Article

Assessing Design Repository Search Effectiveness

Daniel Herrington *, Paul Beery and Kristin Giammarco

Systems Engineering Department, Naval Postgraduate School, Monterey, CA 93943, USA
* Correspondence: djherrin@nps.edu

Abstract: Reuse of design knowledge in systems engineering (SE) has been identified as a potential way to decrease design cycle times and development costs of engineering projects. Design repositories (DR) have been proposed and demonstrated as a way to store design knowledge in a reusable way. Yet little attention has been devoted specifically to the search methods used to identifying useful information within a DR, or how search method effectiveness can be measured or assessed. This paper demonstrates a method for assessing the effectiveness of search approaches within DR. The use of this method in assessing multiple search algorithms is demonstrated by example. The assessment method presented herein can also be used to identify the characteristics of repository data or intended reuse applications that affect the performance of search methods. These results can be used by engineers to inform selection of search methods based on anticipated future DR use cases.

Keywords: design repositories; design reuse; model based systems engineering; systems engineering

check for updates

Citation: Herrington, D.; Beery, P.; Giammarco, K. Assessing Design Repository Search Effectiveness. *Systems* **2022**, *10*, 178. https://doi.org/10.3390/systems10050178

Academic Editor: Edward Pohl

Received: 12 August 2022
Accepted: 27 September 2022
Published: 3 October 2022

1. Introduction

The engineering community is reaching, or has passed, a volume of available technologies so vast that human designers cannot be relied on to recognize or identify the entire set of feasible and valuable system designs that are possible given existing component and subsystem technologies. To deal with the increasing body of human knowledge and the limited mental capacity of humans, the scientific and engineering communities have self-organized into domains. Domain boundaries, while not insurmountable, split the total body of knowledge into more manageable pieces. Most engineers focus within a single or a few domains. Such focus has the effect of limiting the available source technologies and systems that individual engineers are familiar with or aware of, and therefore readily able to integrate in pursuit of new system designs. Systems engineering as a discipline offers a cross-domain perspective. However, limits of the human mind prohibit systems engineers from having the necessary depth of knowledge in every domain required to fully understand the available technologies in the domains relevant to the system of interest in a specific scenario.

Current SE practices and available tools offer a limited ability to reuse design knowledge, leading to a disconnect between the amount of knowledge leveraged during SE projects and the portion of the cumulative body of knowledge that may be applicable to the system of interest. This may result in additional rework, unrecognized design alternatives, or both. To recognize relevant system design information from past efforts, systems engineers currently need to recall applicable designs for reuse, scour existing models or documents that they have access to, or rely on basic search tools. To increase efficiency, a centralized location with flexible search methods is needed. DR are one available tool that can serve as a centralized store of knowledge in a form chosen to enable reuse, either by humans or computers.

A practical obstacle to knowledge reuse (KR) in SE is the collection and storage of design knowledge, including system context, from completed and partially completed projects. Design processes and the subsequent design artifacts vary by organization, change over time, and may not be well documented or easily retrieved. Model Based Systems

Engineering (MBSE) has increased the adoption of formal methods for the documentation of system designs. The SE community has turned to MBSE to manage complexity, maintain consistency, and assure traceability during system development [1]. With the adoption of MBSE within the SE community, digital system models created using MBSE tools create artifacts representing system designs. The use of MBSE artifacts to populate DR offers a potential way to leverage existing designs in support of reuse for future projects.

Research into DR, including their structure and functions, is active and ongoing. No research was identified that included assessment techniques or developed metrics for determining the efficacy of presented DR structures and functions. In order to determine the adequacy of existing DR approaches, guide future research into DR and assess the results of that research, such techniques and metrics are needed. This paper will provide an overview of existing research related to DR and outline their desired functionality. One required function, search (i.e., retrieval) is discussed, and a method is presented for assessing search methodologies within DR. The proposed methodology is demonstrated using three example search methods within an existing DR.

1.1. Background and Related Research

For the purpose of this article, there are three primary research areas that should be understood. These research areas are: systems engineering, design information reuse, and DR. Systems engineering includes the design and development of complex systems. Design information reuse is the process of leveraging existing knowledge from previous work to reduce the overall effort required to complete the design phase of a systems engineering project, and may reduce overall project risk by using existing and proven design solutions. DR are a means of capturing and storing design information to support later use or reuse.

1.2. Systems Engineering

Systems engineering is generally concerned with the application of systems thinking and systems theory in defining, exploring, and resolving problems. Utilization of defined and documented systems engineering processes are integral to that approach. SE processes are well established and supported by years of research and practice. This article will not directly address a specific SE process, instead it addresses one fundamental stage of the system life cycle: system design. The interdisciplinary nature of SE as a field often causes system design artifacts to span multiple engineering domains and represent design characteristics and information at different levels of maturity and fidelity. This article focuses on the opportunity to structure design information from previous SE efforts for reuse, the ways to search stored design information to return useful information for a subsequent engineering problem, and contributes to the SE state-of-the-art by presenting a method to assess the performance of search methods within a DR.

1.2.1. Model Based Systems Engineering

As SE evolved as a discipline in the latter half of the twentieth century, process and method standards were established that generated and relied upon a large number of physical documents. As part of the SE process, practitioners spent a large amount of time writing, validating, and verifying conformity within and between documents and artifacts, and the benefits of those efforts were considered self-evident [2]. In the 1990s and 2000s, the inefficiency of document-based engineering became apparent to the engineering community. For large complex systems, requirements and specifications could exceed thousands of pages, and requirement sets were often ambiguous, incomplete or even contradictory. By relying upon humans to validate design documents, SE processes were labor intensive and introduced opportunities for stakeholder misinterpretation of problem and solution system characteristics. To improve the efficiency of SE and enable engineering teams to tackle more complex engineering problems, MBSE was presented as an alternative to document-based systems engineering.

The use of MBSE is one way that the SE community can better manage complexity, maintain consistency and assure traceability during all phases of the system life cycle [1,3]. A system model provides systems engineers and other stakeholders with a shared view of a system that supports communication, coordination, analysis, and decision making, and serves as the authoritative source of system information [4]. For readers interested in expanding their understanding of MBSE, the motivation for and current state of MBSE are described in [1,5], and INCOSE's Vision 2035 [6]; early industry practices are described using survey results in [7]; and core MBSE concepts are described in [8,9].

In the forward looking INCOSE Vision 2020 publication, MBSE is defined as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [10]. At the 2007 INCOSE Symposium, the benefits of MBSE were presented as follows:

- Improved communications among the development stakeholders (e.g., the customer, program management, systems engineers, hardware and software developers, testers, and specialty engineering disciplines).
- Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes.
- Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness.
- Enhanced knowledge capture and reuse of the information by capturing information in more standardized ways and leveraging built in abstraction mechanisms inherent in model driven approaches. This in-turn can result in reduced cycle time and lower maintenance costs to modify the design.
- Improved ability to teach and learn systems engineering fundamentals by providing a clear and unambiguous representation of the concepts.

Many of the envisioned MBSE capabilities found in the INCOSE Vision 2020 remain beyond the state-of-the-art. While much progress has been made in the field of MBSE, further maturation is necessary to realize the full benefits. In the fourteen years between the release of INCOSE Vision 2020 and today, MBSE has grown from being a new initiative to being viewed as imperative to the entire SE discipline. According to INCOSE's Vision 2035, "the future of SE is model based, enabled by a major digital transformation" [6].

At the core of MBSE is the concept of a model. In SE, a model is a representation of a system—an abstraction that suppresses certain details of the system. Models of a system can be used to communicate within an engineering team or stakeholder community and to predict properties or behaviors of modeled systems [11]. Models are a necessary tool for representing complex subjects because the human mind has a limited capacity for comprehension that is easily overwhelmed by the intricacies of complex systems [12,13]. Simplified models can be used to answer questions about the system under study in place of the actual system [14,15]. Practicality demands that a model is easier to use than the system under study, which leads to the abstraction of system detail out of the resulting model [13].

As a method of communication, models require the user community to apply consistent semantics and syntax to ensure alignment of shared understanding. The intended meaning of the basic concepts of a model are referred to as the semantics of that model. For complex concepts, or concepts that readily map to multiple semantics, a formal notation of semantics is necessary to ensure that the understood meaning of a model is consistent across stakeholders. This formal specification of a modeling approach, that explains the base concepts of a model and how they relate to each other, is called a metamodel. Bézivin and Gerbé [13] provide a succinct explanation of metamodels and how they relate to models and systems. In MBSE, systems are represented by models, which are expressed in a modeling language that aligns to a defined metamodel and domain ontology [1].

A modeling language serves as the formal basis for modeling tools and enables the development of system models [3]. A modeling language is based on a metamodel [16]. To enable effective MBSE, a modeling language must be able to express the fundamental concepts of SE. These concepts include system physical and conceptual hierarchies, connections and interfaces within the system and between the system and its environment, system functionality and resultant behavior, measurable system characteristics, and the relationships among and between system requirements, design characteristics, analysis, and verification [4].

The SE community has not reached consensus regarding a standard modeling language underpinning MBSE efforts, and a number of modeling languages are used in widely available system modeling tools. Models developed in one language may not transfer between tools and can be misunderstood by stakeholders unfamiliar with the source modeling language and extensions used by the model creators [1]. As the SE community moves towards a standard modeling language for MBSE, the chosen modeling language or languages will need to meet the requirements laid out by [4]. A number of modeling languages have been used as the formal basis of commonly used MBSE tools, including Unified Modeling Language (UML), Systems Modeling Language (SysML), Lifecycle Modeling Language (LML), System Definition Language (SDL), and Object-Process Language (OPL), among others.

Ontologies help to enable knowledge management by establishing domain concepts in the form of terminologies, definitions, and relationships. Gruber [17] defines conceptualization as "an abstract, simplified view of the world that we wish to represent for some purpose"—in line with our definition of a model—and defines an ontology as the "explicit specification of a conceptualization". An ontology is then the explicit specification of a model. Along with formally defined model semantics, ontologies support the distribution and reuse of models by explicitly defining the information in the model [18]. Roussey [19] provides an introduction and overview of ontologies and ontology engineering. Yang, Cormican, and Yu [18] present a review of previous literature in the field of ontology engineering relevant to SE.

Formal ontologies can provide a shared understanding of modeled concepts, and have been proven as a means of improving consistency of modeling standards and model interpretations [20]. By providing a consistent vocabulary that is absent of ambiguity, ontologies can support interoperability between humans as well as between humans and machines [19,21]. The interoperability enabled by a common terminology is central to engineering knowledge reuse [22].

