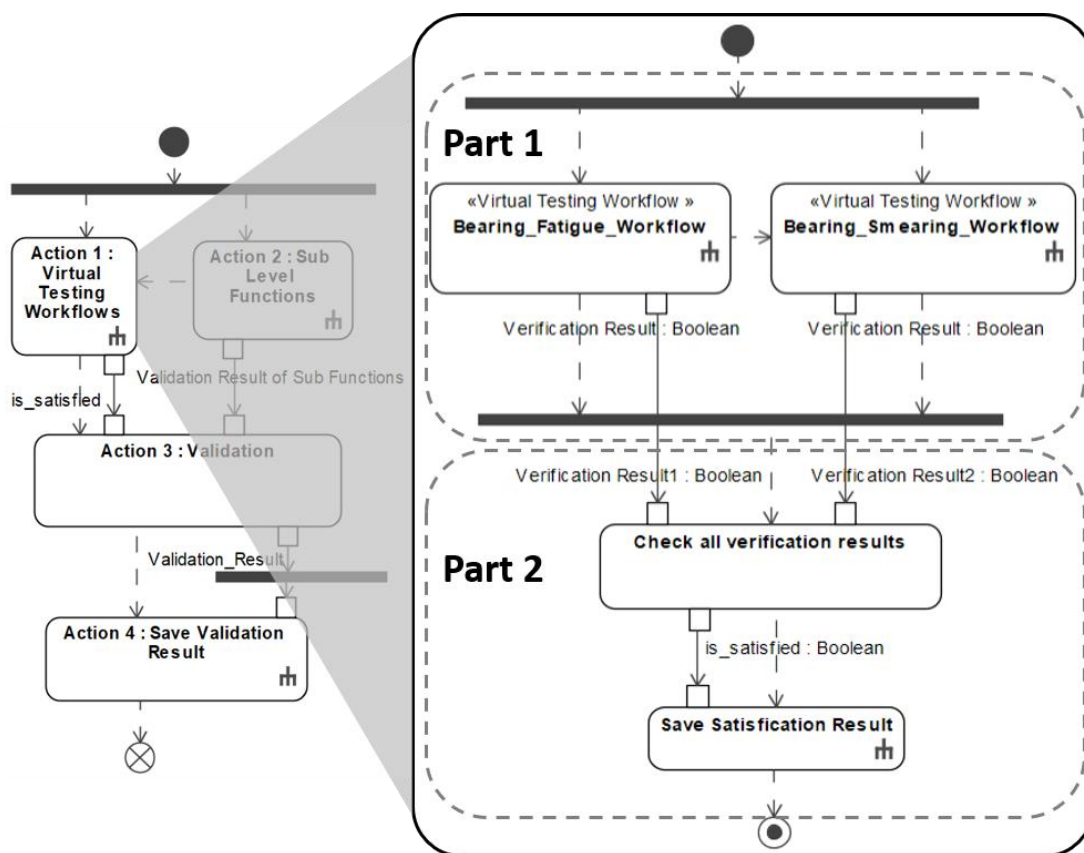
Figure 3. A virtual validation workflow framework.

#### 4.1.2. Structure of Virtual Testing Workflows Based on Domain Model Classification

As mentioned in Section 4.1.1, “Action 1: Virtual Testing Workflows” aims to check if the function is fulfilled by the corresponding solution. This section looks into the structure of “Action 1”, describing in detail how to arrange the virtual testing workflows to achieve this aim.

As demonstrated in [21], a technical system requires different virtual tests. Therefore, multiple virtual testing workflows need to be created for a complex technical system with different test purposes. Executing these virtual testing workflows comprehensively can detect failure risks (such as bearing fatigue analysis) and analyze the performance (such as WT economic analysis) of system solutions. The testing results provide evidence that the system solution fulfills the corresponding function.

“Action 1” invokes the standardized activity that contains the actions which are divided into the following two parts in Figure 4:



**Figure 4.** The workflow to check if the solution fulfills the function based on virtual tests.

Part 1 consists of multiple actions that invoke the virtual testing workflows with different test purposes to virtually test the solution. These actions are designed independently of each other and invoke the corresponding virtual test workflows in parallel. As an example, to check whether the solution “Bearing” fulfills the corresponding function “Support rotation with low friction”, fatigue testing and smearing testing on the solution are required. The testing results of each virtual test should be verified against design requirements and the verification values can be exported through the output pin around the action. The specific method for developing a virtual testing workflow will be introduced in detail in Section 4.2.

Part 2 includes the actions called “Check all verification results” and “Save requirement satisfaction results”. The action “Check all verification results” checks the results of each virtual test. The inputs of this action connect the outputs of the “Virtual testing workflow”



actions through object flows. Only when each virtual testing result is “True”, the solution can be considered feasible, that is, it fulfills the corresponding function. Otherwise, the solution under the current parameter setting is considered not to fulfill the function. The action “Save satisfaction results” saves the result of whether the solution fulfills the function of the system model.

#### 4.1.3. Implementation of the Proposed Virtual Validation Workflows in SysML

The virtual validation process consists of the continuous activities of engineers. The activity diagram provides the ability to model a process in SysML and represent it as a continuous flow of activities. Therefore, a virtual validation workflow can be modeled in the activity diagram (i.e., virtual validation workflow) in this work. Actions serve as the foundation for activities, outlining their execution process. In this research, a notable type of action is the “Call Behavior Action”, which invokes a behavior (assumed to be an activity or state machine) when executed. The symbol for “Call Behavior Action” (see Figure 5) is a round-cornered box containing the action and invoked behavior name separated by a colon: “Action name: Activity name” [22].

