31st Annual INCOSE international symposium
Honolulu, HI, USA
July 17 - 22, 2021

Ontology-Based Search Engine For Simulation Models From Their Related System Function

Author's Name (only one author)
Organization
Mailing Address
Telephone
author.one@gmail.com

Author's Name (first of many)
Organization
Mailing Address
Telephone
author.one@gmail.com

Author's Name (second of many)
Organization
Mailing Address
Telephone
author.one@gmail.com

Author's Name (last of many)
Organization
Mailing Address
Telephone
author.one@gmail.com

Abstract. Today's highly competitive industrial environment is leading companies to drastically reduce their development time for an increasingly early time-to-market, while maintaining a low level of risk. In this context, simulation-based design has become widespread among design teams. Yet, the complexity of today's systems related to the integration of more new information and communication technologies requires multidisciplinary simulations. In charge of designing a simulation architecture that meets a request from system architects, simulation architects specify the simulation models to be provided by disciplinary experts. In this context, to accelerate the design process, capitalization and reuse processes can support the search for similar models in an existing database. Therefore, we propose to develop a search engine for simulation models. This paper focuses on the presentation of our initial works around an ontology-based search engine of simulation models representing the behavior of a given system function. The developed algorithms and similarity metrics are based on both reference functions and ports. The corresponding inference engine has been applied on an autonomous vehicle scenario.

Introduction

Today, simulation has become an indispensable tool for companies in product design, as it brings major gains in technical design: virtual prototyping, early risk reduction, etc. In an increasingly complex design context integrating several disciplines and actors, simulation processes have to face the issues of multidisciplinary simulations. As a result, companies must also ensure proper coordination of the different design actors so that they can formulate a coherent and effective response to the objective that has been set. Unfortunately, the produced simulation models are not always relevant. Indeed, the models suppliers do not always understand the principal objective of the simulation. One solution explored at xxx is to introduce a new actor within the design cycle: the simulation architect, in charge to reformulate the simulation objective coming from the systems engineering world. The xxx project explored how integrate this actor in industrial processes, with the support of a methodology and tools adapted to this new actor. A new project was then launched in order to improve methodologies and tools by focusing on the agile process and margin management.

In parallel, the strong interaction between the different subsystems of current systems leads to an increasing complexity of their design. These interactions require the expertise of different actors and disciplines, especially when producing simulation models. In order to quickly take into account the interactions between simulation experts, simulation architects specify a simulation architecture that has to meet the needs of system architects as closely as possible (Sohier et al. 2019; Brunet et al. 2019). As this simulation architecture has to be provided by many simulation models, simulation architects request the different experts for them. As building a simulation model is time and cost consuming, given the significant amount of numerical simulations used in the industry, the question of being able to reuse existing models has become obvious. While many simulation models are now stored, this massive amount of data is nowadays often under-exploited because of the difficulty to retrieve a model matching the needs.

Simulation-based design challenges

In a context where the duration of design cycles is becoming shorter and shorter in order to face international competition, regardless of the industrial sector, the use and reuse of simulation models are becoming critical for designing ever more complex and innovative products. Indeed, Simulation-Based Design (SBD) processes currently represent the core activity to validate design studies at each step of the design process (Sibois & Muhammad 2015a). Furthermore, Simulation Life Cycle management allows to enhance product development and its decision-making process (Popielas et al. 2010). Rapid advances in computer-aided engineering have resulted in the development of numerous simulation tools to model and evaluate the behavior of real-world systems. Moreover, with the strong expansion of Model-Based System Engineering (MBSE) in the industrial context (Graignic et al. 2013), many manufacturers have adopted SBD approaches (Sibois & Muhammad 2015a), be it for concepts or architecture exploration, multi-disciplinary model optimization or validation, or also to ease model generation and reconfiguration (Paredis et al. 2001). One objective that SBD regularly tackles is a more comprehensive exploration of solutions (concepts, architectures, 3D …) (Singh, Das & Kumar 2013). Indeed, during the conceptual design stage, where detailed parameter are still unknown, it is important to quickly evaluate many various alternatives with coarse simulation models (Cross & Roy 1989). Regarding increasingly multi-disciplinary current products, which integrate new technologies (electronics, communication, Internet of Things…), SBD has become essential to provide designers with flexible and efficient means to computationally explore the interactions between various domains and achieve optimal system performance (Negendahl 2015). Then once a concept or architecture has been selected, simulation models are usually exploited for optimization (Peri & Campana 2005). Another important design step where they can often be used for cost-saving is the critical evaluation/verification phase (Sargent 2010), during which stakeholders validate the system to be developed thanks to virtual "proofs", to avoid time-consuming and costly physical prototypes. Finally, the last use of simulation models is the capitalization (Simulation Data Management Working Group 2014) since MBSE approaches and most industries aim at supporting model reuse (Sibois & Muhammad 2015b). Common industrial objectives, when no rupture technology is at stake, are to reuse 80% of their existing bases, innovation addresses only a limited part of the current developed products. Today, most simulations are computed according to an intended use of a model at a given time but might always be used later for design modifications or as inputs for advanced simulations. Therefore, to improve the efficiency of the simulation models use during the design process, it is important to study a way to store knowledge of the domain of interest with the modeling data, and then to integrate them in some knowledge-based systems to facilitate their reuse and then their automatic definition to speed-up the simulation activity (Sibois & Muhammad 2015b). Archiving the simulation models is essential for later reuse, adaptation, or reference. This is usually ensured by the engineering infrastructure, like PLM or repositories, while models are developed (on-going), then an inventory of existing simulation models has to be done. Leveraging the effort to develop simulation models in future applications is only possible if the perceived effort to reuse is substantially lower than a new development. To become an asset, existing simulation models have to become like parts of existing libraries. For that, an appropriate indexation is mandatory with a summarized standardized description of each model. The effort to fill up such a form is limited, acceptable compared to the expected reuse benefit.

Research issue

During the design process, system architects ask simulation architects to design the appropriate simulation architecture whose simulation results will be able to answer their question. For this purpose, the system architects define the corresponding simulation models to be provided. They either specify the set of models that the model providers (simulation experts) need to develop, test and deliver within weeks, or they search for similar models in an existing database, which they can then quickly customize to meet their needs and reduce lead times.

As a result, capitalization and reuse become critical, in order to reduce development time. In this context, we propose the development of a simulation model search engine that will support the common activities of system and simulation architects. In this paper, we focus on the presentation of our first works around the search engine of simulation models representing the behavior of a given system function.

Background

Related existing approaches

Common search engines usually relate to Web-based applications and aim at supporting special purpose local-search and contextual advertising. In this context, they are composed of three main elements: a Web crawler (computer program that browses the World Wide Web in a methodical, automated manner), an indexer (that indexes the documents collected by the crawler) and the searcher (to solve the user queries by using the generated index and other components for performance improvement) (Marín et al. 2017) . The indexer and searcher directly interface to the concerned database.