The value that ontologies provide, as compiled by [18], extends beyond the SE discipline. Ontologies provide a schema for knowledge graphs, which use a graph-based data model to store large sets of data extracted and integrated from diverse sources of data [23]. The state-of-the-art for knowledge graph concepts and processes are are described in [24–27]. Smajevic and Bork [28] developed and demonstrated a platform to transform models created using modern metamodeling platforms into knowledge graphs, and utilized a set of 5000 UML models to detect model characteristics that may indicate weaknesses in design known as "code smells". Their demonstration of the use of knowledge graphs to analyze UML models is relevant to SE because the most widely used modeling language in SE, SysML, is an extension of UML.

The Web Ontology Language (OWL) is a knowledge representation language published and maintained by the World Wide Web Consortium (W3C) [29]. OWL was developed to support machine processing of documented knowledge by allowing for the explicit representation of terms and the relationships between terms in the form of an ontology. The development and specification of OWL was heavily influenced by the DARPA Agent Markup Language, DAML [30]. W3C maintains OWL specifications within the domain of the "semantic web", an effort with the goal of making internet data machine-readable. OWL allows for individual users to define and share their own ontologies. The meta-metamodel for OWL is the Resource Description Framework (RDF). RDF information is structured as statements about resources in the form of triples expressed as subject-predicate-object, resulting in a "graph based data format which is schema-less and self-describing" [31].

The concept of knowledge graphs is an area of active research within SE. Blackburn and Denno [32] demonstrate the use of semantic web technologies (OWL and RDF) to develop interoperability between domain specific modeling tools and analysis tools [32,33]. Nassar and Austin [34] explored the use of RDF to structure requirements and component properties as a graph and demonstrate component selection and tradespace analysis for a home theater design problem using sequence inference analyses on the resulting graph. Jenkins and Rouquette [35] demonstrate a technique for representing UML and SysML in OWL, creating axioms within the resulting ontologies, and testing these axioms using available OWL tools—a form of model validation. Shani [36] investigates reuse in MBSE, and finds that MBSE projects create design representations that can be managed and queried through the use of RDF and OWL for semantic mediation. Recent research into knowledge graph applications within industry [37–39], recommendation systems [40], digital resource discovery [41], and the population of knowledge graphs with information from patent systems [42] are relevant to SE.

1.2.2. Model Based Systems Engineering Tools

As the state-of-the-practice of SE is transitioning from a document-based paradigm to MBSE, many software solutions have become available in both the commercial space and as free and open source software (FOSS). The capabilities, underlying data structure, and interoperability of these programs varies across a wide spectrum. Development of DR that can support and use data from multiple MBSE tools is currently in progress by the authors.

A review of recent SE literature did not return an authoritative listing of the available MBSE tools. Such a list would become obsolete without constant revision due to ongoing development activities and the introduction of new tools. The information in this section is not intended to offer a complete list of available MBSE tools at the time of authoring, and is instead intended to provide the reader with the understanding that a large number of MBSE tools are available to the SE community. A list of MBSE tools was compiled based the work of [7,43], the Systems Engineering Tools Database managed by INCOSE and Project Performance International [44], and the software marketplace and review website G2 [45]. Tools which appeared in two or more of the source tool listings were further reviewed, and pertinent information regarding the modeling language, ontology, and and meta-metamodel of each tool is shown in Table 1.

A set of system models can be expressed in the form of a knowledge graph. A DR in the form of a knowledge graph may store multiple system models as instances of one or more ontologies expressed in one or more modeling languages and incorporating knowledge from one or more domains. The myriad MBSE tools available for the SE community leads to a knowledge graph composed of diverse ontologies and modeling languages. The lack of underlying consistency in system model ontologies and modeling languages is an obstacle to efficient aggregation, management, and utilization of knowledge created in a heterogeneous set of MBSE tools.

Table 1. Subset of available MBSE tools. Adapted from [43–45].

Product Name	Vendor	Modeling Language	Defined Ontology (Metamodel)	Meta-metamodel Structure
Astah	Change Vision, Inc.	SysML		MOF2
		UML2		
Cameo Systems Modeler	3DS	SysML		MOF2
		UML2		
Capella	Eclipse Foundation	DSL		Ecore
CORE	Vitech	SDL	YES	ERA
Cradle	3SL	SysML		MOF2
		UML2		
Enterprise Architect	Sparx Systems	SysML		MOF2
		UML2		
		BPMN		
GENESYS	Vitech	SDL	YES	ERA
Innoslate	SPEC Innovations	LML	YES	ERA
Modelio	Modeliosoft	SysML		MOF2
		UML2		
		BPMN		
Papyrus	Eclipse Foundation	SysML		Ecore
		UML2		
		BPMN		
Rhapsody Architect for SE	IBM	SysML		Ecore
		UML2		

1.3. Design Reuse

Knowledge reuse, in the context of SE, refers to the reuse of design knowledge captured in previous efforts in support of a system design solution for another engineering problem [46,47], and has the potential to improve the quality, cost, and schedule of engineering projects. KR is the way that engineers avoid the colloquialism of 'reinventing the wheel.' Reuse of knowledge and experience reduces the effort associated with design by allowing engineers to avoid the mental and physical costs associated with initial invention of constituent pieces of their system of interest. Reuse can range from the reuse of simple tools and processes to the reuse of significant pieces of software or even entire systems within a system of systems.

All modern engineering efforts involve some degree of KR. The degree to which existing knowledge can be reused is determined by the applicability of information to the problem of interest, the availability of that knowledge to the engineering team, the ability of engineers to find the applicable knowledge, and their ability to make sense of it in the context of their own work. KR can be applied across the system design process, including decomposition of requirements, alignment of requirements to functions, mapping of functions to components, and identification of potentially unrecognized relationships among system components and attributes. Through KR, the quality of system design can be improved by increasing the quantity of potential design solutions in design tradespaces. The use of demonstrated design solutions and partial solutions can reduce design costs and timelines provided the proper tool sets are in place to efficiently search, understand, and repurpose data from a robust source of design knowledge.

One challenge with reusing SE knowledge is that much of this knowledge is stored in engineers' minds [48]. Demian [49] performed a series of interviews of professionals in the architecture, engineering, and construction industries to support a study of knowledge reuse. The survey results found that a substantial portion of reuse occurs through social knowledge networks, e.g., recollection of previous projects and mentor-mentee relationships. Internal knowledge reuse (i.e., reuse of knowledge from an engineers previous experiences) is the most prevalent form of reuse due to the ease with which engineers can recall both items to reuse and the context of those items in their initial design environment [48]. The importance of context in the reuse of design knowledge is present throughout previous works. To transform this knowledge into a form that can be applied to appropriate problems in the future, knowledge must be made explicit and properly documented [50].

As system complexity grows, the benefits of capturing, managing, and leveraging information from previous experience grows. Accumulated information within SE can be grouped into Knowledge and Know-How (K&KH). Knowledge represents information that can provide a solution to problems and questions, while Know-How provides both solutions and a manner in which to develop solutions [51]. Within SE, K&KH align roughly to knowledge about a domain or system of interest and knowledge of systems engineering activities, respectively, [52].

Regardless of the documentation method used to capture SE K&KH, the viability of different reuse methods is affected by the source of information. Wu et al. [52] identify three types of SE K&KH reuse, differentiated by attributes of the source of the information: opportunistic reuse, when the source project for the information was not developed with consideration for reusability; planned reuse, when the source project for the information was developed with reuse in mind; and variance, when multiple projects share a core knowledge base or model. Design repositories can support all three forms of reuse, but primarily improves opportunistic reuse.

Cloutier and Verma [50] describe six methods of K&KH reuse, covering a range of problem specificities. Heuristics offer general guidance across a range of applications [53]. Based on previous experience, heuristics provide guidance for situations with reasonable but not absolute certainty of outcome when followed, and are sometimes called “rules of thumb”. A trade-off of the ambiguity of heuristics is that they are fallible, sometimes provided incorrect guidance when used in practice [54]. In addition to heuristics, templates offer a basic starting point for a particular application, providing format but no guidance regarding other design decisions. Templates have more specific applications than heuristics, but do not provide significant direction aside from format. Processes offer a third a form of reuse, defined as a “set of interacting activities, which transform inputs into outputs” [50]. Processes are better specified than templates, as they describe part of the applicable context (inputs and outputs), as well as a high-level structure. Fourth, frameworks offer a reusable, “logical, organizing structure used to classify information, concepts, data, etc.” [50]. Frameworks are an example of formalized and documented engineering Know-How. Fifth, the authors discuss patterns in engineering, which is expanded below. A collection of different patterns grouped by similarity of application, called a pattern language, is the last method described.

Christopher Alexander [55] introduced a “pattern language” as a set of patterns that helped to express design in terms of relationships between the parts of a building, and the rules for applying those relationships in other contexts. Numerous works have asserted the value of patterns in engineering and systems engineering [50,52,56,57]. Patterns, defined in the context of System Architecture, are “a high-level structure, appropriate for the major components of a system. [Patterns] express the relationship between the context, a problem, and a solution, documenting attributes and usage guidance. Patterns are time-proven in solving problems similar in nature to the problem under consideration” [50]. Patterns support reuse by specifying a problem-solution pair with enough rigor to allow the same solution to be applied with a degree of confidence to an identified set of applicable problems. Patterns, while more specific in their application, are still a form of heuristic [58], and their successful application is not guaranteed, and still requires good engineering judgement.

Wu et al. provide a summary of previous research into the application of patterns for reuse in a MBSE environment [52]. The INCOSE MBSE Patterns working group, co-chaired by Bill Schindel and Troy Peterson, has introduced and socialized Pattern Based Systems Engineering, which seeks to improve working speed, expertise leverage, and systems knowledge within the SE by orders of magnitude through the reuse of general system patterns [59]. Their work presents a methodology for development teams to take generalized system-level models and customize and tailor these model patterns to meet the needs of their specific system of interest. The working group envisions that this effort will transition applicable MBSE projects from "learning to model" to "learning the model"–reducing the technical barrier of new modeling tools and passing expertise between teams and projects as patterns are improved. Kristin Giammarco presented an method for identifying reoccuring architecture design aspects and documenting the identified aspects as abstract patterns that can then be used as design rules by which design maturity can be assessed [60]. The Object Management Group's (OMG) Reusable Asset Specification (RAS) provides a "set of guidelines and recommendations about the structure, content, and descriptions of reusable software assets" [61]. The RAS addresses the elements of reuse that apply to engineering and endeavors to support reuse by providing a standard packaging for design knowledge storage and transfer. RAS meets the criteria outlined by Alexandrian Patterns by including a problem and solution pair, as well as the context for application of the solution to a similar problem, guidance for usage, and documented attributes, which are stored as artifacts. Phister et al. have proposed a method to formalize the capture of SE knowledge relying on a systems conceptual foundations and a SE pattern catalogue [62].