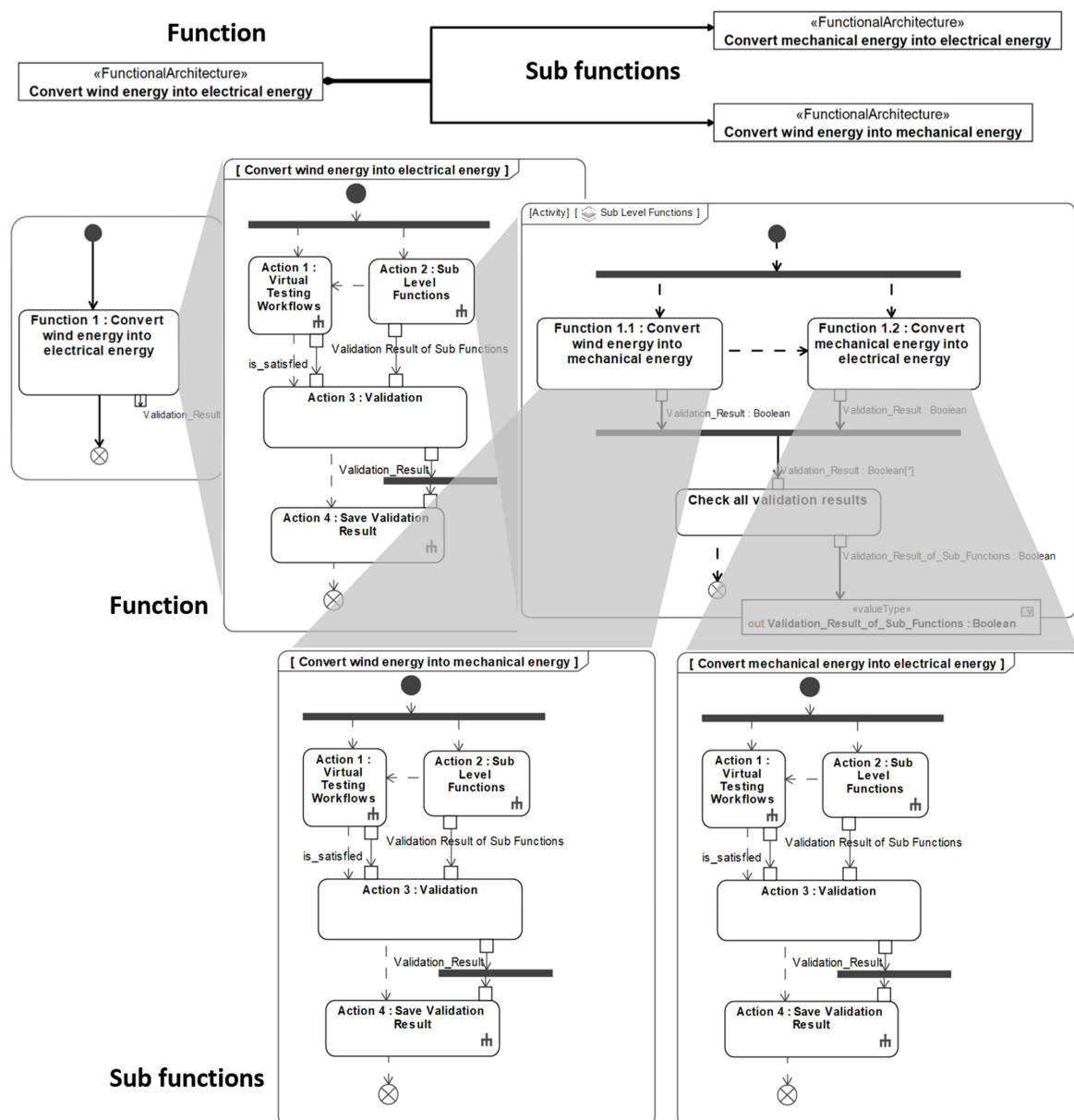


Figure 5. The framework of the hierarchical functional validation workflows in SysML.

Input and output pins, represented by small rectangles surrounding the action, receive items containing relevant data information (e.g., validation results). Data consists of data name and data type, separated by colons. The square brackets after the data type contain the multiplicity of data values, and the asterisk indicates that the number of data values can be arbitrary. Object flows transfer items between pins and can be used to route items from the pin of an activity to the pins of its sub-actions or to connect pins of other actions directly. An object flow is shown as a solid line connecting the flow's source to its destination, with an arrow pointing toward the destination.

Control flows, represented by dashed lines with arrows, convey the sequence and timing of actions executed within an activity. When a control flow connects two actions, the destination action cannot commence until the source action is complete. Control nodes, including join, fork, decision, merge, initial, and final nodes, can synchronize incoming concurrent flows and decide incoming alternative flows, thereby further specifying the order of actions.

In Figure 5, the nested virtual validation workflows of the WT system are shown in the activity diagram. Each function in the system's functional architecture corresponds to a virtual validation workflow. As an example, the "Virtual Validation Workflow" activity of the function "Convert wind energy to electrical energy" contains two sub-virtual validation workflows for validation of the functions "Convert wind energy to mechanical energy" and "Convert mechanical energy to electrical energy", respectively. Each function corresponds to a standardized design virtual validation workflow. The standardized design ensures that the virtual validation workflow can be easily redesigned, with related functions being reused in different systems.

Figure 6 shows that the virtual validation workflow framework can be semi-automatically extended and redesigned in SysML. Engineers can easily drag and drop existing "Virtual testing workflow" activity from the corresponding function SysML package into "Action 1: Check if the function is fulfilled by corresponding solution" to agilely add or replace any virtual testing process with a specific test need. In addition, engineers can also easily drag and drop the existing "Virtual validation workflow" activity from the corresponding virtual testing SysML package into "Action 2: Sub Level Functions" to agilely add or replace any virtual validation process for a specific function. It is worth noting that when a new workflow is added (either a virtual validation workflow or a virtual testing workflow), a new workflow result is generated correspondingly. These results should also participate in logic gate calculations and be checked.

#### *4.2. The Framework of Virtual Testing Workflows Based on the Functional Architecture*

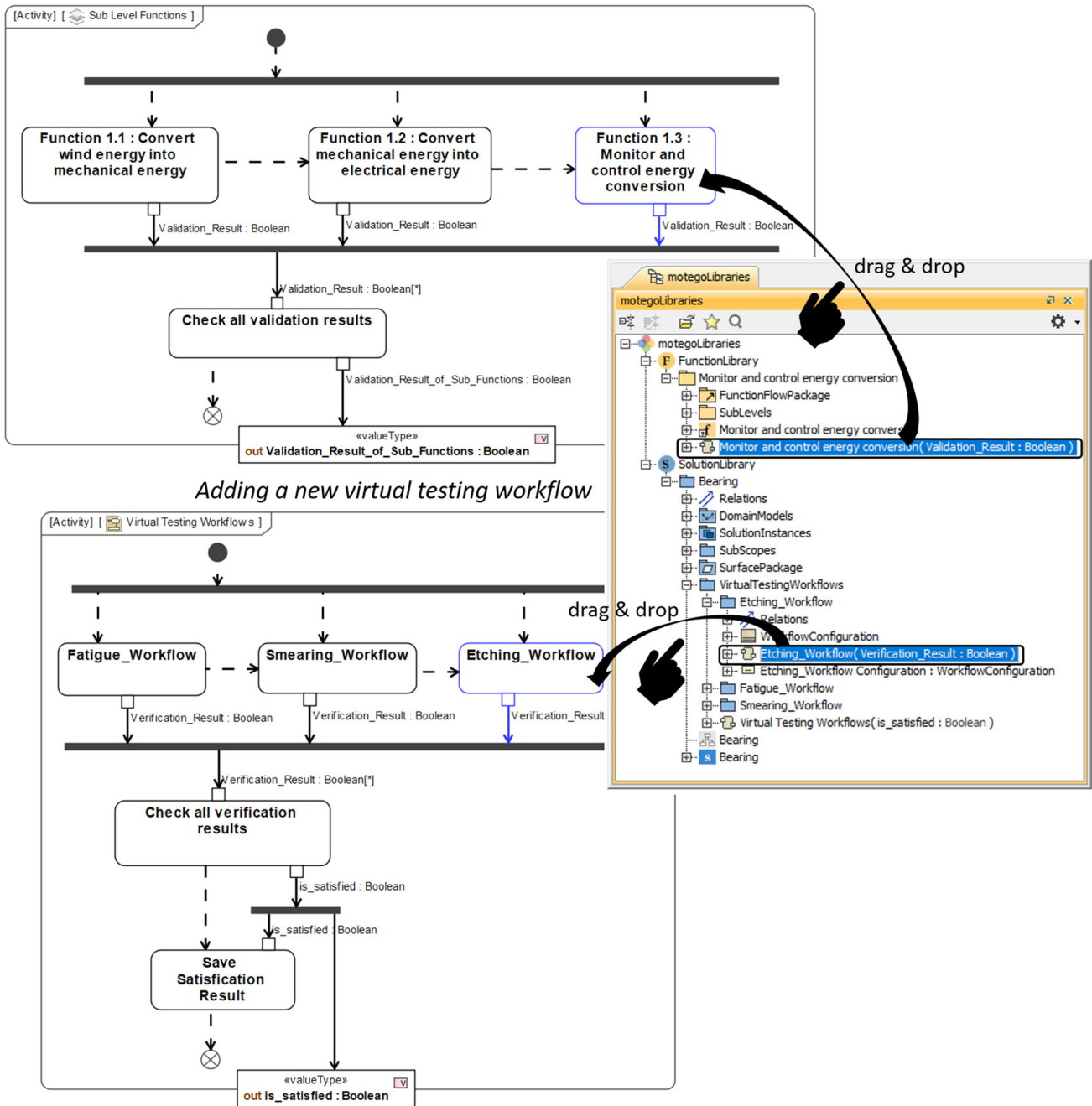
As described in Section 4.1, for the virtual validation of a function's fulfillment by its corresponding solution, various virtual testing workflows need to be designed for distinct test purposes. These workflows model and perform a series of simulation activities. To perform appropriate simulation activities, engineers typically need to choose the right domain models and manually sequence their simulations in virtual testing workflows. To minimize manual effort in sequencing workflows, this section introduces a framework for building multiple virtual testing workflows, enabling engineers to easily design and reuse them for different testing purposes.