Among the existing tools, Product Data Management (PDM) and Product Lifecycle Management (PLM) tools manage the structured storage in a database of all product information generated during all phases of the product lifecycle (Bergsjö et al. 2007). They have been developed to ensure the capitalization of the digital design chain, but only manage encapsulated data, not allowing to search for models according to given physical attributes (domain, inputs/outputs...).

A solution lies in the enrichment of these stored data with semantic-related ones. Knowledge Organization Systems (KOS) are information organization mechanisms used in many engineering activities (Pieterse & Kourie 2014), and in particular during design (for capturing, indexing, reasoning, processing, retrieving and decision making from information) to formalize and represent knowledge with faithful semantics capable of ensuring a correct understanding of the system data that is common to all the actors involved (Sowa 2000). Among the KOSs, there are different systems (Pieterse & Kourie 2014), of increasing spectrum: dictionaries, taxonomies, thesauri and ontologies. A *dictionary* is a list of words in one or more languages, usually in alphabetical order, explaining the meaning of each word. A *taxonomy* is a hierarchically organized collection of elements and their attributes, grouping the elements in a domain into categories and sub-categories composed of several levels of refinement. A *thesaurus* is a collection of controlled and standardized terms representing concepts in a selected field, in order to specify the semantic relations of equivalences, hierarchical (generic/specific), associative and contrast relations between its elements. An *ontology* is like an extension of a thesaurus since it contains more detailed information about concepts, deeper hierarchical levels of concepts and richer relationships between concepts in a more formal structure. Finally, the semantic expressivity of ontologies exceeds that of other KOSs due to the availability of inference rules. As a result, in order to define a formalization and representation of knowledge with faithful semantics to ensure a correct understanding of the system data that is common to all actors involved, many researchers in the knowledge-based engineering (KBE) community are turning to ontologies (Sowa 2000; Sure & Studer 2003). An ontology is an explicit specification of a conceptualization, which formally represents knowledge as a set of concepts in a domain and the relationships between pairs of concepts (Gruber 1995). Indeed, as ontology is a powerful tool for manipulating semantics, it is used for many design activities (Brandt et al. 2008; Kumar 2008; Štorga, Andreasen & Marjanović 2010): capture, indexing, reasoning, processing, retrieval, but also decision making (Rockwell et al. 2009). Finally, it can be used to automatically support processes related to a search engine. As a result, we propose develop a simulation models search engine from their related system function based on an ontological approach.

Function definition, description and identification

A function is a "black box" transformation of system inputs to produce outputs using resources and respecting control rules. Thus, functions describe what the system does (the “what") regardless of how they will be implemented (the “how”). The SADT method, which inspires the IDEF0 standard (IEEE 1998), provides a graphical representation of a function (Figure 1).

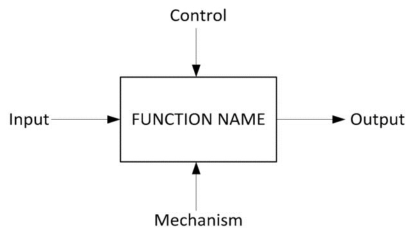


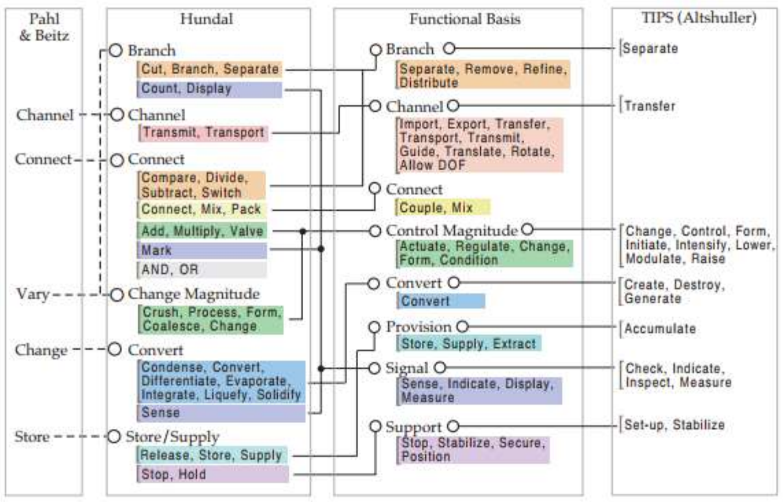
Figure 1: Function IDEF0 representation

Chakrabarti et al. define a function by its different views that can be synthesized into three categories (Chakrabarti et al. 2005): (i) the system view (the function is described by the parameters of the equipment, or even their interpretation at the environment level), (ii) the semantic view (the function is defined by input and output flows), the function here is considered as a change of state of an object or a system. (iii) the syntactic view (the function uses an informal (e.g. noun + verb) or formal (e.g. mathematical transformation) syntactic representation). Szykman et al. (Szykman, Racz & Sriram 1999) have defined the attributes required to qualify a function (Table 1).

Table 1: Description of the attributes of a function according to (Szykman, Racz & Sriram 1999)

Name of the function	unique character string.
Type	refers to a generic function class, included in the taxonomy.
Documentation	character string used to describe the function.
Method	character string (or a file path or Web URL), this element differs from documentation in that methods are intended to include computer-processable information (such as a computer program, code fragment, rules, constraints).
Input and output streams	refers to data structures representing input and output streams.
Subfunctions	list of references to other functions, allowing a function to be broken down into several subfunctions.
Sub-function of	defines the parent reference function.
Referring_artifact	artefact to which the function refers.

In order to describe the functions, other authors have proposed to index them in some semantic classes. Stone and Wood (Stone & Wood 1999) developed a functional basis by exploring the three previous works of Pahl and Beitz, Hundal and Altschüller. Pahl and Beitz have classified functions under 5 classes and flows in 3 classes, their classification remains of very high level. The set of functions and flows will be more detailed with Hundal. Hundal has increased the number of reference functions into 6 classes with more specific functions. However, Hundal's taxonomy does not have an exhaustive list for mechanical functions. The TIPS system developed by Altschüler and inspired by that of Pahl and Beitz, proposes a set of 30 functional descriptions to describe all mechanical functions. The functional base developed has extended the set of function classes to 8 classes and gives a definition to each of the classified functions. It encompasses all the functions of the previous work but requires that a function/flow pair be expressed as "verb + object" (with functions as verbs and flows as objects). Thereafter, Hirtz et al. have proposed a reconciled improved functional basis, based on the work presented above by Szykman et al. and Stone and Wood (Hirtz et al. 2002). Later on, Nagel, Stone, and McAdams have developed an Engineering Biology thesaurus based on the work of the University of Toronto, Indian Institute of Science, and Oregon State University (Nagel, Stone & McAdams 2010). Each of these three universities has developed a thesaurus of functional terms and names, with classes of primary functions associated with secondary and tertiary classes and their correspondences in the biological field, to highlight bio-inspiration. An example of some previous classifications is provided in Figure 2.