The value of design reuse is evident [63] not only for SE in general, but for specialized fields, such as MBSE, as they progress in maturity. To aid in that progression, engineers have developed multiple supporting tools and techniques. One tool that is particularly relevant to SE design knowledge reuse is DR.

1.4. *Design Repositories*

DR offer a centralized store of design knowledge in a form chosen to enable reuse, either by humans or computers [64]. DR can be populated with design knowledge and artifacts from completed or ongoing engineering efforts. DR are knowledge-based design artifact modeling and storage systems that are used to aid in the collection, representation, distribution, and reuse of design knowledge [64]. As with design libraries, DR increase the amount of design knowledge readily available to individual engineers and designers or design teams.

1.4.1. Design Repository Precursors

Early DR research extended the concepts of a design catalogue and a design prototype. Design catalogues, introduced by Roth [65], support engineers in the design process by providing a collection of solutions to fundamental design problems [66]. A design prototype "provides [the basis for the start and continuation of a design] by bringing all the requisite knowledge appropriate to the design situation together in one schema" [67]. Research related to design catalogues published since the 1990s has focused on the use of early design catalogue concepts to structure knowledge accumulated during research into other topics or the use of existing catalogues, with no research into the advancement of design catalogue generation methods [68].

The Defense Advanced Research Projects Agency (DARPA) funded a series of research projects in the late 1990s that set the academic foundation for DR within SE. These efforts evolved into Active Catalog [69–71], the National Institute of Standards and Technology Design Repository Project (NIST-DRP) [72], and the Collaborative Device Modeling Environment (CDME) [73].

The desire to reduce engineering time spent on communication of design knowledge through improved design knowledge reuse led to the development of design libraries to support the selection and design of components [72]. The NIST-DRP perceived the value of DR as providing a clear data source for information presented to human designers. Initially, the objective for the NIST-DRP was a set of case studies with formal presentations of system form and function , from which engineers could abstract information for application to other design efforts. However, the primary contribution of the initial NIST-DRP effort was the creation and demonstration of a data language for modeling design knowledge. The effort chose to sacrifice language features that enabled efficient and effective computational processing in order to provide rapid and easy human comprehension. Later NIST-DRP efforts recognized the importance of effective computational processing in support of computer based design generation [64].

Concurrently with the NIST-DRP effort, [69] developed and presented a prototype design library called 'Active Catalog'. Active Catalog is a direct progenitor of SE DR, developed to support the engineering of components by improving the capacity of designers to reuse detailed design knowledge. The prototype system stored physical attributes of components, as well as function and behavior data for each part. By combining component data with structured domain knowledge, the prototype allows for more effective search than traditional queries or keyword matching. The intent of the Active Catalog prototype work was to support a "try-before-you-buy" methodology for engineering design .

Design libraries, as demonstrated by Active Catalog, offered an improvement over alternative data sources, namely general internet queries, dedicated parts referral services (which connected engineers to parts companies based on keywords, but did not store component level information), and centralized parts catalogs, which worked as a central clearing house and provided part level data from suppliers that were enrolled [69]. Each of these alternatives offered limited data for identified components, and search coverage was limited by keyword association and published part data.

The Active Catalog research team identified several high-level requirements for the Active Catalog prototype's approach to knowledge management. For the prototype to meets its intended purpose, it needed to meet the following design requirements [70,71]:

1. Active and Dynamic Data. Information in the Active Catalog supports usability by humans and computers. In addition to being used for human understanding, the information is to be used as inputs and building blocks during application development. The Active Catalog program supports dynamic data through the processing of data in user environments to meet needs of a specific user.
2. Composable and Integrated Data. Active Catalog was built as a standalone application but assumes that data will be shared and used cooperatively. The envisioned system architecture would hold information on a set of trusted-sites, which would provide reusable modules and ontologies to support integration.
3. Multi-dimensional Data, Sharable in Multiple Domains. Active Catalog information represents physical parts in addition to software components. The objects represented in the Active Catalog have multiple views and interact with other objects across multiple domains, which is represented in the Active Catalog.
4. Syntactically and Semantically Accessible Data. Active Catalog makes use of both traditional search techniques, leveraging keyword search and statistical correlation techniques, and semantic search capability leveraging ontologies and mappings between components and functions.
5. Interoperable Information, Executable in Distributed Environments. Active Catalog information was intended to be used in a distributed fashion, and the system architecture needs to account for the transmission and use of data in multiple pieces of software.

Active Catalog included a set of simulations that could be populated with inputs from selected components and ran to generate predicted performance of components acting as a subsystem. Simulations were feasible due to the component set included in the library belonging to a small set of part types, with understood physical properties and interactions. The prototype DR did not provide any generalized methods for the construction of ontologies or flexible simulations. At the time of publication, the Active Catalog included 300 classes of pumps, 3000 pump attributes, 72 classes of motors, and 66 attributes for motors [74], while these required attributes were generated to support a DR intended mostly to be used for human component selection in support of design, they were suitable to serve as a DR for computer-based design synthesis for more complex pump systems.

Active Catalog allowed for search based on product class and product attribute ontologies organized in a graphical hierarchy (a hierarchy in which each node can have multiple parents), which allows users to narrow search criteria as they browse. Product, application, and functional relationships and specific component attributes could be leveraged as search criteria. Combinations of any of the above search criteria could be used to narrow the search. Flexibility in search criteria increases the effectiveness of search within Active Catalog by allowing the user multiple ways to screen out unwanted results. The structure of information in the Active Catalog into ontologies increased search coverage by leveraging domain knowledge to bridge differences between the level of abstraction present in search criteria and the design data stored in the repository [70].

Ling et al. [71] provide a demonstration of the use of Active Catalog in support of the electro-mechanical design of pump and rotor systems. Active Catalog included simple taxonomies for types of pumps and their attributes, and a semantic network for pump knowledge, shown in Figure 1. The authors demonstrate that these connected layers of knowledge allow the system to deduce materials from function, such as if a pump will be used to transfer corrosive fluid, then the pump needs to be corrosive resistant. This lets users describe the application of a component they are searching for without having to know all of the technical terms or implied requirements associated with meeting those specifications.

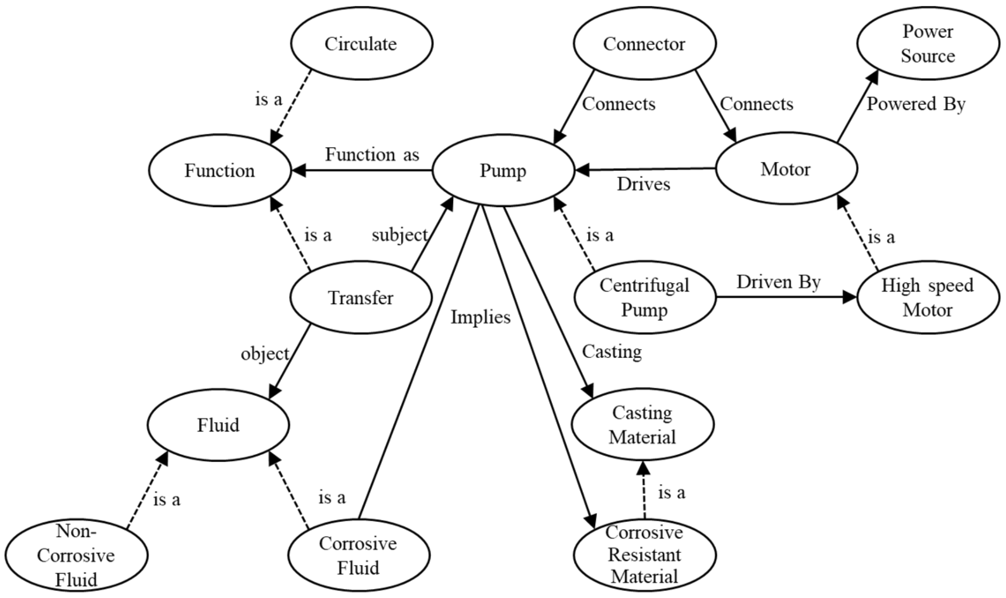

Figure 1. Pump semantic network. Adapted from [71].

In a submission to the 1998 American Society of Mechanical Engineers (ASME) Design Engineering Technical Conference, a group of researchers described the capabilities provided by Active Catalog as follows [74]:

- creating queries for parts based on their intended use rather than merely parametric specifications;
- refining those queries to take account of constraints imposed by other components;
- providing multi-modal information to help designers assess and compare candidate parts; and
- generating simulation models of candidate parts and integrating them to provide simulation models of candidate systems.

Active catalog provided a theoretical and demonstration foundation for more comprehensive efforts in the development of tools for reuse of design information systems engineering, including DRs.

Research performed in support of the NIST-DRP, led by Simon Szykman, distinguishes DR from traditional design databases across several characteristics:

- Traditional design databases contain only a limited representation of an artifact (such as drawings or CAD models), while design repositories attempt to capture a more complete design representation, and may include function, behavior, design rules, simulations, and models.
- Traditional design databases contain a limited set of data types and tend to be static sources. Design repositories may include formal data and information models, mathematical simulation models, and more.
- Design repositories support retrieval and reuse of design knowledge, the representation of physical and functional hierarchies, behavior and performance simulations, and some degree of design reasoning automation.

The primary benefits of a design repository relate to supporting distributed design teams, increasing the amount of available design information, and providing engineers with the ability to retrieve and reuse design knowledge [64].

Further research in DR built upon the initial efforts of Active Catalog, NIST-DRP, and CDME. Szykman [64] described interfaces that were developed to support the creation and navigation of design knowledge models in DR and their role in DR software architectures. Bohm and Stone [75] identified shortcomings in computer generated designs from DR due to physical relationships between components not being captured in repositories if two components were not functionally related. Their work described and implemented the addition of "supporting functions" to DR, which capture the physical functions of components, such as how a screw couples two physical components together. Bohm and Stone hosted their repository system on the internet, supporting design knowledge exchange between researchers in academia and industry [76]. Bohm et al. [77] present the necessary data schema required to capture the fundamental design knowledge necessary for a robust DR, expanding upon earlier works.

