COMPUTER-AIDED DESIGN & APPLICATIONS, 2017
VOL. 14, NO. 1, 17–27
http://dx.doi.org/10.1080/16864360.2016.1199752

Computer-Aided Design
Taylor & Francis
Taylor & Francis Group

Utilizing design intent information to aid in the synthesis of multi-domain systems

Matt R. Bohm[1], Robert L. Nagel[2] and Marie K. Riggs[1]

[1]University of Louisville, USA; [2]James Madison University, USA

ABSTRACT
The IDEAS (Iterative DEsign of Alternative Solutions) application is presented as a prototype engineering tool that enables design space exploration during the transition from problem clarification to concept generation based on initial form and fit design conceptualizations. The goals met through development of the application prototype are to demonstrate the feasibility of form and fit based design space exploration for engineered systems and to demonstrate the feasibility of abstracting functional design intent from a form and fit based engineering design paradigm. Meeting these goals is the first step toward developing a computational system that allows functional design intent to be captured and archived during conceptual design for use later in an existing suite of computational function-based design tools.

KEYWORDS
Functional design intent; form and fit; concept generation; design process

1. Introduction

In engineering design, customer needs collected early in the design process leads to a variety of design requirements. These design requirements contain objectives, constraints, functions, and specifications that become the "intent" of engineered products, processes, and systems. Throughout the design process, design intent can be further described by several different physical attributes, non-physical attributes, and the decision rationale leading to the creation of an artifact. Fields from computer science to engineering design to systems engineering all attempt to encapsulate, quantify, record, and observe this design intent, and systems have been built to meet this need [23, 10, 8, 51, 19, 59]. This research focuses on capturing functional design intent during the conceptual design phase for electromechanical systems. There are two overarching goals: (1) demonstrate the feasibility of abstracting functional design intent from a form-driven engineering design paradigm, and (2) demonstrate the feasibility of computationally-assisted, form-based design space exploration for engineered systems. The IDEAS (Iterative DEsign of Alternative Solutions) application has been developed to meet these goals.

IDEAS allows designers to explore concepts based on function while not having to formally consider functionality. IDEAS enables design space exploration during the transition from problem clarification to concept generation based on initial form and fit design conceptualizations. The IDEAS application leverages a database of reverse engineered products and algorithms to abstract the underlying functionality associated with form and fit to help explore concepts and component alternatives. A design process using the IDEAS application allows engineers to apply the abstracted functionality as a baseline for generating concept alternatives on their own, but also, allows exploration of concept and component alternatives suggested by the application. Consequently, using IDEAS, an engineer can explore early design concepts without considering function. Instead, conceptual design decisions based on form and fit are used to extract functional information required of a conceptual design and may be used as a starting point for functional-based design using an existing suite of function-based design tools.

2. Background

The IDEAS application leverages prior research efforts in the area of functional modeling and design information capture mechanisms to capture and represent component models based on archived functional information of existing products. The following subsections provide

CONTACT Matt R. Bohm matt.bohm@louisville.edu

background relevant to the architecture and development of the IDEAS application.

2.1. Functional abstractions

Function provides the design intent describing how a design will need to operate before even component systems are selected for the design, and a functional model is an abstraction of a system which allows complex design problems to be simplified into representations more readily solvable using fundamental principles. The roots of flow-based functional modeling during engineering design can be traced back to the field of Value Analysis [41, 48]. In engineering design, a functional model is often a description of a product in terms of the elementary functions and flows that are required to achieve the product's overall function or purpose. A graphical form of a functional model is represented by a collection of sub-functions connected by the flows on which they operate [8, 10]. This structure is a way for a designer to see what types of functions are performed without being distracted by any particular form the artifact may take. The formalization of functional modeling through a common lexicon such as the Functional Basis [28] allows for the creation of a design repository where components, represented by the function-flow pairs, can be archived and used during later design activities. This archival of design intent not only allows the designer to understand the historical reasons why a component or architecture was selected, but also allows better decision making during new design activities. In this research, flow-based functional models based on the Pahl and Beitz methodology [46] was chosen as they represent the interconnectivity of chosen components (through their functional abstraction) in an engineered system. The black box model, illustrated as Fig. 1, describes the high-level transformation intended for systems, and the input and output flows identify all flows required for the operation of the product.