Pahl & Beitz
Hundal
Functional Basis
TIPS (Altshuller)
Channel
Connect
Vary
Change
Store
Branch
Cut, Branch, Separate
Count, Display
Channel
Transmit, Transport
Connect
Compare, Divide, Subtract, Switch
Connect, Mix, Pack
Add, Multiply, Valve
Mark
AND, OR
Change Magnitude
Crush, Process, Form, Coalesce, Change
Convert
Condense, Convert, Differentiate, Evaporate, Integrate, Liquefy, Solidify
Sense
Store/Supply
Release, Store, Supply
Stop, Hold
Branch
Separate, Remove, Refine, Distribute
Channel
Import, Export, Transfer, Transport, Transmit, Guide, Translate, Rotate, Allow DOF
Connect
Couple, Mix
Control Magnitude
Actuate, Regulate, Change, Form, Condition
Convert
Convert
Provision
Store, Supply, Extract
Signal
Sense, Indicate, Display, Measure
Support
Stop, Stabilize, Secure, Position
Separate
Transfer
Change, Control, Form, Initiate, Intensify, Lower, Modulate, Raise
Create, Destroy, Generate
Accumulate
Check, Indicate, Inspect, Measure
Set-up, Stabilize

Class	*Secondary*	*Tertiary*	*Bio Correspondents*
Channel	Export		Bind, block, breakdown, excrete, inactivate
	Transfer	Transport	Circulate, conduct, diffuse, pump
		Transmit	Communicate, transduce
	Guide	Translate	Synthesize, transcribe
Connect	Couple		Extend, link, overlap, stretch
		Link	Activate, bind, project
	Mix		Contract, exchange, fragment
Control Magnitude	Stop	Inhibit	Cover, destroy, inhibit, surround
Convert	Convert		Decompose, degrade, develop, grow, mutate, photosynthesize
Provision	Store		Convert, deposit, photosynthesize
		Collect	Breakdown, concentrate, digest, reduce
Support			Develop, wrap
	Stabilize		Bind, connect

Figure 2: Example of functional classifications, proposed: *on the left,* by Stone and Wood (Stone & Wood 1999) in relation to Hundal's taxonomy, that of Pahl and Beitz, and the functional description of TIPS; and *on the right*, the Engineering Biology functions thesaurus proposed by (Nagel, Stone & McAdams 2010).

Concerning the search/identification of functions to support design activities, a generic model (SAPPhIRE Model) has been developed by Chakrabarti and Srinivasan (Chakrabarti et al. 2005; Srinivasan & Chakrabarti 2011) to allow the causality of natural and artificial systems to be modelled, by structuring the information in a database system of both domains (natural and artificial). Indeed, this model represents a causal description language (State - Action - Part - Phenomenon - lnput - oRgan-Effect) for natural and technical systems, which describes the analysis and synthesis of technical systems and the corresponding design results (Srinivasan and Chakrabarti 2011). This model coupled with databases implemented in a software (IDEA-INSPIRE) facilitates then the automated search for functions and associated solutions.

Simulation model definition, description and identification

A simulation model is a numerical transfer function that makes it possible to represent a physical phenomenon (control law, state machine, thermal phenomenon, etc.) of a system with a certain degree of approximation, in a given formalism (language) and tool (software environment). It represents a behavior, i.e. the way in which a function is realized by the structure of a component. As a result, behavior is specific to the levels at which the function and structure of a device are defined.

A physical simulation model is composed of parameters, ports and variables and behavior (equations). The variables characterize the ports but can also be local (i.e. variables internal to a computer program). The physical quantity associated with these different features is described by a physical dimension and/or a unit. Oren & Zeigler are the first to propose a concept of structure and different characteristics to describe a simulation model, as well as definitions for each term (Ören & Zeigler 1979). Thus a simulation model is defined by its structure (static/dynamic), its inputs/outputs, its initialization and ending conditions, its interpretation and display configurations. More recently, Sirin et al. define the notion of Model Identity Card (Sirin et al. 2015). MIC is an "identity card" for simulation models. The motivations for the creation of this concept are, among others: (i) the formal description of simulation models, (ii) a format for communication and exchange of information between the different actors: tool for collaboration, (iii) it allows to specify a need for models to compose future simulation architectures; (iv) it aims at becoming a standard for describing simulation models (Brunet et al. 2019; Sohier et al. 2019). The MIC is presented in the form of a tree that contains information specific to the simulation model it describes. It is broken down into 6 categories (Table 2).

Table 2: MIC description

General information	name and version of the model, the date of creation, possibly the name of its author and its software license.
Implementation	concerns all the details of the implementation: name of the development language, compiler, integration constraints, possible dependencies (external libraries),...
Validation and Verification	these terms define the procedures for checking models: verifying that they match the physical systems that they must reproduce digitally.
Port(s)	a simulation model integrates ports, which themselves contain one or more variables that interface between the external connections and the internal functioning of the model.
Parameter(s)	a simulation model is usually developed to be reused with different values. It therefore contains parameters that must be filled in during its use. A default value can sometimes be provided.
Physical definition	formal indications on the internal composition of the model: its physical domain, its order, the linearity or not of its equations, an indication as to its speed of execution (time scale), etc.

Concerning the simulation model searching, some researchers have used optimization algorithms based on genetic algorithms to find patterns with similarities (Kessentini, Langer & Wimmer 2013). However, this work is limited to the identification of similar models. Similarly, Fontaine & Hammami compares two pseudo-MIC (containing ports and their related variables with units, parameters with units if avalaible, and internal local variables) by dissimilarity measures between graphs (Fontaine & Hammami 2018). Even if this approach is efficient for model composition, it does not discriminate sufficiently between models, as it does not contain any semantic information related to the behavior covered, to allow an efficient model search when designing a simulation architecture.

Interactions modeling

The functional architecture describes the functions of the system and the links between them. The links/interactions between functions can be of a logical (AND/OR) or temporal nature (sequential, parallel, iteration). A simulation architecture is an assembly of numerical simulation models, connected to each other via their ports. Be it for functional or simulation architecture, the links or interactions between their artefacts are defined by data flows circulating in (input/output) ports. The ports integrate one or more variables that interface between the external connections and the internal functioning of the model or function. They can be defined in a given direction: input, output, input/output. The data flow allows the exchange of data, material, energy, signals, etc, from one function to another or between two models or components. The control flow allows to trigger a function (logical sequence), but does not transmit data flow.

(Szykman, Racz & Sriram 1999) have described the attributes that allow to represent a flow: its name, its type (generic flow class from a flow taxonomy), its documentation, its source and destination (physical artifacts corresponding to the sources and destinations of the flows for a given function), its properties and the reference functions it relates). Then they proposed a generic taxonomy of flows to reduce ambiguity at the modeling level. We focus the state-of-the-art only on data flows (unlike control flows) as they are intrinsic to the I/O of functions and models (the two artefacts we are interested in). Stone and Wood (Stone & Wood 1999) based their work on three existing studies from Pahl & Beitz, Hundal and de Altshuller, to introduce a library of flows, with the following three classes of flows: Energy, Matter and Signal (Figure 3).