Bohm, Vucovich, and Stone [78] demonstrate the application of function-component analysis leveraging morphological matrix techniques and chi-matrix techniques to generate design concepts from a DR. Wilson et al. [79] built a DR for information from the biological and engineering domains and leveraged information modeling to allow retrieval of potential opportunities for biomimicry based on search terms from the engineering domain. Their effort demonstrated cross-domain applications of DR, and applications of logic across a functional hierarchy. Bohm et al. [80] used DR to generate concept designs in support of assessing environmental impacts of potential design solutions during the conceptual design phase. Ferrero et al. [81] generated a sustainability DR to support life cycle analysis of a set of 47 consumer products, ranging in complexity from a soda can to a 3D printer. They leveraged the DR as data source for multiple assessments of environmental impact.

The utility of DR, particularly for computer based or computer enabled design, requires consistency. Standard modeling languages used in the field of Systems Engineering—including UML, SysML, LML, SDL, and OPL—are used to improve consistency across models. The use of system design information captured during MBSE efforts to populate a DR with a set of system models using a heterogeneous set of modeling languages is currently being researched by the authors as a tool to support the reuse of MBSE design information generated by modern SE practices.

1.4.2. Design Repository Takeaways

DR have been introduced, extended beyond their initial instantiations, and use cases have been demonstrated in existing research. To date, the area of DR functional assessment remains mostly unresearched. Assessing the ability of DR to meet their intended purposes is necessary to evaluate available and emerging DR technologies and identify areas for functional improvement. Consistent with the goals of DR as defined by Szykman [64] and Bohm and Stone [75], DR should be assessed on their ability to perform the following tasks:

1. Capture
2. Store
3. Search / Retrieve
4. Present
5. Distribute
6. Export
7. Synthesize

This paper provides a novel approach to assessing DR search functionality. The method may be applied to compare competing search algorithms within a single DR, compare search method effectiveness across multiple DR, or compare effectiveness of multiple search methods for subsets of designs within a single DR.

2. Materials and Methods

As shown in the previous section, reuse of knowledge compiled and generated during the design phase of systems engineering efforts requires a search and retrieval method. For subsets of the knowledge stored in a DR to be presented, distributed, or exported, users need a way to identify what information in the DR is relevant to their specific problem. These search functions place constraints on the information in the DR in order to remove extraneous information and reduce the returned information to a manageable subset. Search functions are designed to return results that best fit user defined input criteria. Within a SE DR, these inputs may be system characteristics from a system of interest, with the goal of finding similar existing systems, or data from an unknown system, with the goal of identifying a system that meets the provided criteria.

The Active Catalog prototype efforts included work to maximize the effectiveness of search functions within the application [70]. The measures used to assess search effectiveness were hit ratio, the proportion of items returned in a search which are useful, and coverage, the proportion of useful items in the total search domain that were returned in the search result. The authors found that an ideal search method, whether it is supporting a human user or computer application, maximizes both hit ratio and coverage while consuming as little processing power as possible .

For the purposes of this article, system design information will be treated as a directionless graph, with system entities represented by nodes (or vertices), connected by edges, representing relationships of any type, with attributes associated with both nodes and edges. Applications of graph theory to systems engineering are well described in Harrison [82]. As an example, LML defines entities as “something that can exist by itself and is uniquely identifiable,” and includes the following 12 entity types: Action, Artifact, Asset, Characteristic, Connection, Cost, Decision, Input/Output, Location, Risk, Statement, and Time, as well as multiple children entities. In addition to entities, LML includes relationships and attributes. LML modifies the standard entity, relationship, attribute approach slightly by including attributes on relationships, providing the adverb (relationship attribute), as well as the noun (entity), relationship (verb), and attribute (adjective) language elements needed to describe a system in detail [83].

A system design, as stored in the DR, is defined by the set of nodes, edges, and attributes associated with any of the constituent nodes and edges. Through this paper, the notation for System S, made of the set of nodes (N) and edges (E) will be denoted as $S = (N_S, E_S)$. Search criteria, defined as a non-empty set of nodes, will be denoted as (N_C). The following sections describe the context of DR search methods, to include potential challenges that must be overcome, proposed metrics for measuring search methodology performance, and a novel approach for assessing search performance.

2.1. Criteria Robustness

Within a DR, search methods need to identify systems that meet search criteria regardless of the portion of the returned system that is specified in the search criteria. Though it is a trivial matter to identify a system in a DR when presented the entire system specification, where $(N_C) = (N_S)$, search criteria cannot be expected to fully match a system in the repository. Criteria robustness defines the proportion of system information met as search criteria, defined as $R(N_C, N_S) = (N_C \cap N_S)/N_S$. High criteria robustness requirements limit the utility of search methodologies.

2.2. Criteria Accuracy

Not all applications of search methodologies require an exact and complete match between search criteria and solution systems. The ability to provide partial solutions when complete solutions are not readily available may offer utility to users of DR. To account for the possibility of imperfect matches, search algorithms must be capable of providing results when the provided search criteria are not perfectly reflective of designs within the repository. Examples of sources for incorrect data include completely incorrect criteria (intentional or accidental, such as a false recollection of a system detail), instances where search criteria are related but not the same as information within the DR (e.g., two different parts within the same part family), or due to different specification of search criteria (e.g., “electric motor” vs. “electric motor unit”). Domain specific ontologies can be used to supplement design details stored in a repository to improve coverage by providing information regarding the relationships between similar concepts [71].

2.3. Assessment Measures

In order to assess and compare search methodologies, multiple metrics are useful depending upon the application. For the purposes of this paper, success is treated as binary, with the score of a search methodology defined as the likelihood of success over a large set of trials with varied criteria robustness and accuracy. A search success is defined as when the search algorithm returns the expected system design, which is known a priori, for a given criteria set. Search effectiveness will vary by search method, DR structure, and the constituent systems that are included in the DR. The bounds of search robustness and search criteria accuracy used for assessment can be tailored to reflect the expected use environment for the DR.

$$Score_{\text{Method}} = \frac{\sum_a^b \sum_c^d \sum_1^n T_S}{\left(\frac{b-a}{k} + 1\right)(d - c + 1)(n)}$$

where:

a, b are the bounds of the number of search criteria
c, d are the bounds of search criteria accuracy
k is the size of search criteria accuracy levels
T_S is the search iteration success (binary 1 or 0)
n is the number of search iterations per robustness and accuracy level

2.4. Search Method Assessment Procedure

The proposed process for generating the search method score is iterative. For each search iteration, a target system from the DR is identified. Search criteria are selected based on the search robustness and search criteria accuracy parameters. Search criteria are generated for each search iteration by selecting system characteristics associated with the target system (i.e., a subset of connected nodes comprising a system) at random, with quantity equal to the robustness level. When criteria accuracy is less than 100%, a random number between 0 and 1 is compared to the search accuracy parameter for each criteria node. When the random number exceeds the search accuracy, a random node from the entire DR is selected, otherwise a node connected to the target system is selected. This step serves to generate inaccurate data for a portion of search criteria. Upon generating the criteria for each search iteration, the criteria are provided to the search methodology. The results of the search method are compared to the target system, with a match resulting in success (i.e., $T_S = 1$). This process is depicted in Figure 2.

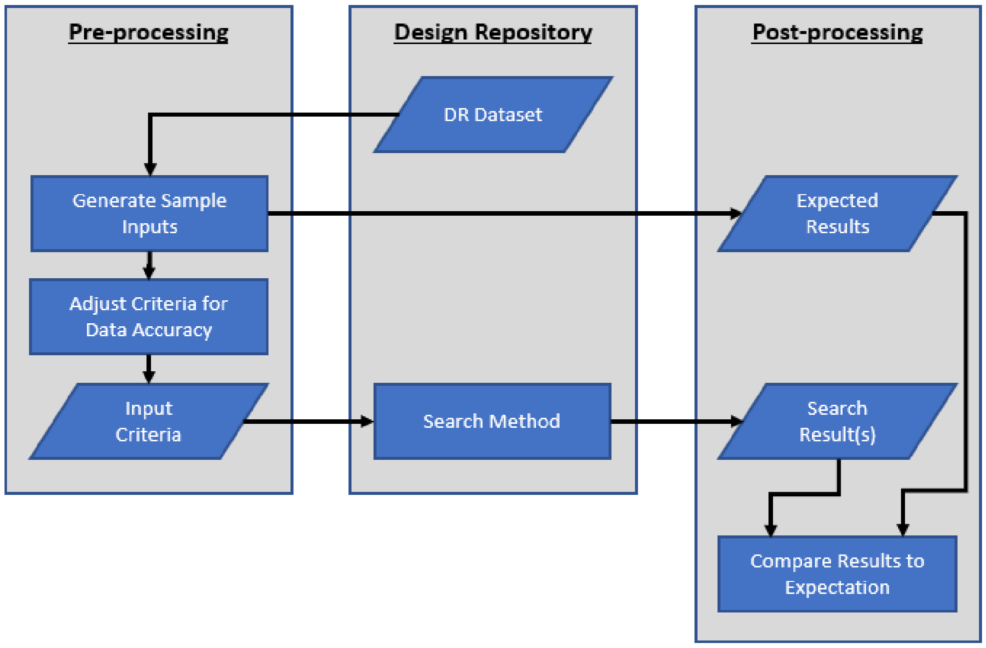

Figure 2. Search assessment method.

When feasible, this methodology should include multiple iterations over each system in the DR at each robustness and accuracy level. The random selection of search criteria introduces variability into the methodology scores returned, which can be reduced by collecting a larger sample. Upon completion of all iterations, the search method scores of all candidate methods can be compared to identify the best performing method for the DR in question for the range of informational accuracy and search criteria robustness assessed.

2.5. Case Study

The authors applied the search assessment methodology presented above to a graph-based design repository (GBDR) populated with data from the Oregon State University Design Repository (OSU-DR). A description of the GBDR structure, content, and existing search methodologies are described below, followed by the results of the search assessment methodology.

2.6. Case Study Design Repository

A locally hosted GBDR was constructed using a Neo4j graph database. The GBDR was populated with data from the OSU-DR, which was extracted from the public OSU-DR using Python web-scraping techniques. The OSU-DR project, originally funded by the National Science Foundation, involved researchers from the University of Missouri–Rolla, The University of Texas at Austin, and the National Institute of Standards and Technology. At the time data was extracted for the purposes of this paper, the OSU-DR contained product design knowledge for 182 systems, stored across multiple database tables, as shown in Figure 3 [77]. Artifact and function data were imported into the GBDR. Artifacts include entire systems, sub-assemblies, and single components, as well as the relationships between these entities.