##### *4.2.1. Modular Design of Virtual Testing Workflows*

Virtual testing workflows employ a modular design, breaking down the workflows into smaller components called modules. Each simulation activity serves as a module for performing the related domain model simulations. Simulation activities are nested according to the domain model's classification. In this work, domain models are categorized based on system scope, model purpose, and model fidelity [6]. The system scope describes the extent of the physical system that the domain model deals with. The model purpose describes the objective of the simulation for the domain model. The model fidelity describes the degree of exactness of the domain model simulation results compared to

reality. These classifications are interconnected; for instance, a simulation activity associated with a particular system scope includes simulation activities that execute models with varying purposes.

*Adding a new virtual validation workflow*



**Figure 6.** Adding virtual validation workflows & virtual testing workflows.

Modular simulation activities, organized according to domain model classification, offer significant flexibility in designing virtual testing workflows, which ensures that domain model simulation activities can be easily reused in different virtual testing workflows. Therefore, workflows can be easily adapted when the solution is used in different system contexts.

As shown in Figure 7, corresponding to the system architecture [23], the system scopes should be regarded at the highest level within the nested structure. In the framework,

each system scope contains domain models with different model purposes. In addition, analogous to the hierarchical design of the functional virtual validation workflow, a large system scope contains small system scopes. As an example, the activity in the scope “Wind Turbine” is divided into two parts:

- The first part is a parallel arrangement of actions that, respectively, invoke the simulation activities regarding the domain models of the solution “Wind Turbine”. These domain models are classified according to the purpose of the model, such as Load analysis models and AEP models of the wind turbine, etc.
- The second part is the action, which contains sub-actions arranged in parallel. These sub-actions are used to invoke the activities regarding its sub-scopes (e.g., Nacelle) of the “Wind Turbine”. These sub-scope activities follow the same design as above, and also contain their own sub-scopes (e.g., Gearbox, Bearing), and so on to form a hierarchical structure.

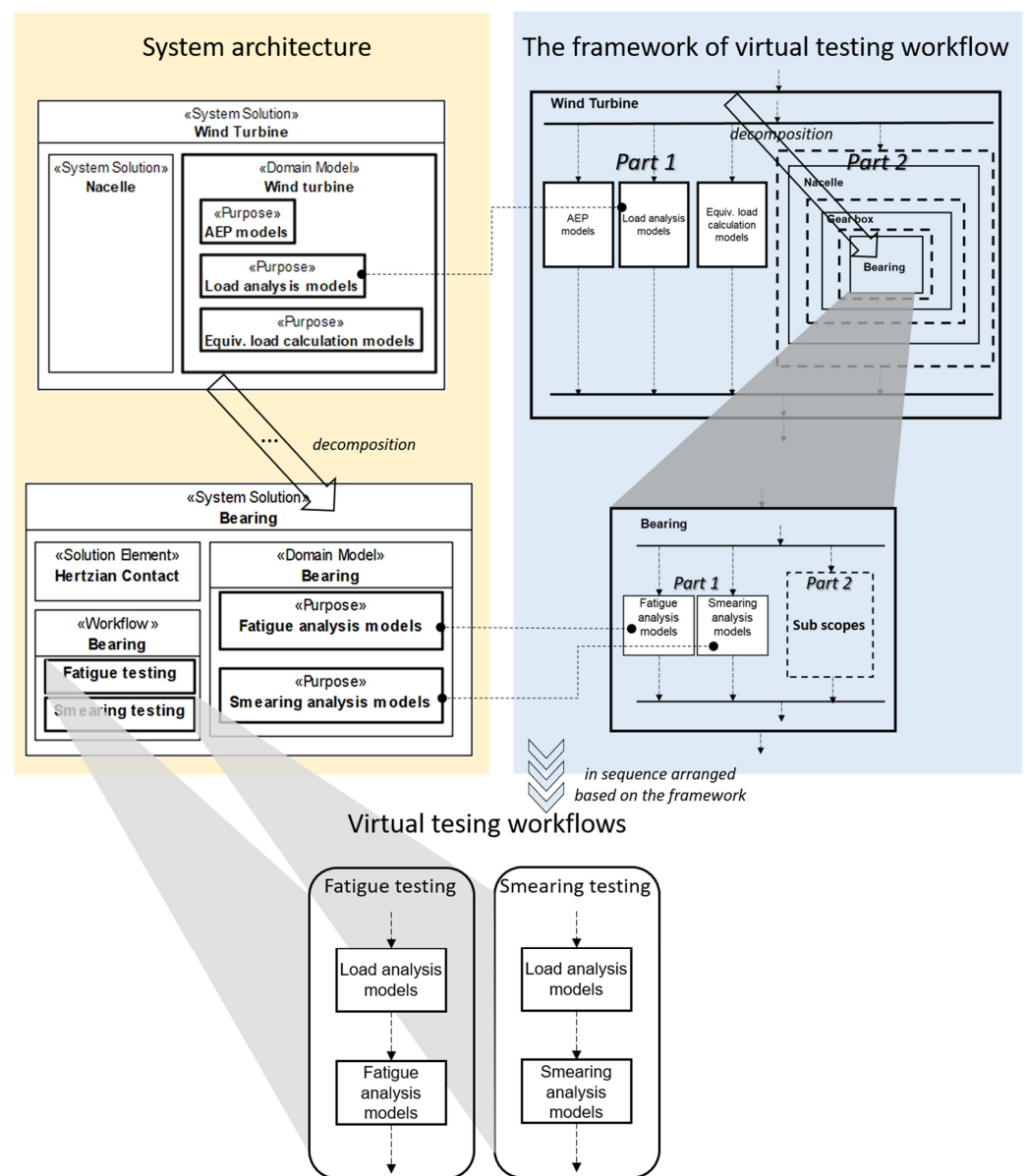
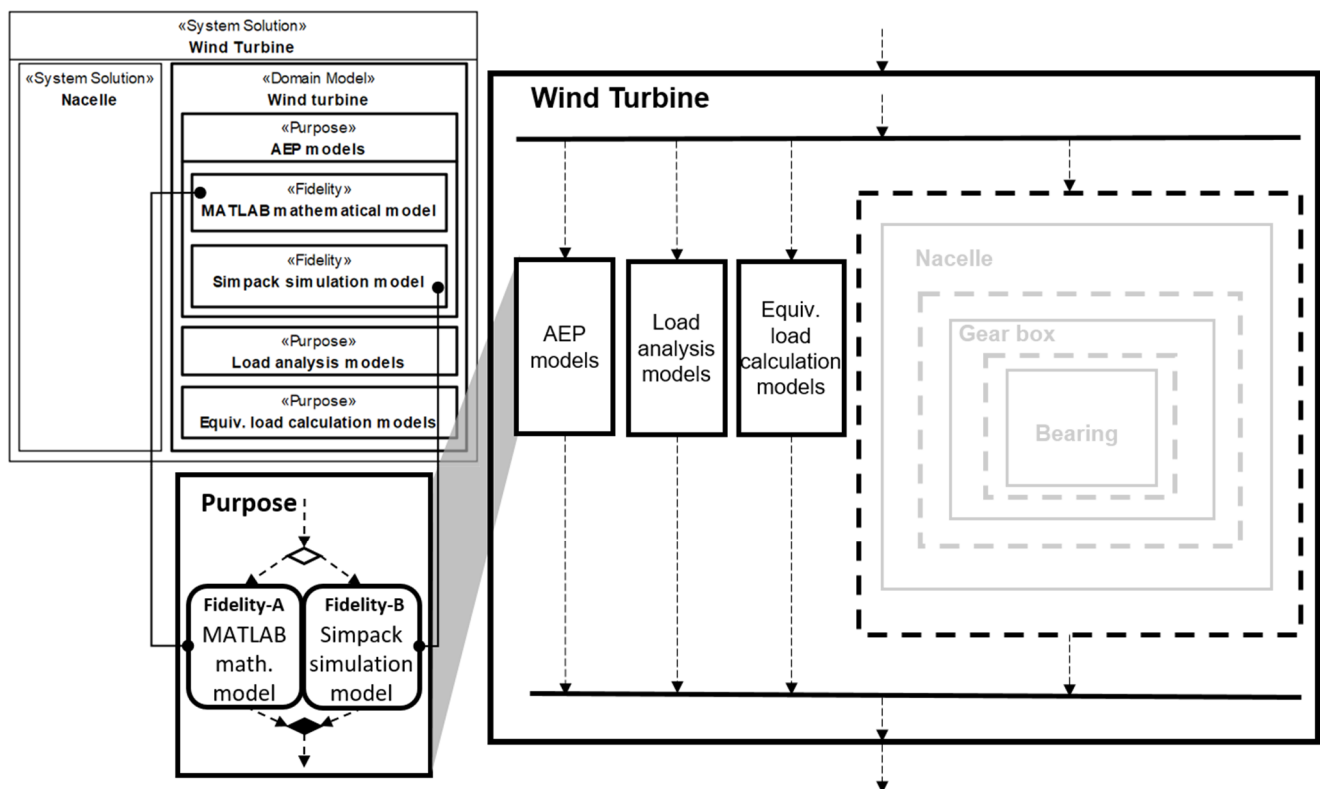