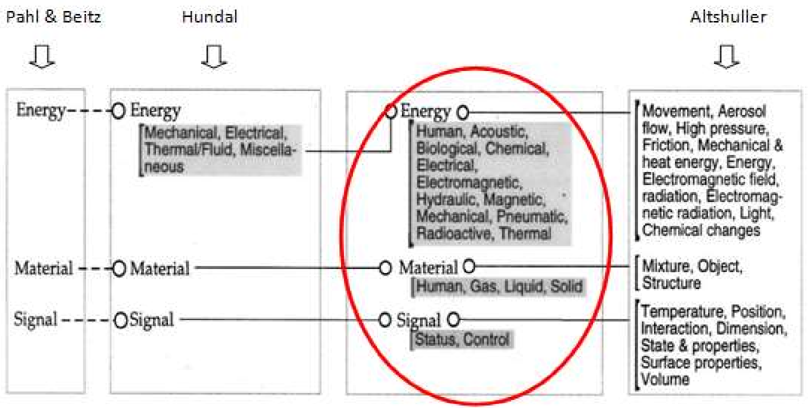

Pahl & Beitz
Hundal
Altshuller
Energy
Material
Signal
Energy
Mechanical, Electrical, Thermal/Fluid, Miscellaneous
Material
Signal
Energy
Human, Acoustic, Biological, Chemical, Electrical, Electromagnetic, Hydraulic, Magnetic, Mechanical, Pneumatic, Radioactive, Thermal
Material
Human, Gas, Liquid, Solid
Signal
Status, Control
Movement, Aerosol flow, High pressure, Friction, Mechanical & heat energy, Energy, Electromagnetic field, radiation, Electromagnetic radiation, Light, Chemical changes
Mixture, Object, Structure
Temperature, Position, Interaction, Dimension, State & properties, Surface properties, Volume

Figure 3: Comparative flows taxonomies from (Stone & Wood 1999)

This flow library allows the representation of a flow at several levels of abstraction. In each flow class, the flows are divided into base (e.g. Energy) and sub-base flows (e.g. Electrical). Finally Hirtz et al. merged all these previous works to propose a more completed reconciled taxonomy at three levels: primary, secondary and tertiary flows (Hirtz et al. 2002). The primary flows are matter, signals and energy. The "Matter" class includes five other specified secondary categories with an extended list of tertiary categories. The "Signal" class includes two secondary categories with an extended list of tertiary categories. The "Energy" class includes 13 specified secondary categories and an expanded list of tertiary categories with associated powers described in force and flow analogy.

Finally, many existing works describe taxonomies related to simulation functions and models, but no model search engine developed has relied on the identification of the function supporting the behavior described in a simulation model to more appropriately target the ad hoc model sought.

Model search engine proposal

Approach global description

In our approach, we propose to support the design of the simulation architecture from the functional architecture of the system (describing the functions to be realized by the system), by enabling the search for existing simulation models (via their MIC). As a result, the proposed simulation model search engine aims at filtering the MICs of existing models from the system functions associated with the behavior represented in the model being searched for. This search engine will then take as inputs the system function defined by the system architects and select the existing simulation models that meet the need, linking the MICs to the functions associated with the models being searched for (via the description of the expected model behavior).

For this purpose, we have coupled two perspectives: one based on a thesaurus of references function and one based on ports analysis. Both of them are based on ontology, in order to be able to define any semantic relationship between two terms or two concepts (identical, synonym, homonym, hyponym/hypernym and kind of).

Reference Functions-based approach. Starting from a functional architecture and a component architecture of the system to be evaluated, for which the system architect faces a question requiring simulation, a thesaurus of reference functions (RFs) is used to add the formal meaning of system functions, independently of the vocabulary used by the user to name it. Each system function, we will call User Function (UF) can then be allocated to several reference functions via their FIC. As a result, one or several FICs have been associated to each model identity card (MIC) of existing simulation model. Based on the RFs thesaurus, RFs will be allocated to the FICs defined in the functional architecture. The search engine will then consist in searching all existing models (via their MICs) having the same RFs (or the semantically most similar ones) as those of the functional architecture of the system to support simulation architects to retrieve relevant existing models to define the appropriate simulation architecture.

Ports Analysis-based approach. In this approach, the search engine will estimate the similarity distance between the ports of the UF requested by the system architect and those of the function stored in the database, using ports attributes (domain, direction, flow), in order to propose the most relevant models. The similarity between ports is also evaluated using the reference port concept. The reference port approach contains an unequivocally meaning of a port. A port can be associated to several reference ports (as it can be a “multiport”).

Ontology definition

For the sake of representation simplicity, we use the UML language, and in particular the UML classes and corresponding relationships in the following figures, to represent our ontological concepts and their dependencies.

Ontological concepts.

The “Flow” concept (related to the flow between two ports) is characterized by the following four attributes: Flow_Name, Nature (e.g. material, energy, signal), Type (e.g. solid, mechanical, control) and Detail (e.g. composite, angular velocity, analog). Based on this concept, we have created a database of reference flows, inspired by the library proposed by (Hirtz et al. 2002) and supplemented by other relevant flows.

The "Port” concept is limited in our ontology to the three attributes necessary for the search engine application, even if it can actually include more features (notably variables or other ports as originally defined in the MIC) : Name, Domain, Direction and Flow, as, each port is identified by the type of flow it carries or can carry (at this stage, we do not differentiate these two cases).

The “User_Function” (UF) concept relates a system function defined in the functional architecture. Its attributes are defined by an identity card: by analogy to the MIC for models, we introduce the identity card of a function (FIC).

Concerning the Function Identity Card “FIC” concept, its attributes have been chosen from the aforementioned existing literature on the description of a function and from the various functional models (UML/SysML, APTE, SADT, EFFBD, etc.): Function_Name, Control_Data, On_what_it_acts, Main_Function, Related_State_machine, To_whom_or_what_it_provides_service, Allocation_to_the_component, Allocation_to_the_Use_Case, Port.

The concept of "Reference Function" (RF) refers to the categorized function used to overcome ambiguity in the vocabulary/terms used by designers to describe system functions (UF). Its attribute are: Function_Name, Function_Description, Function_ Type (e.g. combination, distribution, assembling, transmission), Port

The “Reference Port” (RP) concept is similar to Reference function for ports. It allows ports to be categorized unequivocally regardless of the terms used by designers. It includes the attributes: Name, Domain, Direction and Flow .

The Model Identity Card (MIC) concept is defined with its original attributes defined in Table 2.

Ontological dependencies. To develop the search engine and in particular the different algorithms constituting it, we relied on the relations between the previously defined ontological concepts (Port, Flow, User_Function, FIC, Reference_Function, MIC), as defined in Figure 4. Indeed, the FIC characterizes the User_Function (from the functional architecture or a model database). Each FIC have to be (automatically or manually) allocated to one or more Reference_functions. FIC, MIC and RF contain at least two ports, thus composition links have been established between the FIC/MIC/RF and the Port concept.