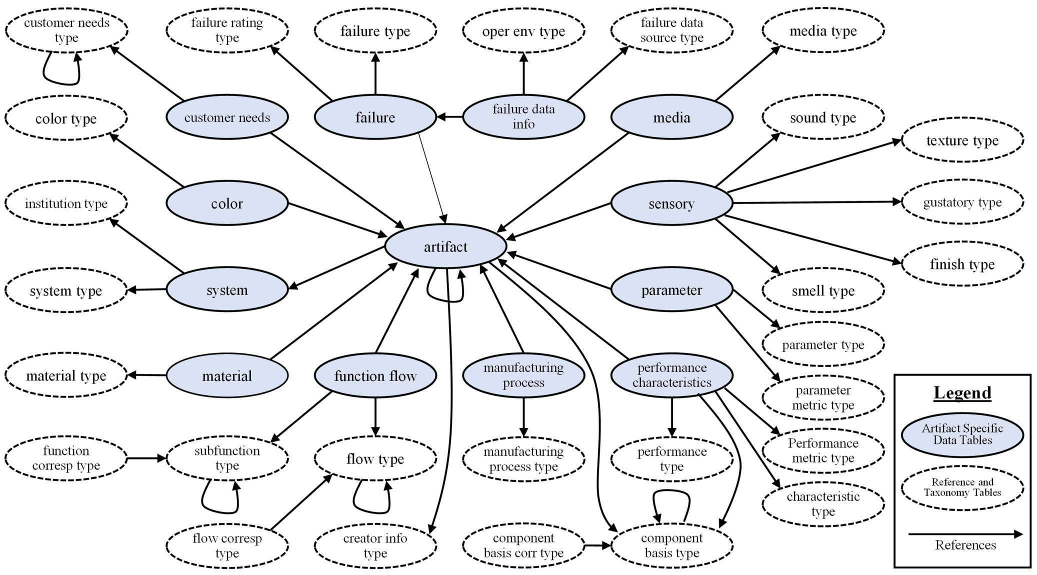

Figure 3. Graphical view of repository database tables. Adapted from [77].

The populated GBDR contains 3961 nodes, representing each system as well as the related artifacts and functions, and 16,656 edges, representing the relationships between nodes. As an example, the GBDR information for the system "digital scale" is shown in Figure 4. The top-level system node is shown in blue, with artifacts (representing the hierarchy of physical subsystems and individual parts) in brown and functions in green. The total number of nodes in the GBDR is lower than the combined total number of rows in the OSU-DR artifact and function tables because artifacts repeated across multiple systems in the OSU-DR are represented by a single node in the GBDR.

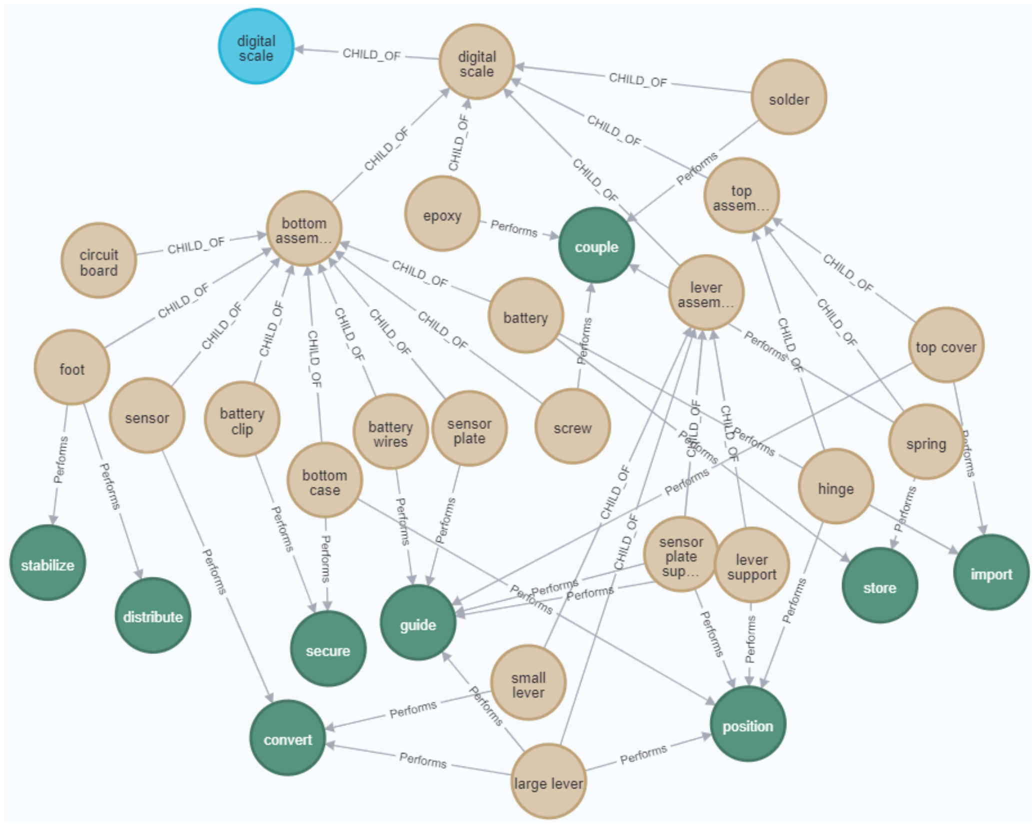

Figure 4. Neo4j visualization of 'digital scale' system information.

2.7. Assessed Search Methodologies

The goal of search methodologies in the GBDR presented above is to identify and return systems that best meet provided search criteria, defined as a system including nodes that best match the provided input criteria. Three search methodologies have been implemented in the GBDR and are described below.

2.7.1. Complete Input Criteria Coverage

The first search methodology uses simple query logic to identify the list of systems in the GBDR that meet all search criteria provided. In cases where multiple systems meet all search criteria, a single system is selected randomly and returned as the search result. When no system in the DR is found that meets all the search criteria, no result is returned.

2.7.2. Frequency Weighted Set Comparison

Among the systems in the GBDR, many artifacts and functions are found within multiple systems. Of the 5928 total artifacts, 2802, or 47%, appear in multiple systems. There are 3703 distinct artifacts, with each distinct artifact appearing in an average of 1.6 systems. Functional reuse is much more common within the GBDR, with a total of 1909 appearances of 47 distinct functions, making the average number of system occurrences for each function 40.7 systems.

Artifact and function overlap between systems causes nodes that are shared by multiple systems to appear more frequently when system information is randomly selected. This has the effect of reducing the predictive value of an artifact that appears in a significant number of systems, as it is associated with a larger portion of the DR population. An example from the OSU-DR is the artifact "screw", which appears in 20 systems, making the "screw" artifact less of a distinguishing characteristic than the artifact "scanner belt tensioner" that is uniquely aligned with a single system.

Systems with shared nodes have a higher likelihood of meeting randomly selected search criteria than if the systems constituent nodes did not appear in other systems. To account for this fact, as well as the effect of node frequency on appearance likelihood, a search method was developed that provided additional weight to less frequently appearing nodes and increased the selection standard for systems with many nodes or nodes that occur across multiple systems in the DR. The search methodology relies on calculating node frequency and system scores prior to running the search algorithm, which may not be feasible in large or continuously changing DR. This search methodology is described below.

<u>Prior to Search Procedure</u>

Steps 1 and 2 occur before the search methodology can be conducted and must be updated when changes to the DR are made.

1. Each node is assigned a score equal to the inverse of the node frequency across all systems in the DR. $node\_score_i = \frac{1}{count(i)}$
2. Each system is assigned a score equal to the sum of the node scores for all nodes associated with that system. $system\_score_j = \sum node\_score_{ji}$

<u>Search Methodology</u>

3. The search algorithm generates a fit score for each system in the DR equal to the sum of the node score for the intersect of nodes provided as search parameters and the nodes associated with the system. $ReturnMax\left(\frac{\sum node\_score_i}{system\_score_j}\right)|N_S \cup N_C$

2.7.3. Jaccard Similarity

A third search method, based on the work of Paul Jaccard [84], calculates the Jaccard similarity index for the search criteria and each system in the repository, defined as the ratio of the quantity of nodes included in their intersection to the quantity of nodes included in their union. The system with the highest Jaccard similarity index is returned by the search methodology as the result.

$$ReturnMax\left(\frac{|N_S \cap N_C|}{|N_S \cup N_C|}\right)$$

3. Results

The methodology introduced in Section 2 was applied to the three search methods available in the example GBDR. Search criteria input bounds {a,b} were set at {1,7}, search accuracy bounds of {0,1} were used for {c,d}, with search accuracy level size of of $k = 0.05$. Iterations per robustness and accuracy level was set at 10 for each system in the GBDR, resulting in a total n = 1470 search iterations for each system model in the DR. For each of the three search algorithms, 267,540 searches were performed. The resulting search method scores are presented in Table 2.

Table 2. Case study assessment results.

Search Methodology	Search Method Score
Best Complete Match	22%
Frequency Weighted Set Comparison	52%
Jaccard Similarity	65%

Search method scores were generated for each robustness and accuracy level to allow for further investigation of performance. This detailed output was used to further investigate the suitability of each search method, and to identify changes in performance across the assessed input ranges. Figures 5–7 show these scores graphically with a series for each criteria robustness level, using data criteria accuracy and search method score along the horizontal and vertical axes, respectively.

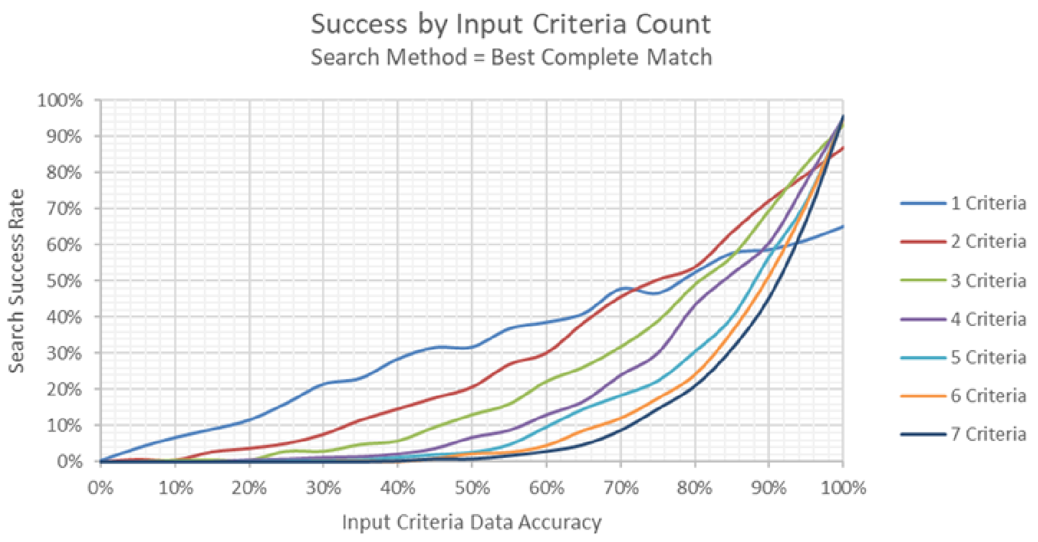

Figure 5. Results for search using best complete match.