Figure 7. The nested framework of virtual testing workflows corresponding to the hierarchical system scopes.

Ref. [19] mentioned that model fidelity also needs to be considered in the virtual testing workflow. This work further develops simulation activities with specific model purposes. As shown in Figure 8, similar to the nested structure design of the activities with a specific system scope, an activity for the same purpose includes sub-simulation activities with different model fidelities. For example, the “AEP Models” activity is used to calculate the annual energy production of a WT and this activity contains two actions arranged in parallel. These two actions, respectively, invoke two simulation activities with different model fidelity, which are named the “MATLAB mathematical model (Fidelity-A)” and the “Simpack simulation model (Fidelity-B)” to calculate the values of the annual energy production.



**Figure 8.** The nested framework of virtual testing workflows based on the model classifications.

In the virtual testing workflow, the activity with specific model fidelity can perform external simulations by triggering constraints in ibds; details are already published in the related paper [19].

#### 4.2.2. Implementation of the Proposed Virtual Testing Workflows in SysML

The previous chapter created a nested framework for virtual testing workflows based on modular simulation activities. This chapter describes how to use this framework to easily select different simulation activities and execute these simulation activities step by step in an iterative manner based on SysML. This enables engineers to automatically design and easily execute virtual test workflows that adapt to different test scenarios.

As mentioned earlier, virtual testing workflows typically perform a series of simulation activities in a specific order. In order to automate the entire virtual testing process step by step, workflow designers usually need to provide the necessary model information for each step of the virtual testing process. Designers can pre-give the information of these domain models according to the classification of the domain models, such as the scope of the model, the purpose of the model, and the fidelity of the model, so that the workflow can automatically iteratively run according to the customized sequences. When the scope,



purpose, and fidelity of the model are determined, the required simulation activity of the domain model is also determined. After the sequence has been set up, the virtual testing workflow becomes reusable.

The model information can be pre-set in the instances of the system model so that the virtual testing workflows can be easily re-executed by engineers. Figure 9 shows how different virtual tests can be executed in SysML through an iterative process based on the architecture. An action named “Workflow Steps Information” is designed to read required domain model information and pass them through the pin to the framework that contains all the simulation activities. To avoid the repetitive creation of simulation activities, this study designs a unique corresponding simulation activity for each domain model in the framework. The workflow automatically finds and executes the corresponding simulation activity based on the model information, which is regarded as the completion of a simulation process. The above process is repeated by executing an iterative loop and the model information of each step is obtained in turn to sequentially execute the relevant simulation activities in the workflow.