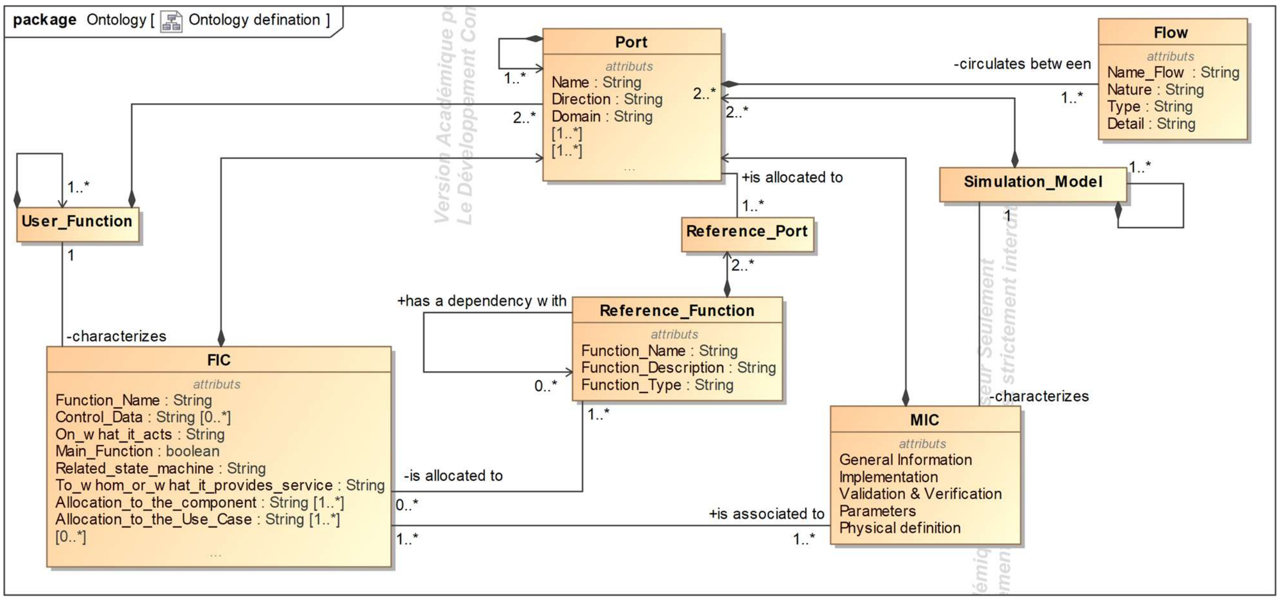


Figure 4: Defined ontology

Hypotheses

The developed search engine is based on a certain number of hypotheses/pre-requisites. Indeed, it assumes the existence of several or one database(s). The first one contains the existing User Functions

(UF) and their link to existing simulation models via a link between their respective FIC and MIC. The second one concerns the existing simulation model with their related MIC. We assume, for this first development, that the FICs and MICs are composed of defined ports (domain, direction, flow). Similarly, the FICs in these databases have been allocated to one or more reference functions. This allocation can be manually performed either by the simulation architects or by the system architects, in compliance with the User_Functions (UF) of the functional architecture of the system. It can also be supported by the search engine to allocate them automatically or semi-automatically (by suggestion to be validated), via a *Port_Port algorithm* applied to the FIC and RF concepts (as detailed hereafter), allowing to compare two ports of different concepts according to the different attributes of the ports, and a second *FIC -RF* algorithm allowing to globally compare 2 concepts estimate a similarity metric between RFs of two FICs. Initially, existing simulation models in the database have been associated via their MIC to the corresponding FIC(s) of the UF representing their behavior. Reference functions have been allocated for each FIC (FIC_A) of the User_Functions (UFs) contained in the functional architecture provided by the System Architect , The objective of the search engine is then to identify if there are some existing simulation models that have been previously associated to one or several FIC(s) (FIC_B) similar to the FIC(s) (FIC_A) of the UF(s) contained in the functional architecture, which could respond quickly to its need, in order to limit the development of new simulation models and thus reduce design time and costs.

Algorithms & similarity metrics

Reference Functions-based search algorithm. In this algorithm, the aim is to search in the database for relevant existing models, by identifying the associated FIC-Bs that are allocated to the same RFs as the FIC-As requested by the simulation architect. For this purpose, we have developed a *RF-RF algorithm* applied to FIC-A and FIC-B, to compare two reference functions allocated to these 2 concepts and a *FIC-RF_FIC-RF algorithm* to compare a set of RFs of a concept with the set of RFs of another concept, applied to FIC-A and FIC-B, to estimate a similarity metric (RF_Score) between the FIC-Bs found in the database of existing models with those specified in the simulation architecture (FIC-As).

The *RF-RF algorithm* allows to compare two RF allocated to two different concepts, based on their attributes. The inputs of this algorithm are the attributes of the $RF1$ and the $RF2$. The output of this algorithm relates to their semantic relationship (Identical, Hyponym/Hypernyme, Homonym, Synonym, KindOf) between $RF1$ and $RF2$. The algorithm identifies the semantic dependencies between the two RFs based on their attributes based on the Table 3 correspondences.

Table 3: Semantic dependencies identification

Identity	$RF1\_attributes = RF2\_attributes$
Hyponymy/Hyperonymy RF1 is hyponym of RF2 RF2 is hypernym of RF1	$(RF1\_name = RF2\_name)\ AND\ (RF1\_ports \in RF2\_ports)$
Homonymy	$(RF1\_name = RF2\_name\ )\ AND\ (RF1\_ports\ \notin\ RF2\_ports)$
Synonym	$(RF1\_name \neq RF2\_name)\ AND\ (RF1\_attributes = RF2\_attributes)$
KindOf relation	$(RF1_{name} \neq RF2_{name})\ AND\ (RF1\_Function\_type = RF2\_Function\_type)\ AND\ (RF1\_ports \in RF2\_ports)$

Focusing on the *FIC-RF_FIC-RF algorithm,* it aims at finding the existing model(s) (via their associated FIC-B) closest to the UF analyzed (via its FIC-A) by the simulation architect. Since this determination by filtering will be based on the comparison between the respective reference functions allocated to each of these FICs. Since the previous *RF-RF algorithm* gives a qualitative information on the semantic relationship between these RFs, we propose for example that the user can define a reference value for each of these relationships (Identical, Hyponym/hypernym, Synonym, KindOf), in order to be able to rate a global adequacy value between 2 FICs, based on the normalized value of the sum of all reference values associated with all semantic relationships existing between all the RFs of the 2 FICs being compared. Another example of quantitative similarity factor related to the hyponym/hypernym relations could be the following RF_Score:

$$RF\_Score(FIC-A, FIC-B) = \frac{1}{nb_{RF}(FIC-A)} \sum_{i=1}^{nb_{RF}(FIC-A)} \frac{1}{l(RF_i(FIC-A), RF(FIC- \ ))} \tag{1}$$

Where:

- $nb_{RF}(x)$: number of reference functions for the FIC x
- $RF_i(x)$: ith Reference Function for the FIC x
- $l(x,y)$: calculate the number of levels before reaching a common reference function from the references functions x and y

Ports Analysis-based search algorithm. In this algorithm, the aim is to search in the database for relevant existing models, by identifying the associated FIC-Bs that have the same ports as the FIC-As defined by the system architect. For this purpose, we have developed two sub-algorithms: (i) the *Port_Port algorithm* to compare two ports of different concepts according to their respective attributes (nature, domain, direction, flow_name, etc.), (ii) the *FIC-Ports_FIC-Ports algorithm* to estimate a similarity metric based on ports analysis ($Interface\_Score$) between the FIC-Bs associated to existing models in the database with those defined in the functional architecture (FIC-As).