A sub-functional model, illustrated as Fig. 2, decomposes the overall functional black box into specific flow transformations. These transformations define the operations required of the system such that the identified input flows do become the identified output flows through the operation of the system.

2.2. Design information capture mechanisms

Recording and using historical product design information and knowledge, such as function, has been the focus of a significant amount of work including in design theory [21], design evolution, rule-based design strategies [52], electronic documenting and updating [5, 33], Product Data Management (PDM) systems [56], XML integration tools [64], and object oriented structures [34]. Although, many product information management

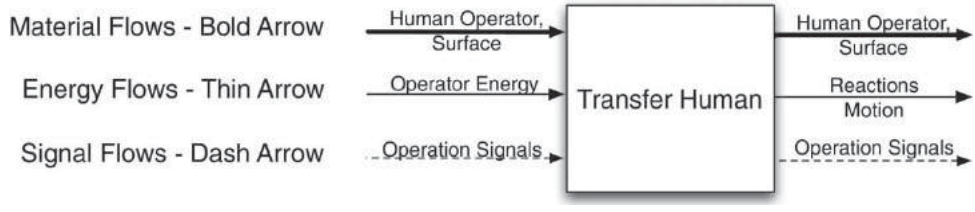

Figure 1. Example black box functional model.

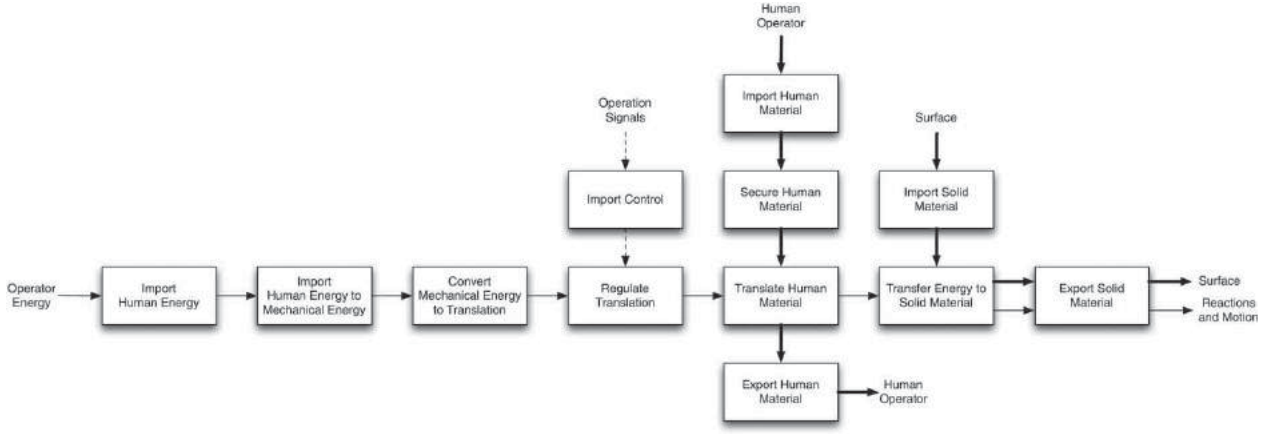

Figure 2. Example sub-functional model.

systems have developed without any coherent and unified structure [17]. In an effort to better standardize these systems the National Institute for Standards and Technology (NIST) developed data exchange standards [22, 1, 47] that still surpass the capabilities of many commercial PDM systems. Digital modeling to support various design activities and decisions include collaborative CAD models [45], feature-based models [55], cooperative visualizations [32], interactive assembly models [4, 19], web-based product visualization tools [20, 35], and web-based distributed product realization support [2, 66, 67]. The Internet information sharing CERN created a new opportunity for product design support, and one area that has grown rapidly is digital libraries or repositories [24]. Some of the early applications of repositories to product design include database-driven fixture design [40], the ability to search, annotate, and organize product models [36, 49], and functional product representations [6, 11, 28].