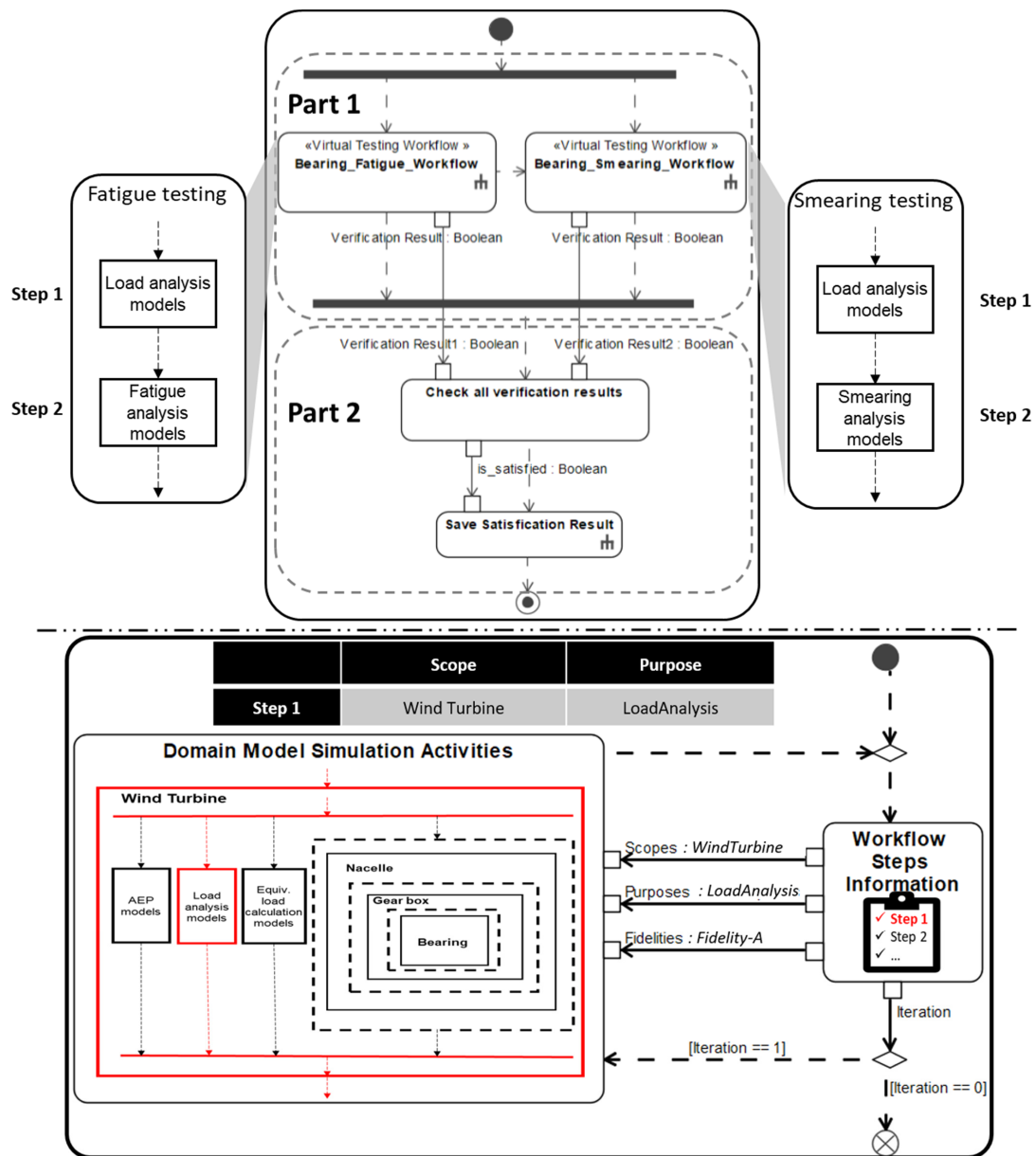
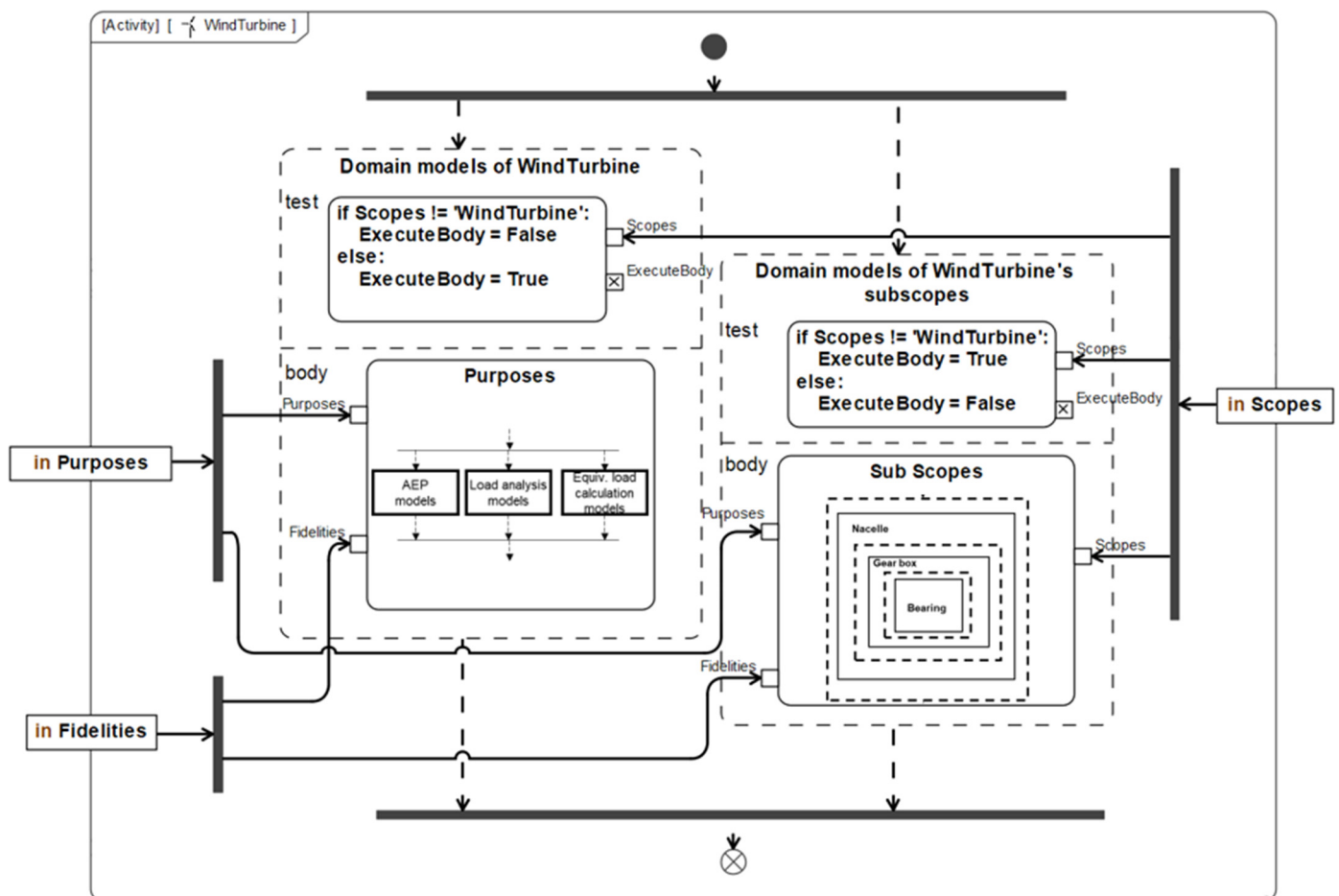


Figure 9. Execution of the simulation activities in the workflow through an iterative loop.

Figure 10 shows how to implement the architecture in the SysML based on the model information to achieve automatic decision-making. Taking the bearing fatigue test as an example, when the virtual test starts, the model information to be executed in the first step (i.e., “Wind Turbine”; “LoadAnalysis”) is passed to the framework. A conditional node action called “Domain models of WT” is created at the scope “Wind Turbine” of the nested framework to determine whether to execute the domain model in the Wind Turbine scope. A conditional node comprises a collection of clauses, each with a test and a body, functioning similarly to an “if” statement in programming languages, where the action is performed only under specific conditions. When the conditional node begins execution, the test portion of the clause is carried out; if the test conditions evaluate to “True”, the clause’s body segment is executed. Specifically, when the scope information is “Wind Turbine”, the action “Purposes” will be executed, which encapsulates all domain model simulation activities within the “Wind Turbine” scope. Another conditional node action called “Domain models of WT’s sub scopes” is also created in parallel in this scope to determine if the domain model that needs to be executed is in the sub-scope of the “Wind Turbine” scope (e.g., Nacelle). This goes on until the required model scope is found.