Considering the *Port_Port algorithm*, the inputs are the attributes of each port (Name, Direction, Domain, Flow with related attributes: Flow_Name, Nature, Type, Detail) to be compared, and the output is a port_port similarity factor $Intscore$ that estimates the level of resemblance between two ports. As the similarity can address any of the port attributes, we propose to the user to filter the attribute(s) to consider, by weighting each attribute similarity rate, in order to compute the global port_port similarity factor $Int_{score}(P_1, P_2)$ defined as follows:

$$Int_{score}(P_1, P_2) = \sum_{i=1}^{7} \alpha_i \ attribut_i\_similarity\_rate \tag{2}$$

Where $attribut_i\_similarity\_rate = \begin{cases} 1 & if\ port\_1.attribut_i = port\_2.attribut_i \\ 0 & else \end{cases}$ (3)

The *FIC-Ports_FIC-Ports algorithm* aims at providing a similarity score between two FICs based on the similarity factors ($Intscore$) of their respective ports. Such a similarity score can be defined as the following $Interface\_Score$:

$$Interface\_Score(Fic-A, FIC-B) = \frac{\min(nb_{port}(FIC-A), nb_{port}(FIC-B))}{\max(nb_{port}(FIC-A), nb_{port}(FIC-B))} \cdot \sum_{i=1}^{nb_{port}(inpSF)} \frac{Int_{score}(Port_i(FIC-A), Port_i(FIC-B))}{nb_{port}(FIC-B)} \tag{4}$$

where:

$n_{bport}(x)$: number of ports of the FIC x
$Port_i\ (x)$: ith port of the FIC x
$Int_{score}(p_1, p_2)$: similarity factor between the ports p_1 and p_2

The ratio of the minimum number of ports over the maximum number of ports allows to minimize the *interfaceScore* when the functions have not the same number of ports.

Inference engine implementation and case-study application

For the implementation of this first interference engine, we have chosen the full naïve identification (without any initial filter to reduce the exploratory space) of the most relevant MIC-Bs (related to existing simulation models) regarding a given FIC-A included in the functional architecture of the System, to support Simulation Architects to build rapidly and efficiently the appropriate simulation architecture.

For the demonstration, we have implemented the previous two similarity scores related respectively to the reference function-based and port analysis-based algorithms: RF_Score and $Interface\_Score$, on a case-study. The global score is done by calculating the average between the two previous scores, but can be weighted, according to the designers' preferences. Then, as each FIC is associated to a MIC, the existing MICs are then ordered by ascending order of score. Simulation architects can then choose the MIC that best matches the simulation request using this score and their own expertise.

Figure 5 presents an extract of the Reference Functions tree database, each node allows to identify a function subtype, and illustrates the RF_Score based on the calculus of the number of levels before reaching a common reference function from the references functions of a FIC-A and a FIC-B.

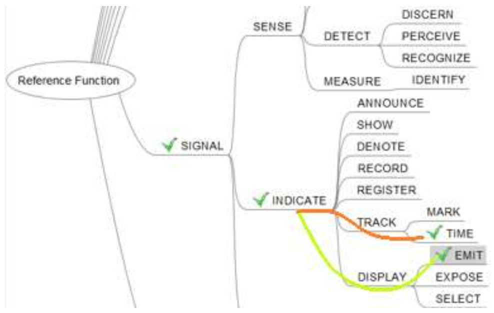


Figure 5: Reference Functions tree with levels identification for the RF_Score.

The case study is the stop at traffic light for an autonomous vehicle (Sohier et al. 2019). The system models were build using the Nine Views Matrix methodology, including the functional architecture detailed in Figure 6. This architecture is stored in a xml file that contains information about the functions, ports, and links between ports. Moreover, one or several RF have been allocated to each FIC of UFs.

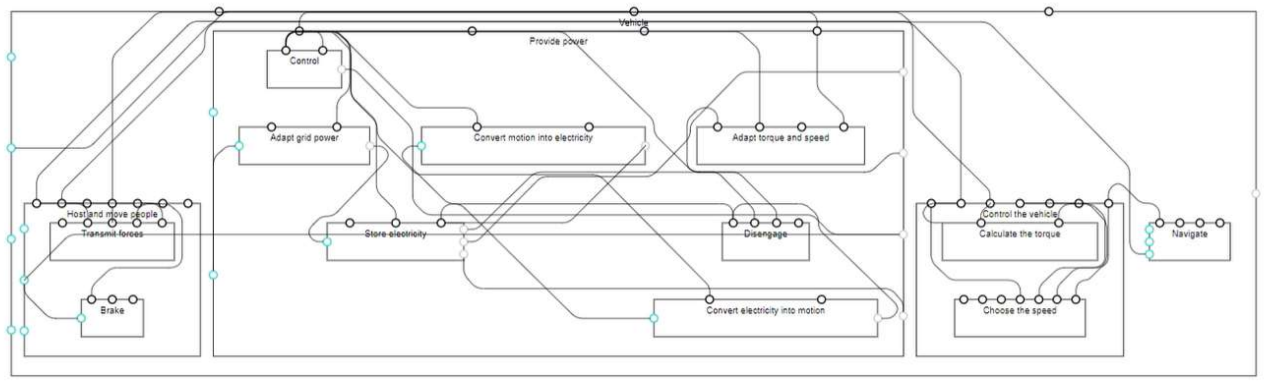
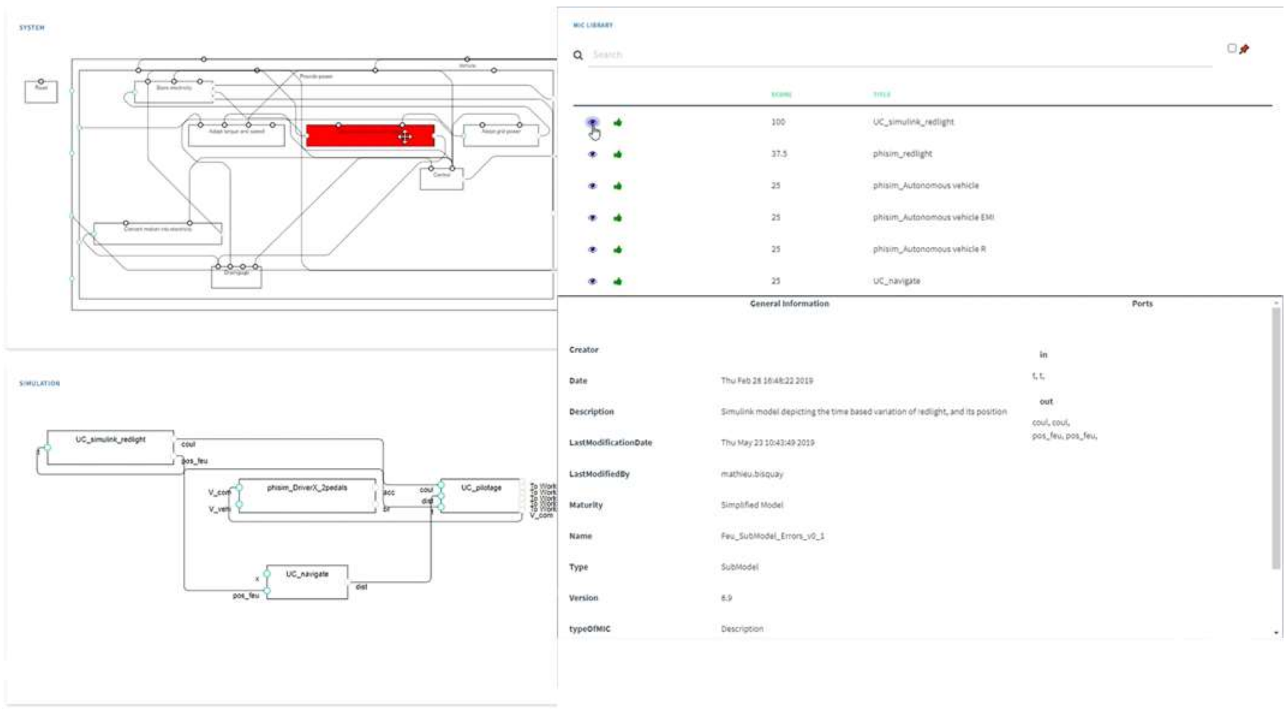