The research community has also offered up various methods of capturing pieces of design intent information proposing solutions like Design Repositories, SysML/UML based modeling systems, and other schemes to capture design rationale and reasoning. The Systems Modeling Language (SysML) specification was originally published in 2007 by the Object Modeling Group (OMG) [25, 65]. SysML is a graphical modeling language that enables engineers to analyze, specify, design, verify and validate engineered systems. On the other hand, the objective of a design repository is to allow designers to store and retrieve design knowledge at various levels of abstraction, from form (components, subassemblies and assemblies) to architecture description to function. The infrastructure supporting the Design Repository is its information ontology [7]. The information ontology describes what types of design information can be stored, the relationship of those elements and the extensibility of including new and additional types of design information. Through their Design Repository Project, NIST set out to define basic guidelines of a Design Repository and how archived design information could be useful to designers [42, 50, 57]. The NIST Design Repository representation model is a basic framework to help guide what type of product information is collected and how the elements of information are related to each other.

3. Methodology

The Design Repository's [42, 50, 57] archival schema for product information based on functional modeling with the Functional Basis [28] allows the IDEAS application to work. Grammar rules, based on earlier work into functional grammars for both archival an conceptual design [38, 37, 44], provide the algorithmic backbone for IDEAS. The following subsections describe the approach implemented by the IDEAS application.

3.1. Motivation

Current automated methods do not assist with the translation of functional requirements into creative solutions, which is a key component of this research.

3.2. IDEAS application approach

In order to explore the possibility of a form initiated concept generation paradigm, initial steps require a systematic approach to abstracting functional descriptions from an initial form-based concept seed. From there, the abstracted functionality can be used as input to the existing concept generation algorithms—freeing the designer from having to consider function independently of form and fit. The overall approach followed is decomposed into three specific activities. The IDEAS application is the computational implementation of the following approach.

3.2.1. Capture chains of envisioned or existing components for a given concept by using computer parse-able natural-language component terms

Users are allowed to specify an initial solution by listing chains of components envisioned in their product by using an augmented component naming taxonomy [40]. From a computational standpoint, information regarding components infers how the user intends those components to be connected to one another. A basic search of the Design Repository shows prior observable connection orderings (prior archived design intent) of components. A framework allows for an easy and logical manner to gain information about components in a particular concept. With a semi-logical ordering of input components, an algorithm statistically determines intended component order.

3.2.2. Query the design repository for relative functions and flows based upon the natural language component input

Using the user designated component chains, the repository is queried to determine prior design intent related to prior function and flow component solutions. Most artifacts in the repository are given a common name as well as a more accurate component basis taxonomy name [45, 47]. The component naming taxonomy exists

to remove ambiguity from common names and to aid in the clustering of design information.

3.2.3. Apply AI reasoning to derive a functional representation of the concept

Reasoning is one of the major topics of research within the AI community [6, 11, 28]. Prior research has produced grammar rules that generate function structures from overall input and outputs of a product [7, 33] and to transform functions to components (that solve the functionality) [1]. To understand function-flow connectivity of prior engineered systems, four stages of grammar rules have been developed based on the function and flow terms of the Functional Basis.

3.2.4. Computational implementation – the IDEAS application

There are four underlying mechanisms to the IDEAS application: (1) the capture and representation of component models; (2) search algorithms that operate on a repository of design information to determine discrete component functionality; (3) grammar rules to synthesize a continuous functional representation of the design problem; and (4) concept generation techniques to search for and synthesize alternative concepts.