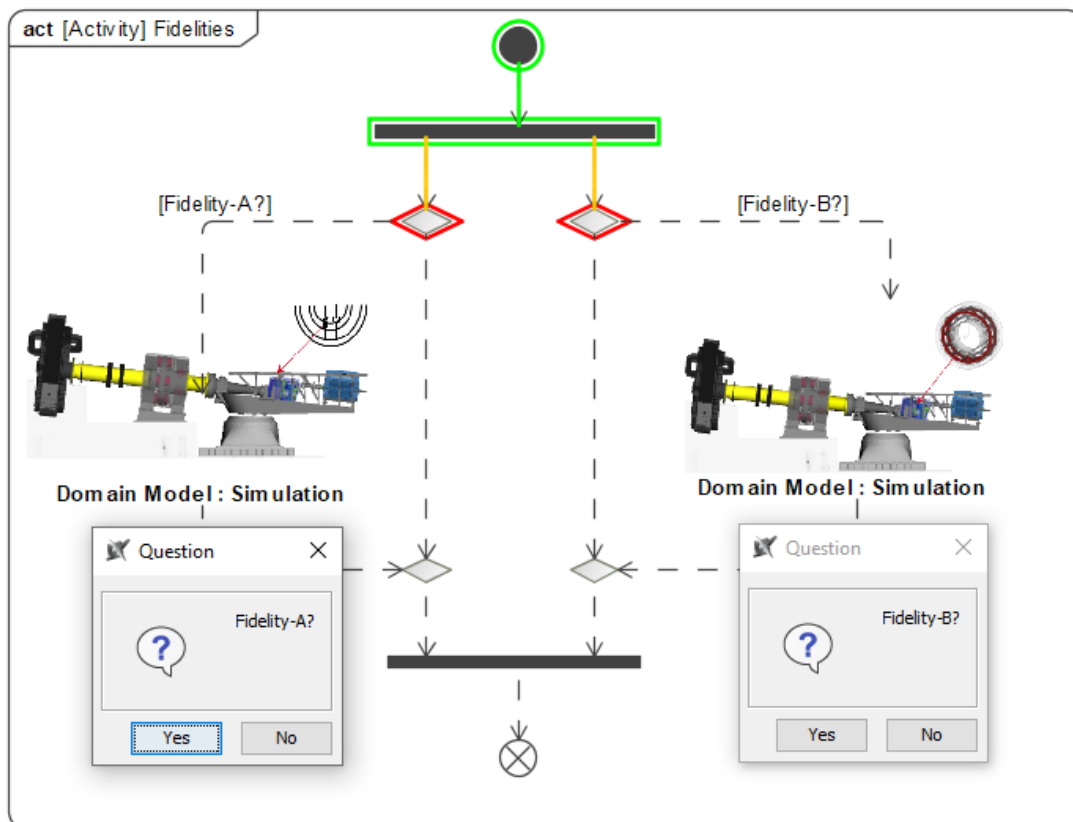


**Figure 10.** Determination of the system scope of the domain model automatically based on conditional node activities.

After determining the scope of the model to be executed, the purpose of the model also needs to be determined based on the information of the model. Similar to the above judgment process, conditional node actions are created for different model purposes separately and these actions are arranged in parallel. The purpose information of the model will be delivered to all of these actions to determine whether the model that needs to be

executed belongs to any of them. In the same way, engineers can set the appropriate model fidelity before the workflow starts and the workflow can automatically find the models that need to be executed.

It is worth noting that for information that is difficult to be given in advance, such as model fidelity, engineers can also manually select the existing alternative models during workflow execution. As shown in Figure 11, while the workflow is in progress, engineers can manually choose the suitable models at the decision node. The symbol of the decision node is a diamond that has one or more output flow that is offered to the engineer for selection using guard conditions. For example, for bearing fatigue testing, engineers can select “Fidelity-A” (e.g., Spring damper MBS model) or “Fidelity-B” (e.g., Lambda MBS model) to calculate the loads on the bearing.



**Figure 11.** Determination of the fidelity of the domain model manually based on decision nodes.

Similarly, engineers can define the desired model fidelity before initiating the workflow, enabling the workflow to automatically identify the models that require execution.

## 5. Case Study

Taking the virtual validation scenario described in Section 3 as an example, the proposed workflows can be used to validate parts of the system functions of the WT system at different hierarchy levels by performing specific virtual tests. When the virtual validation begins, the workflows validate each function in turn from the top functional level to the detailed functional level, until the end of the functional hierarchy. The top-level virtual validation workflow of the WT is created in the SysML package “Convert Wind Energy to Electricity” at the function layer. The virtual validation workflow performs an activity called “Check if the corresponding solution fulfills the function” to check that the function can be fulfilled by the selected solution “Wind Turbine A”. The designer parameterizes the solution “Wind Turbine A” in the SysML instances. This function can be decomposed into two sub-functions in this case study, which are “Convert wind energy into mechanical

energy” and “Convert mechanical energy into wind energy”. These two functions also have their own virtual validation workflows. The two workflows can be added to the top-level virtual validation workflow by dragging and dropping from the model tree. By analogy, for each new function created by the engineers, the corresponding virtual validation workflows in different functional hierarchy levels need to be created. Among them, “Support rotation with low friction” is a sub-function of the WT system that this case study focuses on. “Bearing A” is a sub-solution in “Wind Turbine A” designed to satisfy this sub-function.

Designers design multiple virtual testing workflows for each solution to provide evidence that the solution can fulfill the corresponding function. For example, as mentioned in Section 3, the AEP calculation is an important virtual test that is used to calculate the performance of the entire system. Therefore, in this case, the AEP calculation is used as a representative criterion to verify that “Wind Turbine A” satisfies the function “Convert wind energy to electricity”. AEP calculation is the predicted annual energy production of a WT calculated based on a given rated average wind speed. In this work, a co-simulation model is used to calculate the AEP. The co-simulation model combines the mechanical and control model, as well as directly providing the engineer with the simulation results of the transient power output of the generator. These simulation results are post-processed to obtain the AEP results. The co-simulation model involving a mechanical model and a control model for wind turbine systems is conducted to optimize speed control using pitch angles as the control output. The mechanical model considers inputs such as wind loads and outputs, including drivetrain torque, rotor speed, structural loads, and blade pitch angles. On the other hand, the control model processes inputs such as current generator speed and current pitch angle as references to determine the appropriate pitch angle settings.

For the components in a system, failure analysis can be an important virtual test to test the reliability of the component. The result of the failure analysis (i.e., the Lifetime) can therefore be used as a criterion to verify that “Bearing A” satisfies the function “Support rotation with low friction”. In order to avoid fatigue failure of bearings, engineers often use statistical methods to analyze and calculate bearing lifetime. The basic fatigue rating life is calculated using the number of rotations in which 90% of all bearings in a specific group achieve or exceed without failure (probability of failure: 10%). A standardized formula, also known as the catalog method (ISO 281), is the common means of calculating a bearing’s basic rating life. ISO 281 is applicable to the bearing under continuous rotation subjected to axial and radial loads [24]. The activity “Check if the corresponding solution fulfills the function” contains and executes the virtual testing workflows. The testing results are compared with the design requirements for verification. Taking the function “Support rotation with low friction” as an example, the virtual testing workflow “Failure Testing” is created in the SysML package “Bearing” at the solution layer. The workflows can be added to the activity “Check if the corresponding solution fulfills the function” by dragging and dropping. That is, by performing the virtual testing workflow “Failure Testing”, the simulation result “Lifetime” will be used as a criterion to check whether the solution “Bearing A” fulfills the corresponding function. Simulation testing results must be compared to the relevant design requirements to assess whether the function is fulfilled based on the verification. For example, bearing lifetime serves as one criterion, assisting in determining if the chosen solution satisfies the function according to the lifetime calculation. If the simulated lifetime meets or exceeds the required lifetime, the verification results are considered true, indicating a passed verification. Otherwise, the verification fails.