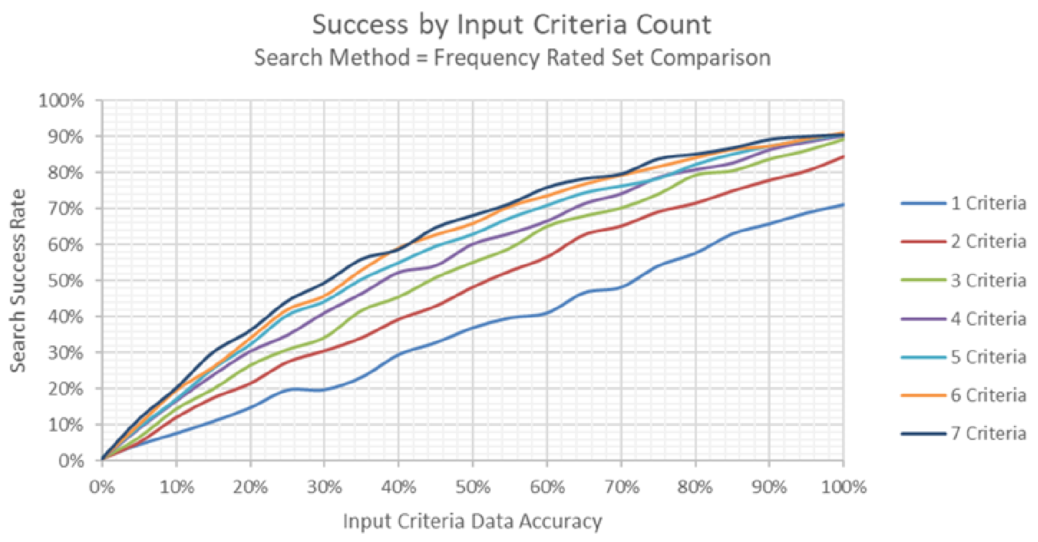

Figure 6. Results for search using frequency rated set comparison.

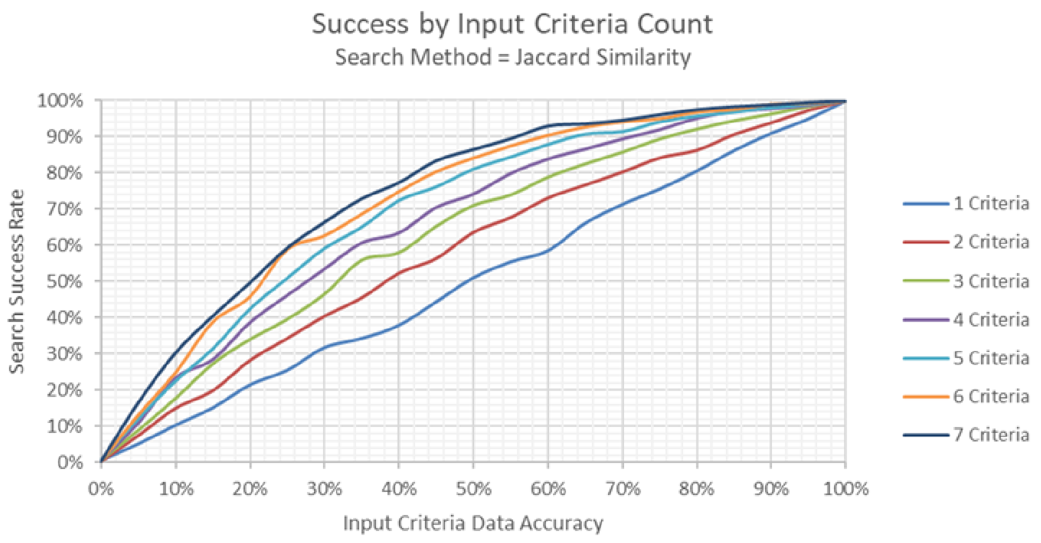

Figure 7. Results for search using Jaccard similarity.

4. Discussion

Jaccard Similarity was found to be the best performing search method across the range of inputs, followed by Frequency Weighted Set Comparison, with the Best Complete Match methodology performing the worst. These results would not necessarily hold for a DR with a different structure or a different set of data.

Counter to intuition, the Best Complete Match search method performed worse as additional search criteria are added, unless data accuracy is greater than 90%. As shown in Figure 5, when search accuracy is below 90%, every additional input criteria beyond the second reduces the search success rate further. The intended application of the GBDR in supporting early design work through the identification of potential reuse sources suggests that users are unlikely to know exactly what they are searching for, resulting in data accuracy below 100%. We concluded that the Best Complete Match search method was not ideally suited for the GBDR, regardless of the high search success rate seen when data accuracy is 100%.

The Frequency Rated Set Comparison search method performed slightly better than the Best Complete Match search method when a single search criterion was provided. As additional search criteria were included, the search success rate for this search method increased at all levels of data accuracy. Improvements are visually detectable in Figure 6, with decreasing marginal benefit seen for each added search criteria. When applied to the GBDR data set, the maximum search method score received was 91%, relative to 100% for Jaccard Similarity and 96% for Best Complete Match. Despite a lower maximum specific performance than the Best Complete Match search method, the Frequency Rated Set Comparison search method was deemed a better search method for a GBDR due to higher performance when multiple search criteria were provided and more resilient performance across a range of data accuracy levels.

Reviewing the detailed data generated for the Jaccard Similarity search method, the most readily apparent result is that success rates converge on 100% for every number of search criteria provided. Benefits from additional search criteria are seen for the range of data accuracy between 5% and 95%—any value between complete accuracy and completely random inputs—with reducing marginal benefits of additional search criteria throughout this range. The Jaccard Similarity search method was superior to both evaluated alternative search methodologies for all inputs where data accuracy was greater than 0%.

To demonstrate the superiority of the Jaccard Similarity search method, a comparison of search methods with consistent input criteria is shown in Figure 8 and summarized in Table 3. The effect of changes in data accuracy on performance of the three search methods can be seen. With five search criteria, the Jaccard Similarity search method is best across all levels of search criteria data accuracy, maintaining roughly 15% higher success rates than the Frequency Weighted Set Comparison search method where search criteria accuracy is greater than 30%. The Best Complete Match search method has worse performance than either of the other search methods when input criteria accuracy is below 100%.

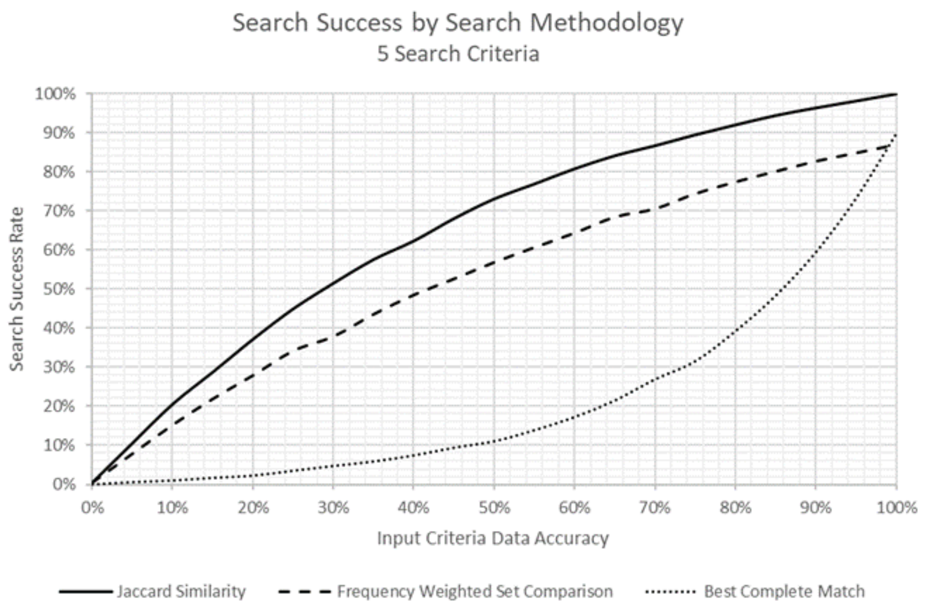

Figure 8. Results by search method and input criteria data accuracy using five input criteria.

Table 3. Search method scores for various criteria data accuracy (DA) ranges.

Search Method [1]	**$0 \leq$ DA ≤ 100**	**$50 \leq$ DA ≤ 100**	**DA = 100**
Best Complete Match	22%	39%	90%
Frequency Weighted Set Comparison	52%	73%	87%
Jaccard Similarity	65%	88%	100%

[1] For each search method and data accuracy pair, a total of n = 1820 searches were performed with five search criteria for search robustness level between 1 and 7 (inclusive).

If user search habits were known for the GBDR used as a case study, a subset of robustness and data accuracy level inputs could be used to generate the method scores used to compare performance. Since the GBDR used does not have an active user base, this was not possible at the time results were generated. Table 3 provides three potential data accuracy ranges, and the resulting method scores for the three GBDR search methodologies examined, while Jaccard Similarity is consistently the best search method, the ranking of the remaining search methodologies is not consistent across the evaluated subsets. These results show the importance of evaluating performance of search methodologies in operationally representative conditions, which may provide different results than when performance is measured for the entire set of conditions within theoretical boundaries.

5. Conclusions

As MBSE tool adoption continues to expand, the available engineering design knowledge that can feasibly be captured and reused will similarly grow. Reuse tools, such as DR, will be necessary if the SE community wishes to efficiently reuse this knowledge. This paper presents and demonstrates a methodology for assessing search algorithm performance within DR. If DR continue to be built as tools for the SE community, instantiations should be built with multiple search methodologies to support different applications. Assessment procedures, such as the one introduced in this paper, should be included in DR tools, to provide users with effectiveness metrics reflective of their use habits and DR contents.