3.2.5. Capture and representation of component models

Components are navigated and selected using a component naming taxonomy [20], or alternatively, can be searched through direct term match and natural language synonyms [66] simultaneously. Once a component is identified, the user is shown an icon view along with a brief textual description and can choose to add the component to the diagram or continue to search for alternative components. After component(s) are added to the working diagram, their directional relationships to one another are defined by first selecting the arrow tool and then tapping the two components that are to be connected together. Directionality is established by selecting the source component first and the destination component second. The component connection scheme resembles a graph consisting of nodes (components) and arcs (connections) and is similar to a configuration flow graph [4, 24, 40, 67].

3.2.6. Search the repository to determine discrete component functionality

At this stage, designer preferences are needed in order to provide additional information such that computational reasoning about the intended use of their product may proceed. Data from the repository will be utilized to determine which functions and flows are solved by each of the given input components. Most artifacts in the repository are given a common name as well as a more accurate component basis taxonomy name. For example, a user might list "small dc motor" as a common name, but also choose "motor" from the component naming taxonomy. The component naming taxonomy exists to remove ambiguity from common names and to aid in the clustering of design information. Across the entire repository, each component naming term is associated with, on average, 17.7 unique function-flow pairs. This non-exclusive relationship between function flow pairs and components occurs because some components solve more than one function for a particular implementation and some components have multiple distinct uses. It is therefore necessary to determine which function(s) and flow(s) are intended by the user's selection of a particular component. Examination of repository data shows that, in general, 70% of both functions and flows are realized within the first 30% of unique instances of a particular component. This finding suggests that the 70/30 allocation is Pareto optimal [36, 49]. The outcome of this step in the process is a rank ordered list of functions and flows that are associated with each component that has been identified by the user.

3.2.7. Apply grammar rules to synthesize a continuous functional representation

The IDEAS application uses two stages of grammar rules to synthesize a continuous functional representation of a design problem. Grammar rules are associated with individual functions and dictate allowed incoming and outgoing flows. The approach is semi-automated and requires user interaction to designate an initial input flow at the beginning of the functional model generation process. Descriptions of the two stages follow.

- ***Stage 1 grammar rules*** are associated with individual functions and dictate the allowed incoming and outgoing flows. A set of rules has been developed that allows for a semi-automated approach to functional model generation. In the semi-automated approach, the user is asked to designate a flow at the input of a function chain when multiple flows are associated with a function or when a function definition states that an output flow must be different from an input flow. Previous research has concluded that the secondary level of the Functional Basis is sufficient for most types of representation [60], and as such, the grammar rules are only associated with the secondary level of the Functional Basis. The following definitions and global rule for each tier are used: Continuing Flow is a flow that is both the output of the previous function and the input to the next function. Dangling

Flow is a flow that is connected to a single function (incoming or outgoing) but does not continue to or originate from another function within the functional model. As a global rule, no functions may be duplicated sequentially.

- ***Stage 2 grammar rules*** are intended to be applied as post-processing technique to increase the accuracy of generated functional models. Foundations for a portion of the Stage 2 grammar rules stem from research efforts involving flow classification schemes denoting flows as either being a primary or carrier flow [42, 58]. The primary and carrier flow grammar rules work to introduce carrier flows to energies present in the generated functional model.

3.2.8. Generation of alternative design concepts4

The IDEAS application makes use of an existing concept generation method known as MEMIC (Morphological Evaluation Machine and Interactive Conceptualizer) [2, 5, 42]. Another computational concept generation method translates an input function structure into a matrix form that describes the adjacency between functions. This input undergoes a series of matrix multiplications that map functionality to solutions (components) and filters out component-to-component connections that are not possible based on repository data [42]. The output is a set of concept variants solving the input functionality [42].

4. The IDEAS application – implementation and example

The initial application requirements and their translation into technical requirements are described in Section 4.1. This section also contains conceptual sketches and mock-ups of the user interface and its development. An example of the IDEAS application is described in Section 4.2.

4.1. Requirements and implementation of the application

The IDEAS concept involved specification of components with an end goal of having an application that can suggest functionality or alternate components to be used in a conceptual design. The initial requirements for the application were as follows.