Figure 12 shows the validation results of the WT system under different design parameters. The same as the scenario in Section 3, customers expect a higher energy production from the system. Therefore, the calculated AEP value of the original solution “Wind Turbine A” is lower than the changed AEP requirement. Therefore, the requirement verification fails. The failed verification result states that the solution “Wind Turbine A” does not fulfill the function “Convert wind energy into electricity”. The “Wind Turbine

B” is designed to meet AEP requirements, while the larger blades and generator result in higher loads on the redesigned rotor and a corresponding increase in loads on the bearings. Therefore, the lifetime of the bearing is lower than the design requirement, and it is also regarded as a verification failure. Bearings have been redesigned in “Wind Turbine C”. It has been verified to meet not only AEP requirements in the scope of WT, but also lifetime requirements in the scope of bearing.

| #  | Name                      | △ Lifetime : Time | VerificationResult : Boolean             | ValidationResult : Boolean               | ▽ AEP : Energy | VerificationResult : Boolean             | ValidationResult : Boolean               |
|----|---------------------------|-------------------|--|--|----------------|--|--|
| 1  | WindTurbine A             |                   |  |  | 8.25 GWh/year  | <input type="checkbox"/> false           | <input type="checkbox"/> false           |
| 2  | Drivetrain A              |                   |  |  |                |  |  |
| 3  | RotorBearingArrangement A |                   |  |  |                |  |  |
| 4  | Bearing A                 | 452,120 hours     | <input checked="" type="checkbox"/> true | <input checked="" type="checkbox"/> true |                |  |  |
| 5  | WindTurbine B             |                   |  |  | 13.37 GWh/year | <input checked="" type="checkbox"/> true | <input type="checkbox"/> false           |
| 6  | Drivetrain B              |                   |  |  |                |  |  |
| 7  | RotorBearingArrangement B |                   |  |  |                |  |  |
| 8  | Bearing B                 | 82,014 hours      | <input type="checkbox"/> false           | <input type="checkbox"/> false           |                |  |  |
| 9  | WindTurbine C             |                   |  |  | 13.32 GWh/year | <input checked="" type="checkbox"/> true | <input checked="" type="checkbox"/> true |
| 10 | Drivetrain C              |                   |  |  |                |  |  |
| 11 | RotorBearingArrangement C |                   |  |  |                |  |  |
| 12 | Bearing C                 | 430,110 hours     | <input checked="" type="checkbox"/> true | <input checked="" type="checkbox"/> true |                |  |  |

Figure 12. The verification & validation results of the WT system under different design alternatives.

It is worth noting that the virtual testing workflows can be easily adapted when the system context changes. “Bearing” is a common solution in WT systems. The solution “Bearing” can be used not only in rotor-bearing arrangements, but also in high-speed bearing arrangements. In the reuse process, designers can easily add the corresponding validation workflow, virtual testing workflow, and modular domain model simulation activities into the framework. Due to changes in the system context, the designer might need to change the step information of virtual testing workflows to execute the appropriate domain model. For example, to calculate the lifetime of the bearing of the high-speed bearing arrangement, loads on the bearing can be calculated from a “Nacelle MBS” domain model instead of the “Wind Turbine MBS” domain model.

### 6. Discussion

The case study shows that the proposed workflow design framework can be applied to support the virtual validation process of complex technical systems. The virtual validation process based on the function-oriented system architecture by using a standardized designed virtual validation workflow has advantages over the classical validation process mentioned in the previous literature.

First of all, a nested virtual validation workflow is designed and implemented in an activity diagram, which contains sub-virtual validation workflows based on a hierarchical functional architecture. Therefore, designers can easily use automated workflows to fully validate the functions at all hierarchy levels. Compared to classical methods that rely on the manual work and experience of engineers, using the proposed virtual workflow allows early and reliable detection of the risk of technical system failure due to redesign. This work shows that since the nested framework of the virtual validation workflows corresponds to the system’s functional architecture, virtual validation workflows can be created synchronously as new functions are added and related sub-virtual validation workflows can also be created as functions are decomposed.

Secondly, each function includes a corresponding standardized designed virtual validation workflow, such as fatigue analysis and smearing analysis. The results of these virtual tests can be used as validation criteria. This work finds that the functions can be validated based on these virtual tests, which contain the related simulation activities of the domain models at different hierarchy solution levels. Compared to classical methods, the



standardized workflow design in this work ensures that the virtual validation workflow can be easily redesigned, with related functions being reused in different systems.

At last, the modular design based on model classification greatly improves the design agility of the virtual testing workflows. A virtual testing workflow consisting of modular simulation activities allows testers to select appropriate models. Additionally, the workflow with modular design can be easily adapted in different system contexts. Compared to classical methods, the virtual testing workflows proposed in this work can automatically and sequentially execute the corresponding simulation activities through iterations according to the step information provided by the engineer.

This work uses a WT system as an example to demonstrate the application of workflow in a technical system. Although the scope of the system displayed by the WT system is very specific, as the modeling method of the system architecture is function-oriented, and the workflow design is also modularized based on the system architecture, the conceptual design of the workflow is considered to be scalable in theory. However, the current workflow still has limitations. While engineers can use pre-designed workflows to automatically perform the virtual validations of system solutions, the workflow still relies on experienced engineers to manually determine the order of a series of simulation activities in a workflow. Therefore, to further automate the virtual validation process while reducing potential errors from manual steps, a mechanism should be developed to automatically arrange simulation sequences based on dependencies between parameters in the proposed workflow and ensure data matching between models. After the virtual validation process, engineers need to optimize and redesign the system based on the validation results and perform the re-validations. However, this method does not give a solution that would provide an adaptive system design after virtual validation is failed. Therefore, it is necessary to extend the workflow to support optimization mechanisms. The designer should be assisted in redesigning the system after virtual validation and decide whether the next iteration can start or not until the design converges toward a solution that fulfills the requirement. As the design loops can be implemented by optimization algorithms (e.g., Multidisciplinary Analysis and Optimization (MDAO)), the automation of the proposed workflow can be further improved. Finally, the proposed structured virtual validation workflows still need to be created manually by engineers. The future perspective is that workflow structures can be created automatically, corresponding to the functional architecture structure.

## 7. Conclusions

In order to efficiently validate whether a system's solutions fulfill their intended functionalities at the early development stage, this work provides a standardized method in the function-oriented system architecture to build well-organized virtual validation workflows. The workflows can support the system's virtual validation across all functional hierarchy levels. System engineers can carry out virtual validation of the necessary system by utilizing the workflow. This work covers the following core content: a nested structure for the virtual validation workflow is developed that corresponds to the functional hierarchy of the system. The decomposed solutions can be virtually validated at all hierarchy levels to satisfy the corresponding functions. The virtual testing workflows are arranged in parallel to each other as the criteria to provide evidence that the solution fulfills the corresponding function. Furthermore, this work presents a design approach for a standardized virtual testing workflow framework, which is grounded in domain model classification. The following key statements can be made: each domain model corresponds to a simulation activity. The simulation activity is designed modularly, which increases its reusability. Simulation activities are nested based on the system scope, model purpose, and model fidelity within the framework. In a virtual testing workflow, engineers execute the corresponding simulation activities in the framework step by step through an iterative cycle. One of the case studies of this work validates the WT system, focusing on the scope of "Wind Turbine" and its sub-systems "Bearing". The impact on the system caused by

design changes is observed by performing the automated virtual validation workflows, and “Bearing” solutions can be easily reused in different system contexts.