5.1. Future Work

This article has demonstrated the alignment between systems engineering, knowledge reuse, DR, and DR assessment methodologies. There are a variety of potential research subjects with direct applicability to DR development and utilization. The areas of GBDR, design knowledge synthesis, and context-based search all offer potential new ways to reuse design knowledge. The following research areas will further increase the value of existing knowledge in solving future engineering problems.

5.1.1. Graph Based Design Repositories

DR in existing research have required a consistent ontology or informational model for the systems which are included. As MBSE tools are introduced, it is unlikely that this requirement can be met because of the myriad informational structures and schema used in different tools. GBDR offer an opportunity to store data from systems that do not have a consistent defined structure or ontology. Machine learning and big data techniques may be able to glean useful information from these irregular data sets, allowing for consumption of a larger portion of existing data sets comprised of MBSE artifacts with heterogeneous ontologies or modeling languages. The adaptation and application knowledge-based engineering and design automation techniques within MBSE would expand the state-of-the-art for both MBSE and knowledge-based engineering [85].

5.1.2. Context-Based Search

DR presented in existing literature have stored limited system design information and have not demonstrated search methods that consider system context. The importance of context in the reuse of design knowledge is present throughout previous works. The presented search methodologies, and the assessment approach introduced are only concerned with system entities – future work could expand these methods to include relationships and attributes as well. Context-sensitive search, with search criteria (Nc,Ec), representing both system entities and relationships, will be examined in future works.

5.1.3. Design Knowledge Synthesis

By using algorithms to identify and/or suggest matches between nodes in multiple system models, synthesis of information from multiple source models will become possible, which will benefit systems engineers by identifying potential system design solutions that cannot be feasibly identified using DR as currently described in the literature. Search methods that return synthetic results including data from multiple MBSE tool data sources is a promising area for further research. This example of applying knowledge-based engineering and design automation techniques from the engineering design discipline to SE problems aligns with the Digital Engineering Roadmap developed by the Systems Engineering Research Center [86].

Author Contributions: Conceptualization, D.H.; methodology, D.H.; software, D.H.; validation, D.H. and P.B.; formal analysis, D.H.; investigation, D.H.; data curation, D.H.; writing—original draft preparation, D.H.; writing—review and editing, P.B. and K.G.; visualization, D.H.; supervision, P.B. and K.G.; project administration, D.H. All authors have read and agreed to the published version of the manuscript.

Funding: Article processing fees funded by the Research and Sponsored Programs Office at the Naval Postgraduate School.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: OSU-DR is available at http://function2.mime.oregonstate.edu:8080/view/index.jsp (accessed on 15 March 2021). Python code used to scrape, clean, and import into a GBDR and the search method score generation queries, which repeatedly query a Neo4j database using Python, are available at https://github.com/dherrington/Design-Repository-Search-Assessment (accessed on 1 September 2022).

Acknowledgments: This research is partially supported by the Naval Postgraduate School. Any opinions or findings of this work are the responsibility of the authors, and do not necessarily reflect the views of the sponsors or collaborators.

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

Abbreviations

The following abbreviations are used in this manuscript:

ASME	American Society for Mechanical Engineers
CAD	Computer-aided Design
CDME	Collaborative Device Modeling Environment
DARPA	Defense Advanced Research Projects Agency
DR	Design Repository
FOSS	Free and Open Source Software
GBDR	Graph Based Design Repository
INCOSE	International Council on Systems Engineering
K&KH	Knowledge and Know-How
KR	Knowledge Reuse
LML	Lifecycle Modeling Language
MBSE	Model Based Systems Engineering
NIST-DRP	National Institute of Standards and Technology Design Repository Project
OMG	Object Management Group
OWL	Web Ontology Language
OSU-DR	Oregon State University Design Repository
RAS	Reusable Asset Specification
RDF	Resource Description Framework
SE	Systems Engineering
W3C	World Wide Web Consortium

References

1. Madni, A.M.; Sievers, M. Model-based systems engineering: Motivation, current status, and needed advances. In *Disciplinary Convergence in Systems Engineering Research*; Madni, A.M., Boehm, B., Ghanem, R.G., Erwin, D., Wheaton, M.J., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 311–325; ISBN 978-3-319-62217-0
2. Micouin, P. MBSE, What Is Wrong with SysML-First Issue. Available online: https://hal.archives-ouvertes.fr/hal-02070455/document (accessed on 9 September 2022).

3. Vaneman, W.K. Evolving Model-Based Systems Engineering Ontologies and Structures. In Proceedings of the INCOSE International Symposium, Washington, DC, USA, 7–12 July 2018; Volume 28, pp. 1027–1036.
4. Friedenthal, S.; Burkhart, R. Evolving SysML and the system modeling environment to support MBSE. *Insight* **2015**, *18*, 39–41. [CrossRef]
5. Stoewer, H. Model based Systems Engineering (MBSE) Missing Link In the digital Enterprise Strategy? In Proceedings of the INCOSE International Workshop, Los Angeles, CA, USA, 25–28 January 2014.
6. INCOSE. *Systems Engineering Vision 2035*; INCOSE: San Diego, CA, USA, 2022.
7. Estefan, J.A. Survey of model-based systems engineering (MBSE) methodologies. *Incose Mbse Focus Group* **2007**, *25*, 1–12.
8. Weilkiens, T.; Lamm, J.G.; Roth, S.; Walker, M. *Model-Based System Architecture*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
9. Wymore, A.W. *Model-Based Systems Engineering*; CRC Press: Boca Raton, FL, USA, 2018; Volume 3.
10. INCOSE. *Systems Engineering Vision 2020*; INCOSE: San Diego, CA, USA, 2007.
11. Dickerson, C.E.; Mavris, D. A brief history of models and model based systems engineering and the case for relational orientation. *IEEE Syst. J.* **2013**, *7*, 581–592. [CrossRef]
12. Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML: The Systems Modeling Language*; Morgan Kaufmann: Waltham, MA, USA, 2014.
13. Bézivin, J.; Gerbé, O. Towards a precise definition of the OMG/MDA framework. In Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001), San Diego, CA, USA, 26–29 November 2001; pp. 273–280.
14. Beery, P. *A Model-Based Systems Engineering Methodology for Employing Architecture in System Analysis: Developing Simulation Models Using Systems Modeling Language Products to Link Architecture and Analysis*; Technical Report; Naval Postgraduate School Monterey: Monterey, CA, USA, 2016.
15. Beery, P.; Paulo, E. Application of model-based systems engineering concepts to support mission engineering. *Systems* **2019**, *7*, 44. [CrossRef]
16. Vaneman, W.K.; Carlson, R. Model-Based Systems Engineering Implementation Considerations. In Proceedings of the 2019 IEEE International Systems Conference (SysCon), Orlando, FL, USA, 8–11 April 2019; pp. 1–6.
17. Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.* **1995**, *43*, 907–928. [CrossRef]
18. Yang, L.; Cormican, K.; Yu, M. Ontology-based systems engineering: A state-of-the-art review. *Comput. Ind.* **2019**, *111*, 148–171. [CrossRef]
19. Roussey, C.; Pinet, F.; Kang, M.A.; Corcho, O. An introduction to ontologies and ontology engineering. In *Ontologies in Urban Development Projects*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 9–38.
20. Pardo, C.; Pino, F.J.; García, F.; Piattini, M.; Baldassarre, M.T. An ontology for the harmonization of multiple standards and models. *Comput. Stand. Interfaces* **2012**, *34*, 48–59. [CrossRef]
21. Bittner, T.; Donnelly, M.; Winter, S. Ontology and semantic interoperability. In *Large-Scale 3D Data Integration*; CRC Press: Boca Raton, FL, USA, 2005; pp. 139–160.
22. Madni, A.M.; Lin, W.; Madni, C.C. IDEONTM: An extensible ontology for designing, integrating, and managing collaborative distributed enterprises. *Syst. Eng.* **2001**, *4*, 35–48. [CrossRef]
23. Noy, N.; Gao, Y.; Jain, A.; Narayanan, A.; Patterson, A.; Taylor, J. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. *Queue* **2019**, *17*, 48–75. [CrossRef]
24. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Philip, S.Y. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 494–514. [CrossRef]
25. Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; Melo, G.D.; Gutierrez, C.; Kirrane, S.; Gayo, J.E.L.; Navigli, R.; Neumaier, S.; et al. Knowledge graphs. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–37. [CrossRef]
26. Gutierrez, C.; Sequeda, J.F. Knowledge graphs. *Commun. ACM* **2021**, *64*, 96–104. [CrossRef]
27. Yan, J.; Wang, C.; Cheng, W.; Gao, M.; Zhou, A. A retrospective of knowledge graphs. *Front. Comput. Sci.* **2018**, *12*, 55–74. [CrossRef]
28. Smajevic, M.; Bork, D. From conceptual models to knowledge graphs: A generic model transformation platform. In Proceedings of the 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), Virtual Event, 10–15 October 2021; pp. 610–614.
29. McGuinness, D.L.; Van Harmelen, F. OWL web ontology language overview. *W3C Recomm.* **2004**, *10*, 2004.
30. Lacy, L.W. *OWL: Representing Information Using the Web Ontology Language*; Trafford Publishing: Bloomington, IN, USA, 2005.
31. Harth, A.; Decker, S. Optimized index structures for querying rdf from the web. In Proceedings of the Third Latin American Web Congress (LA-WEB'2005), Buenos Aires, Argentina, 1 October–2 November 2005.
32. Blackburn, M.; Denno, P. Virtual design and verification of cyber-physical systems: Industrial process plant design. *Procedia Comput. Sci.* **2014**, *28*, 883–890. [CrossRef]
33. Blackburn, M.R.; Denno, P.O. Using semantic web technologies for integrating domain specific modeling and analytical tools. *Procedia Comput. Sci.* **2015**, *61*, 141–146. [CrossRef]
34. Nassar, N.; Austin, M. Model-based systems engineering design and trade-off analysis with RDF graphs. *Procedia Comput. Sci.* **2013**, *16*, 216–225. [CrossRef]