- Users must be able to easily navigate through a hierarchy of components.
- Users must be able to "build-up" their ideas by adding components.
- Users must be able to specify some type of component connectivity.
- The application must incorporate a mechanism to display underlying functionality once a component-based design has been specified.
- The application must incorporate a mechanism to display alternative components.
- The application must be responsive and load quickly.
- The application must provide the ability to save projects and work on multiple different projects.
- The application must meet the guidelines for posting on the Apple App Store™.

The initial customer needs were translated into more technical requirements in the following categories: feature requirements, interface requirements, performance requirements and security requirements. For brevity, Table 1 shows the technical description and specification for only three of the feature requirements.

Before actual interface coding began, several sketches and mock-ups of various interfaces were generated. Sketches went through several iterations with some of the early sketches being very basic menu structures and nested types of navigation. As the understanding of the problem progressed, the sketches became more detailed with the addition of a search field for components and multiple menu bar tools to assist with diagram generation. The sketches are a subset of all of the sketches created. During the design process, feedback was received from mechanical engineering and computer science faculty regarding the desired features of the application and user interface interaction issues.

4.2. Application example

Upon launching the application, the application establishes a connection to a PostgreSQL database hosted at

Table 1. Application technical requirements.

Retrieve List of Components The application interfaces with the database server and retrieves a list of components, grouped categorically. The list is presented to the user and can be navigated and searched. The interaction with, and navigation through, the list should be quick and intuitive. Search is done through the use of component *tags*—an alternative name for the same components comprised of natural language synonyms.
Manipulate Artifacts in a Component Model When the user selects an artifact to insert into the component model, the representation of the component will appear. From there, the user can move, resize, or rotate the selected component into their desired location and orientation. The user can further connect components to each other through the use of an icon. These interactions are kept on a user move stack, and can be undone by using the undo button. Components can also be deleted and renamed.
Save/Load Model The user creates and interacts with models, and states are stored continuously saved during interaction. At any time, the user can load a different saved model or create a new model. Models (as either component or function diagrams) can be shared through email.