Figure 6: Functional architecture of the autonomous vehicle case-study

The human machine interface is realized by a web application, named SimArT and contains the loaded functional architecture. When a specific function of this architecture is selected, the server, launches the inference engine to calculate the score between the selected function and each existing FIC in the database. The results is stored in a list of existing UF. Then, a list of corresponding MIC is created. This MIC list is ordered by ascendant order of score results on the right panel and it is possible to display more information about a given MIC, for example the name of the creator, or the used tools name (Figure 7).



	SCORE	TITLE
	100	UC_simulink_redlight
	37.5	phisim_redlight
	25	phisim_Autonomous vehicle
	25	phisim_Autonomous vehicle EMI
	25	phisim_Autonomous vehicle R
	25	UC_navigate

General Information

Creator	
Date	Thu Feb 28 16:48:22 2019
Description	Simulink model depicting the time based variation of redlight, and its position
LastModificationDate	Thu May 23 10:43:49 2019
LastModifiedBy	mathieu.bisquay
Maturity	Simplified Model
Name	Feu_SubModel_Errors_v0_1
Type	SubModel
Version	6.9
typeOfMIC	Description

Ports

in
t, t,
out
coul, coul,
pos_feu, pos_feu,

Figure 7: Interface of the demonstrator with the identified MICs and related details.

Thereafter, for each function, it is possible to associate a simulation model, in order to automatically build the simulation architecture by connecting compatible ports in to them (Figure 8).

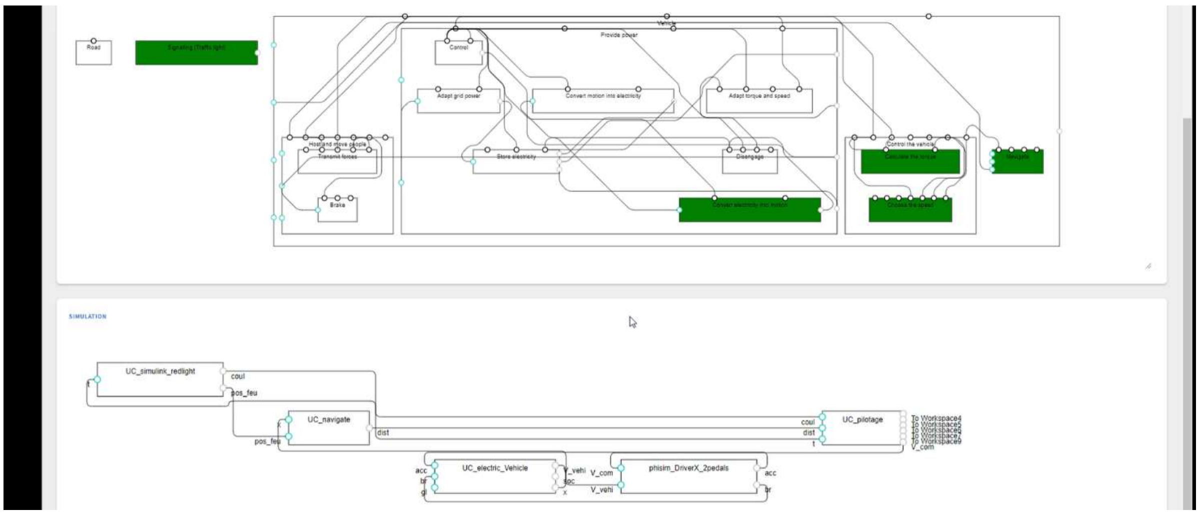


Figure 8: Simulation architecture generated from the functional one thanks to the search engine.

Discussion

This case study validates this search engine for a small number of system functions. The execution takes a few millisecond for a database of 50 system functions. The scaling up should be evaluated with a larger database. In that case, in order to improve the execution time, different solutions are considered. The first one is to optimize the similarity scoring functions, by example by using graph edit distance (Fontaine & Hammami 2018). Another solution could be the use of neural network to perform a first elimination of existing functions that not match the candidate function. Usage of such a methodology involve a classification of functions regarding some attributes for example the kind of physics in ports.

This first step allows us to develop a demonstrator facilitating the identification of existing simulation models/MIC for one FIC. The HMI generated will support further developments to fill the gap of the combinatorial explosion due to the full naïve identification, maybe by introducing some filters to impose some specific attributes for the MIC displayed. Future work will address another usage scenario by searching for an existing simulation model/MIC from a MIC specified by the simulation architects, but also the search of a MIC/model related to more than one FIC/User function, and conversely the study of the case of a FIC/User function is associated to more than one MIC.

Conclusions

This paper focuses on the development of a search engine for simulation models from a system function, based on ontology, in order to facilitate the rapid construction of a simulation architecture. The ontology proposed allows overcoming ambiguity in the vocabulary/terms used by designers, while ensuring an efficient indexation means for simulation models capitalization and reuse. The inference engine implemented has been validated on a first case study (autonomous vehicle) and needs henceforth to be validated, or even optimized with more significant scenarios.

References

Bergsjö, D, Vielhaber, M, Malvius, D, Burr, H & Malmqvist, J 2007, 'Product Lifecycle Management for Cross-X Engineering Design', *The Design Society - a worldwide community*.