35. Jenkins, J.S.; Rouquette, N.F. Semantically-Rigorous Systems Engineering Modeling Using SysML and OWL. Available online: https://trs.jpl.nasa.gov/bitstream/handle/2014/43338/12-5065_A1b.pdf?sequence=1 (accessed on 29 September 2022).
36. Shani, U. Can ontologies prevent MBSE models from becoming obsolete? In Proceedings of the 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 24–27 April 2017; pp. 1–8.
37. Yahya, M.; Breslin, J.G.; Ali, M.I. Semantic web and knowledge graphs for industry 4.0. *Appl. Sci.* **2021**, *11*, 5110. [CrossRef]
38. Li, X.; Lyu, M.; Wang, Z.; Chen, C.H.; Zheng, P. Exploiting knowledge graphs in industrial products and services: A survey of key aspects, challenges, and future perspectives. *Comput. Ind.* **2021**, *129*, 103449. [CrossRef]
39. Meixner, K.; Luder, A.; Herzog, J.; Winkler, D.; Biffl, S. Patterns for reuse in production systems engineering. *Int. J. Softw. Eng. Knowl. Eng.* **2021**, *31*, 1623–1659. [CrossRef]
40. Sun, R.; Cao, X.; Zhao, Y.; Wan, J.; Zhou, K.; Zhang, F.; Wang, Z.; Zheng, K. Multi-modal knowledge graphs for recommender systems. In Proceedings of the 29th ACM international conference on information & knowledge management, Virtual Event, 19–23 October 2020; pp. 1405–1414.
41. Macgregor, G. Resource Discovery in Heterogeneous Digital Content Environments. Ph.D. Thesis, University of Strathclyde, Glasgow, UK, 2020.
42. Sarica, S.; Luo, J.; Wood, K.L. TechNet: Technology semantic network based on patent data. *Expert Syst. Appl.* **2020**, *142*, 112995. [CrossRef]
43. Weilkiens, T. MBSE Tools-Model Based Systems Engineering. Available online: https://mbse4u.com/sysml-tools/ (accessed on 15 November 2021).
44. INCOSE. System Engineering Tools Database. Available online: https://www.systemsengineeringtools.com/ (accessed on 15 November 2021).
45. Available online: https://www.g2.com/categories/systems-engineering (accessed on 15 November 2021).
46. Shani, U.; Broodney, H. Reuse in Model-Based Systems Engineering. In Proceedings of the 2015 Annual IEEE Systems Conference (SysCon), Vancouver, BC, Canada, 13–16 April 2015. [CrossRef]
47. Browning, T.R. Building models of product development processes: An integrative approach to managing organizational knowledge. *Syst. Eng.* **2018**, *21*, 70–87. [CrossRef]
48. Mourtzis, D.; Doukas, M.; Giannoulis, C. An inference-based knowledge reuse framework for historical product and production information retrieval. *Procedia CIRP* **2016**, *41*, 472–477. [CrossRef]
49. Demian, P.; Fruchter, R. An ethnographic study of design knowledge reuse in the architecture, engineering, and construction industry. *Res. Eng. Des.* **2006**, *16*, 184–195. [CrossRef]
50. Cloutier, R.; Verma, D. Applying the concept of patterns to systems architecture. *Syst. Eng.* **2007**, *10*, 138–154. [CrossRef]
51. Gzara, L.; Rieu, D.; Tollenaere, M. Product information systems engineering: An approach for building product models by reuse of patterns. *Robot. Comput.-Integr. Manuf.* **2003**, *19*, 239–261. [CrossRef]
52. Wu, Q.; Gouyon, D.; Levrat, E.; Boudau, S. *A Review of Know-How Reuse with Patterns in Model-Based Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 219–229.
53. Rechtin, E.; Maier, M.W. *The Art of Systems Architecting*; CRC Press: Boca Raton, FL, USA, 2010.
54. Koen, B.V. The engineering method and its implications for scientific, philosophical, and universal methods. *Monist* **2009**, *92*, 357–386. [CrossRef]
55. Alexander, C. *A Pattern Language: Towns, Buildings, Construction*; Oxford University Press: Oxford, UK, 1977.
56. Cloutier, R. *Applicability of Patterns to Architecting Complex Systems*; Stevens Institute of Technology: Hoboken, NJ, USA, 2006.
57. Lea, D. Christopher Alexander: An introduction for object-oriented designers. *ACM SIGSOFT Softw. Eng. Notes* **1994**, *19*, 39–46. [CrossRef]
58. Cloutier, R. Pattern Identification and Management Toolset. In *Evolving Toolbox for Complex Project Management*; Auerbach Publications: Sebastopol, CA, USA, 2019.
59. Schindel, W.; Peterson, T. Introduction to pattern-based systems engineering (PBSE): Leveraging MBSE techniques. In Proceedings of the INCOSE International Symposium, Philadelphia, PA, USA, 24–27 June 2013; Volume 23, p. 1639.
60. Giammarco, K. A formal method for assessing architecture model and design maturity using domain-independent patterns. *Procedia Comput. Sci.* **2014**, *28*, 555–564. [CrossRef]
61. Object Management Group. Reusable Asset Specification. OMG Available Specification Version 2.2. Available online: https://www.omg.org/spec/RAS/2.2/PDF (accessed on 29 September 2022).
62. Pfister, F.; Chapurlat, V.; Huchard, M.; Nebut, C.; Wippler, J.L. A proposed meta-model for formalizing systems engineering knowledge, based on functional architectural patterns. *Syst. Eng.* **2012**, *15*, 321–332. [CrossRef]
63. Corin Stig, D.; Bergsjö, D. Engineering challenges of intrafirm technology reuse. *Syst. Eng.* **2019**, *22*, 243–254. [CrossRef]
64. Szykman, S. Architecture and implementation of a design repository system. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, QC, Canada, 29 September–2 October 2002; Volume 36215, pp. 429–443.
65. Roth, K.; Franke, H.; Simonek, R. Aufbau und Verwendung von Katalogen fur das methodische Konstruieren. *Konstruktion* **1972**, *24*, 449–458.
66. Franke, H.J.; Loffler, S.; Deimel, M. Increasing the efficiency of design catalogues by using modern data processing technologies. In Proceedings of the 8th International Design Conference, Dubrovnik, Croatia, 18–21 May 2004.

67. Gero, J.S. Design prototypes: A knowledge representation schema for design. *AI Mag.* **1990**, *11*, 26.
68. Inkermann, D. What Happened to Roth's Design Catalogues? - A Review of Usage and Future Research. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Virtual Event, 17–19 August 2021; American Society of Mechanical Engineers: New York, NY, USA, 2021; Volume 85420, p. V006T06A042.
69. Luo, P.; Will, P. Active catalog: A knowledge-rich design library facilitating information consumption. In *Proceedings of the Workshop on Knowledge Intensive CAD*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 157–172.
70. Kim, J.; Ling, S.R.; Will, P. Ontology engineering for active catalog. In Proceedings of the 1997 AAAI Workshop on Using Artificial Intelligence in Electronic Commerce, Virtual Organizations and Enterprise Knowledge Management to Reengineer the Corporation, Providence, Rhode Island, 27–28 July 1997.
71. Ling, S.R.; Kim, J.; Will, P.; Luo, P. Active catalog: Searching and using catalog information in internet-based design. In Proceedings of the ASME Design Engineering Technical Conferences, Sacramento, CA, USA, 14–17 September 1997.
72. Murdock, J.; Szykman, S.; Sriram, R. *An Information Modeling Framework to Support Design Databases and Repositories*; American Society of Mechanical Engineers: New York, NY, USA, 1997; Volume 97, pp. 14–17.
73. Iwasaki, Y.; Farquhar, A.; Fikes, R.; Rice, J. A web-based compositional modeling system for sharing of physical knowledge. In Proceedings of the IJCAI (1), Nagoya, Japan, 23–29 August 1997; pp. 494–500.
74. Coutinhi, M.; Eleish, R.; Kim, J.; Kumar, V.; Ling, S.R.; Neches, R.; Will, P. Active catalogs: Integrated support for component engineering. In Proceedings of the ASME Design Engineering Technical Conferences, Atlanta, GA, USA, 13–16 September 1998.
75. Bohm, M.R.; Stone, R.B. Product design support: Exploring a design repository system. In Proceedings of the ASME International Mechanical Engineering Congress and Exposition, Anaheim, CA, USA, 13–19 November 2004; Volume 47047, pp. 55–65.
76. Bohm, M.R.; Stone, R.B.; Szykman, S. Enhancing Virtual Product Representations for Advanced Design Repository Systems. *J. Comput. Inf. Sci. Eng.* **2005**, *5*, 360–372. [CrossRef]
77. Bohm, M.R.; Stone, R.B.; Simpson, T.W.; Steva, E.D. Introduction of a data schema to support a design repository. *Comput.-Aided Des.* **2008**, *40*, 801–811. [CrossRef]
78. Bohm, M.R.; Vucovich, J.P.; Stone, R.B. Capturing creativity: Using a design repository to drive concept innovation. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Long Beach, CA, USA, 24–28 September 2005; Volume 47403, pp. 331–342.
79. Wilson, J.; Chang, P.; Yim, S.; Rosen, D.W. Developing a bio-inspired design repository using ontologies. In Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, San Diego, CA, USA, 30 August–2 September 2009; Volume 49057, pp. 799–808.
80. Bohm, M.R.; Haapala, K.R.; Poppa, K.; Stone, R.B.; Tumer, I.Y. Integrating life cycle assessment into the conceptual phase of design using a design repository. *J. Mech. Des.* **2010**, *132*, 091005. [CrossRef]
81. Ferrero, V.; Wisthoff, A.; Huynh, T.; Ross, D.; DuPont, B. A Sustainable Design Repository for Influencing the Eco-Design of New Consumer Products. Available online: https://engrxiv.org/preprint/view/94/ (accessed on 29 September 2022).
82. Harrison, W.K. The role of graph theory in system of systems engineering. *IEEE Access* **2016**, *4*, 1716–1742. [CrossRef]
83. Lifecycle Modeling Language Steering Committee. Lifecycle Modeling Language (LML) Specification. Available online: https://www.lifecyclemodeling.org/specification (accessed on 29 September 2022).
84. Jaccard, P. The distribution of the flora in the alpine zone. 1. *New Phytol.* **1912**, *11*, 37–50. [CrossRef]
85. Plappert, S.; Gembarski, P.C.; Lachmayer, R. The use of knowledge-based engineering systems and artificial intelligence in product development: A snapshot. In Proceedings of the International Conference on Information Systems Architecture and Technology, Wrocław, Poland, 15–17 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 62–73.
86. McDermott, T.; DeLaurentis, D.; Beling, P.; Blackburn, M.; Bone, M. AI4SE and SE4AI: A research roadmap. *Insight* **2020**, *23*, 8–14. [CrossRef]