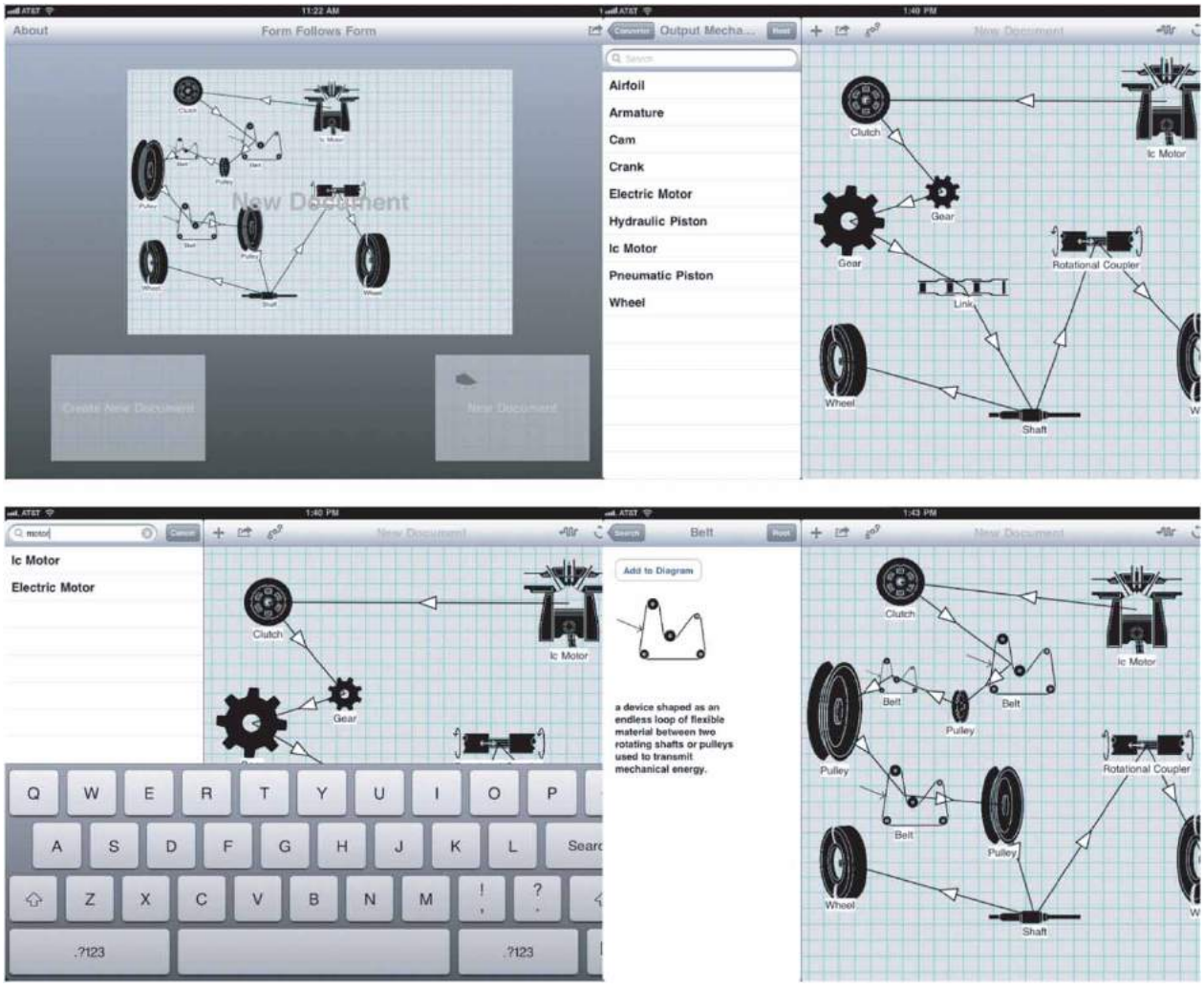

Figure 3. Operational views: (a) Startup Screen, (b) Navigation Split View, (c) Search View, (d) Alternative Concept.

the University of Louisville. The user can choose to create a new diagram, open an existing diagram, delete an existing diagram, or email any existing diagrams as an attachment (Fig. 3a). When creating a new diagram, or opening an existing diagram, the user is presented with a split-pane view. Within the left pane is a list of artifacts retrieved from the database organized by category and subcategory (Fig. 3b). This list can be searched through by name or component tag (Fig. 3c).

When the user selects a desired artifact, the left pane switches to a component display showing the component representation (now a line drawing of the component in question) and a short description of what the component does. The user can tap the "Add to Diagram" button to add the component to the diagram in the right pane. The right pane is the main interface of the application.

Along the top bar are the following: new diagram icon, email diagram icon, view functional diagram icon, the diagram name, the flow tool icon, undo last action icon, and delete component icon. Tapping on the diagram name allows the user to rename the diagram; the same method of renaming is used for renaming components within the diagram. The new diagram icon saves the current diagram state and allows the user to create a new blank diagram, duplicate the current diagram, or return to the main menu of the application. The email diagram icon shows a popover giving the user a choice of what to email; the user can email the entire diagram save file, the diagram screenshot, or the functional model screenshot. Tapping the view functional model icon shows the user the functional model from the current diagram state (discussed later). When the user taps the flow tool icon, and then two different components, a link is drawn between the components in the order they were selected. The undo icon undoes the latest user action: placing, moving, resizing, rotating, or linking of any component. The undo stack holds the last ten user actions. The delete icon