Brandt, SC, Morbach, J, Miatidis, M, Thei\s sen, M, Jarke, M & Marquardt, W 2008, 'An ontology-based approach to knowledge management in design processes', *Computers & Chemical Engineering*, vol. 32, no. 1–2, pp. 320–342.

Brunet, J-P, Sohier, H, Yagoubi, M, Bisquay, M, Lamothe, P & Menegazzi, P 2019, 'Simulation architecture definition for complex systems design: A tooled methodology', *International Conference on Complex Systems Design & Management*, Springer, pp. 153–163.

Chakrabarti, A, Sarkar, P, Leelavathamma, B & Nataraju, BS 2005, 'A functional representation for aiding biomimetic and artificial inspiration of new ideas', *AI EDAM*, vol. 19, pp. 113–132.

Cross, N & Roy, R 1989, *Engineering design methods*, Wiley Chichester.

Fontaine, G & Hammami, O 2018, 'Automatic Model Search for System Model Composition', *2018 IEEE International Systems Engineering Symposium (ISSE)*, IEEE, pp. 1–7.

Graignic, P, Vosgien, T, Jankovic, M, Tuloup, V, Berquet, J & Troussier, N 2013, 'Complex System Simulation: Proposition of a MBSE Framework for Design-Analysis Integration', *Procedia Computer Science*, Atlanta, Georgia, USA, pp. 59–68, viewed 14 January, 2014, <http://www.sciencedirect.com/science/article/pii/S1877050913000082>.

Gruber, TR 1995, 'Toward principles for the design of ontologies used for knowledge sharing?', *International journal of human-computer studies*, vol. 43, no. 5–6, pp. 907–928.

Hirtz, J, Stone, RB, McAdams, DA, Szykman, S & Wood, KL 2002, 'A functional basis for engineering design: Reconciling and evolving previous efforts', *Research in Engineering Design*, vol. 13, no. 2, pp. 65–82.

IEEE 1998, *IEEE SA - 1320.1-1998 : IEEE Standard for Functional Modeling Language - Syntax and Semantics for IDEF0*, Norme, New York, viewed 9 September, 2014, <http://standards.ieee.org/findstds/standard/1320.1-1998.html>.

Kessentini, M, Langer, P & Wimmer, M 2013, 'Searching models, modeling search: On the synergies of SBSE and MDE', *2013 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE)*, IEEE, San Francisco, CA, USA, pp. 51–54.

Kumar, PP 2008, *Design process modeling: Towards an ontology of engineering design activities*, Clemson University.

Marín, M, Gil-Costa, V, Bonacic, C & Inostrosa, A 2017, 'Simulating Search Engines', *Computing in Science Engineering*, vol. 19, no. 1, pp. 62–73.

Nagel, JKS, Stone, RB & McAdams, DA 2010, 'An Engineering-to-Biology Thesaurus for Engineering Design', pp. 117–128.

Negendahl, K 2015, ‘Building performance simulation in the early design stage: An introduction to integrated dynamic models’, *Automation in Construction*, vol. 54, Elsevier, pp. 39–53.

Ören, TI & Zeigler, BP 1979, ‘Concepts for advanced simulation methodologies’, *Simulation*, vol. 32, no. 3, Sage Publications Sage CA: Thousand Oaks, CA, pp. 69–82.

Paredis, CJ, Diaz-Calderon, A, Sinha, R & Khosla, PK 2001, ‘Composable models for simulation-based design’, *Engineering with Computers*, vol. 17, no. 2, Springer, pp. 112–128.

Peri, D & Campana, EF 2005, ‘High-fidelity models and multiobjective global optimization algorithms in simulation-based design’, *Journal of ship research*, vol. 49, no. 3, Society of Naval Architects and Marine Engineers (SNAME), pp. 159–175.

Philpotts, M 1996, ‘An introduction to the concepts, benefits and terminology of product data management’, *Industrial Management & Data Systems*, vol. 96, no. 4, MCB UP Ltd, pp. 11–17.

Pieterse, V & Kourie, DG 2014, ‘Lists, Taxonomies, Lattices, Thesauri and Ontologies: Paving a Pathway Through a Terminological Jungle’, *Knowledge Organization*, vol. 41, no. 3, pp. 217–229.

Popielas, F, Ramkumar, R, Tyrus, JM & Kennedy, B 2010, *Simulation life cycle management as tool to enhance product development and its decision-making process for powertrain applications*, USA.

Rockwell, J, Grosse, IR, Krishnamurty, S & Wileden, JC 2009, ‘A Decision Support Ontology for collaborative decision making in engineering design’, *Collaborative Technologies and Systems, 2009. CTS’09. International Symposium on*, IEEE, pp. 1–9.

Sargent, RG 2010, ‘Verification and validation of simulation models’, *Proceedings of the 2010 winter simulation conference*, IEEE, pp. 166–183.

Sibois, R & Muhammad, A 2015a, ‘Simulation Lifecycle and Data Management’, *Systems Engineering, VTT Technical Research Centre of Finland. Tampere: sn*, p. 22.

Sibois, R & Muhammad, A 2015b, ‘State of the art in modelling and simulation’, *Systems Engineering, VTT Technical Research Centre of Finland Oy. Tampere: sn*, p. 20.

Simulation Data Management Working Group 2014, ‘NAFEMS- What is Simulation Data Management’, viewed 9 November, 2020, <https://www.nafems.org/publications/resource_center/wt02/>.

Singh, AK, Das, A & Kumar, A 2013, ‘RAPIDITAS: RAPId design-space-exploration incorporating trace-based analysis and simulation’, *2013 Euromicro Conference on Digital System Design*, IEEE, pp. 836–843.

Sohier, H, Guermazi, S, Yagoubi, M, Lamothe, P, Maddaloni, A, Menegazzi, P & Huang, Y 2019, ‘A tooled methodology for the system architect’s needs in simulation with autonomous driving application’, *2019 IEEE International Systems Conference (SysCon)*, IEEE, pp. 1–8.

Sowa, JF 2000, *Knowledge representation: logical, philosophical, and computational foundations*, Brooks/Cole Pacific Grove.

Srinivasan, V & Chakrabarti, A 2011, ‘Development of a Catalogue of Physical Laws and Effects Using SAPPhIRE Model’, in T Taura & Y Nagai (eds), *Design Creativity 2010*, Springer London, pp. 123–130.

Stone, RB & Wood, KL 1999, ‘Development of a Functional Basis for Design’, *Journal of Mechanical Design*, vol. 122, no. 4, pp. 359–370.

Štorga, M, Andreasen, MM & Marjanović, D 2010, ‘The design ontology: foundation for the design knowledge exchange and management’, *Journal of Engineering Design*, vol. 21, no. 4, pp. 427–454.

Sure, Y & Studer, R 2003, ‘A Methodology for Ontology-Based Knowledge Management’, *Towards the semantic web: Ontology-driven knowledge management*, pp. 33–46.

Szykman, S, Racz, JW & Sriram, RD 1999, ‘The Representation of Function in Computer-Based Design’.