The contribution of this work is to develop automated virtual validation processes for complex technical systems, providing automated validation, early failure risk detection, and ease of redesign. The function-oriented hierarchical architecture allows for synchronous workflow creation, while the standardized design ensures function reusability. Modular design, based on model classification, improves design agility and adaptability in different system contexts. Future improvements involve the introduction of dependency detection mechanisms for arranging simulation sequences, extending workflows to support optimization algorithms, and automating the creation of workflow structures corresponding to functional architecture.

**Author Contributions:** Conceptualization, Y.Z., G.J., J.B. and G.H.; methodology, Y.Z., G.J., J.B. and G.H.; software, Y.Z.; validation, Y.Z.; formal analysis, Y.Z., G.H. and J.B.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z., J.B. and G.H.; supervision, G.J. and J.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Walden, D.D.; Roedler, G.J.; Forsberg, K.; Hamelin, R.D.; Shortell, T.M. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 4th ed.; Prepared by International Council on Systems Engineering (INCOSE); David, D., Walden, E.S.E.P., Garry, J., Roedler, E.S.E.P., Kevin, J., Forsberg, E.S.E.P., Douglas Hamelin, R., Thomas, M., Shortell, C.S.E.P., Eds.; Wiley: Hoboken, NJ, USA, 2015; ISBN 978-1-118-99940-0.
2. IEA. Wind Electricity—Analysis—IEA. Available online: <https://www.iea.org/reports/wind-electricity> (accessed on 15 January 2023).
3. Dykes, K.; Meadows, R. *Applications of Systems Engineering to the Research, Design, and Development of Wind Energy Systems*; National Renewable Energy Lab. (NREL): Golden, CO, USA, 2011.
4. *ISO/IEC/IEEE 15288; Systems and Software Engineering—System Life Cycle Processes*. IEEE: Piscataway, NJ, USA, 2015.
5. INCOSE. *Systems Engineering Vision 2020*; INCOSE: San Diego, CA, USA, 2007.
6. Jacobs, G.; Konrad, C.; Berroth, J.; Zerwas, T.; Höpfner, G.; Spütz, K. Function-Oriented Model-Based Product Development. In *Design Methodology for Future Products*; Krause, D., Heyden, E., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 243–263, ISBN 978-3-030-78367-9.
7. Drave, I.; Rumpe, B.; Wortmann, A.; Berroth, J.; Hoepfner, G.; Jacobs, G.; Spuetz, K.; Zerwas, T.; Guist, C.; Kohl, J. Modeling mechanical functional architectures in SysML. In *Proceedings of the MODELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 16–23 October 2020*; Syriani, E., Sahraoui, H., Eds.; ACM: New York, NY, USA, 2020; pp. 79–89, ISBN 9781450370196.
8. Estefan, J.A. *Survey of Model-Based Systems Engineering (MBSE) Methodologies*; IncoSE MBSE Focus Group: Pasadena, CA, USA, 2008.
9. Karban, R. *MBSE Practices in Telescope Modeling*; INCOSE MBSE Challenge Team: Colorado Springs, CO, USA, 2010.
10. Gaignic, P.; Vosgien, T.; Jankovic, M.; Tuloup, V.; Berquet, J.; Troussier, N. Complex System Simulation: Proposition of a MBSE Framework for Design-Analysis Integration. *Procedia Comput. Sci.* **2013**, *16*, 59–68. [[CrossRef](#)]
11. Lange, C.; Grundmann, J.T.; Kretzenbacher, M.; Fischer, P.M. Systematic reuse and platforming: Application examples for enhancing reuse with model-based systems engineering methods in space systems development. *Concurr. Eng.* **2018**, *26*, 77–92. [[CrossRef](#)]
12. Zhang, Y.; Hoepfner, G.; Berroth, J.; Pasch, G.; Jacobs, G. Towards Holistic System Models Including Domain-Specific Simulation Models Based on SysML. *Systems* **2021**, *9*, 76. [[CrossRef](#)]
13. Zerwas, T.; Jacobs, G.; Spütz, K.; Höpfner, G.; Drave, I.; Berroth, J.; Guist, C.; Konrad, C.; Rumpe, B.; Kohl, J. Mechanical concept development using principle solution models. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1097*, 12001. [[CrossRef](#)]
14. Ralph, P.; Wand, Y. A Proposal for a Formal Definition of the Design Concept. In *Design Requirements Engineering: A Ten-Year Perspective*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 103–136.

15. Bretz, L.; Tschirner, C.; Dumitrescu, R. A concept for managing information in early stages of product engineering by integrating MBSE and workflow management systems. In Proceedings of the 2016 IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 3–5 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–8, ISBN 978-1-5090-0793-6.
16. Zou, M.; Vogel-Heuser, B.; Sollfrank, M.; Fischer, J. A Cross-disciplinary Model-Based Systems Engineering Workflow of Automated Production Systems Leveraging Socio-technical Aspects. In Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 14–17 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 133–140, ISBN 978-1-5386-7220-4.
17. Spangelo, S.C.; Cutler, J.; Anderson, L.; Fosse, E.; Cheng, L. Model based systems engineering (MBSE) applied to Radio Aurora Explorer (RAX) CubeSat mission operational scenarios. In Proceedings of the 2013 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2013.
18. Höpfner, G.; Jacobs, G.; Zerwas, T.; Drave, I.; Berroth, J.; Guist, C.; Rumpe, B.; Kohl, J. Model-Based Design Workflows for Cyber-Physical Systems Applied to an Electric-Mechanical Coolant Pump. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1097*, 12004. [[CrossRef](#)]
19. Zhang, Y.; Roeder, J.; Jacobs, G.; Berroth, J.; Hoepfner, G. Virtual Testing Workflows Based on the Function-Oriented System Architecture in SysML: A Case Study in Wind Turbine Systems. *Wind* **2022**, *2*, 599–616. [[CrossRef](#)]
20. Rohrig, K.; Berkhout, V.; Callies, D.; Durstewitz, M.; Faulstich, S.; Hahn, B.; Jung, M.; Pauscher, L.; Seibel, A.; Shan, M.; et al. Powering the 21st century by wind energy—Options, facts, figures. *Appl. Phys. Rev.* **2019**, *6*, 31303. [[CrossRef](#)]
21. Zorriassatine, F.; Wykes, C.; Parkin, R.; Gindy, N. A survey of virtual prototyping techniques for mechanical product development. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2003**, *217*, 513–530. [[CrossRef](#)]
22. Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML: The Systems Modeling Language*, 3rd ed.; Elsevier MK Morgan Kaufmann is an imprint of Elsevier; Morgan Kaufmann: Amsterdam, The Netherlands; Boston, MA, USA, 2015; ISBN 9780128002025.
23. Spütz, K.; Berges, J.; Jacobs, G.; Berroth, J.; Konrad, C. Classification of Simulation Models for the Model-based Design of Plastic-Metal Hybrid Joints. *Procedia CIRP* **2022**, *109*, 37–42. [[CrossRef](#)]
24. *ISO 281; Wälzlager-Dynamische Tragzahlen und Nominelle Lebensdauer*. Beuth Verlag GmbH: Berlin, Germany, 2007.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.