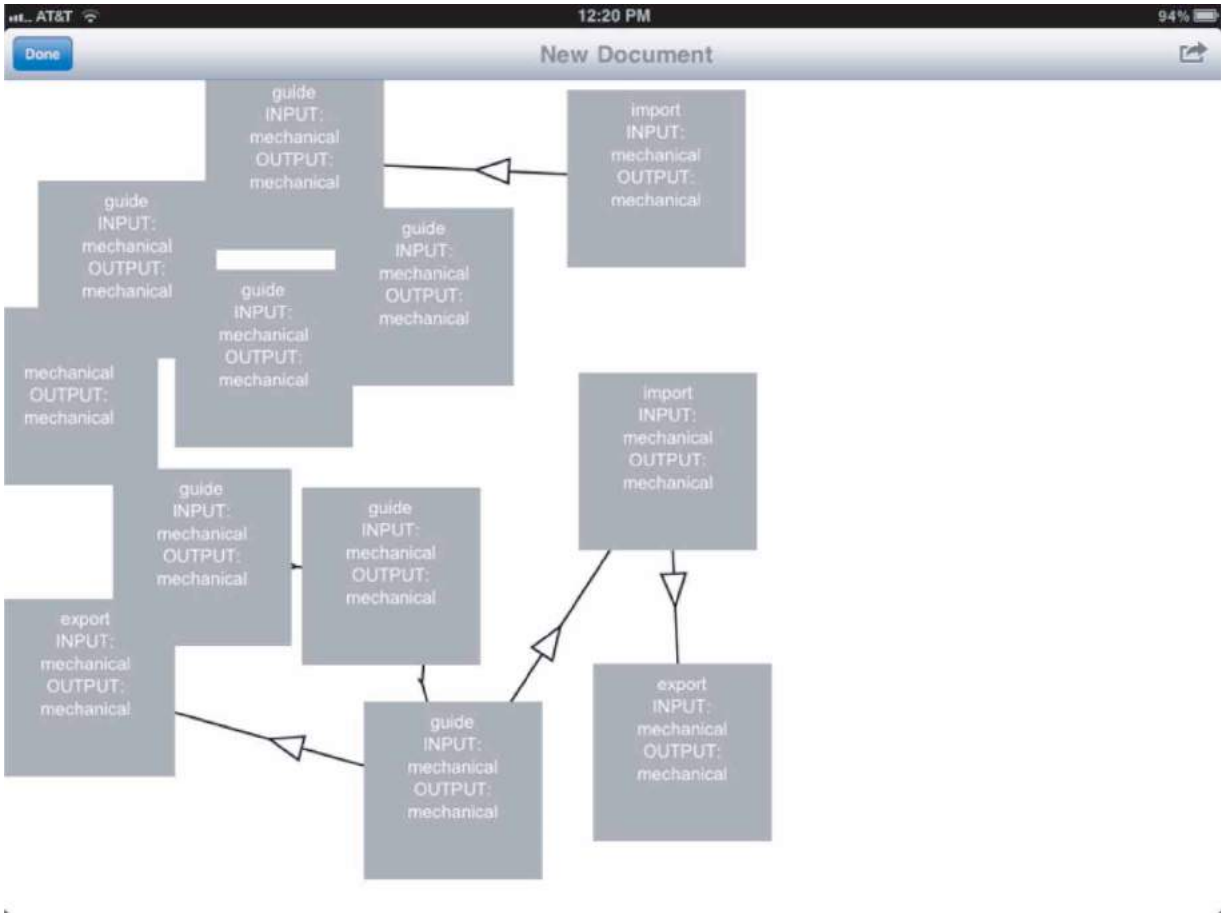

Figure 4. Functional view of the chosen solution concept.

deletes the next selected component and all links to or from the component.

The functional model view converts the component diagram, as it is displayed on the screen of the application, and displays its associated functional model. The functions associated with the components are retrieved from the database server, and flows are interpreted from component functions and input/output flow types. Function blocks are drawn and label fields are inserted within the blocks. A toolbar contains a back button, to return to the component diagram view, an email icon, and the diagram's title.

The screenshots shown in this Section (Fig. 3a-3c) illustrate a high level view of an automotive drive system. Initially, in this example, the application's user (Fig. 3b) specifies the use of gears and chains to transmit mechanical energy from the engine to the drive wheels in the design of a automobile drive system. The alternative design, illustrated in Fig. 3d, suggested by the IDEAS application makes use of a belt and pulley system to transmit the mechanical energy. Although this example is very high level, it illustrates the simplicity and power of being quickly shown different design alternatives—think regular gear driven transmission versus a continuously variable transmission.

Figure 4 shows the underlying functionality of a portion of the automotive system (to see the full set of functions the user must scroll around the page). The functions shown may not be perfectly synthesized; however, it does demonstrate that a user can quickly see the core functionality that underlies their specified components. The underlying data structure of the functional model view is a graph structure with arcs and nodes are annotated with functions and flows as properties. It is possible to export the function structure in various formats such that other computational platforms can reuse the information.

5. Discussion

The IDEAS application builds on a body of research in the area of product archival—specifically the Design Repository which started development in the early 2000 s [8]. As mentioned in the Background section there are several methods of storing and using design intent information

with each method serving a respective need area. The Design Repository used by the IDEAS application contains a set of information more tailored for use in the conceptual phase of design; specifically, information useful for identifying potential alternatives. To increase the usefulness and potential of the IDEAS application the underlying information database needs to be expanded to incorporate additional types of information such as mathematical models, manufacturing techniques, etc., such that designers can begin to consider tradeoffs and issues related to production.

Grammar rules based on likely product functionality derived from known product architectures captured and stored in the Design Repository are used to arrive at functional design intent based on user selected solutions (form) and their chosen interaction (fit) [18, 43, 9]. Frequently, as designs move from the early phases (problem clarification and concept generation) into the later phases (design embodiment and detail design), functional design intent identified early in the design process is either lost or disassociated from both the concepts explored and the resultant decisions. The overarching goals in creating the IDEAS application are to demonstrate the feasibility of form and fit based design space exploration for engineered systems, and to demonstrate the feasibility of abstracting functional design intent from a form and fit based engineering design paradigm. With the completion of the phase one prototype for the IDEAS application (presented herein), these goals have been achieved. Meeting these goals is the first step toward developing a computational system that allows functional design intent to be captured and archived during conceptual design for use later in the design process and with an existing suite of function-based design tools. Once a digital representation of a functional model is generated and abstracted, the data can be ported into other applications for concept generation [13, 15, 14, 62, 38, 63, 16] and visualization [3], failure [54, 53, 61, 29] and risk analysis [39, 27, 26], mathematical simulation [31, 30] and design archival [57, 12, 8, 6] or the model can be used as is.

Two avenues of continued research are being pursued as future work. First is to further encapsulate additional pieces of design intent. Using the Design Repository as the computational backend for the IDEAS application can allow for anticipated intent to be prepopulated into a design solution. The design engineer can either accept the prepopulated design intent or can update with the system noting designer preferences for use in future design solutions. With a larger body of design information being captured during conceptual design, it becomes important for this information to travel with the design through the design process. Currently, the IDEAS application is limited to the conceptual design phase; future iterations will need to connect with existing PLM software systems in order for broader adoption to be feasible.

The second avenue of continued research is to explore the impact of using the IDEAS application on student learning of the engineering design process. In teaching engineering design, we have found that when students first learn the engineering design process, they tend to fixate on the first or second solution to which they arrive without considering alternatives. However, without fully exploring the solution space, it is impossible for a designer to know with any level of confidence that a proposed design is the best design for the customer. We postulate that this design fixation often negatively influences the student's overall impression of engineering design. Our hypothesis is that students who use the IDEAS application in addition to traditional ideation approaches will develop a larger pool of possible solutions, and consequently, will learn the value of more fully exploring the solution domain. Longitudinal and cross-sectional studies are being developed to test explore this hypothesis.

6. Conclusions

The goals of the IDEAS application creation were achieved upon the completion of the phase one prototype and the implementation of the grammar rules outlined in this paper. Goals of the phase one prototype of the IDEAS application included demonstration of the feasibility of form and fit based design space exploration for engineered systems and demonstration of the feasibility of abstracting functional design intent from a form and fit based engineering paradigm. Achieving these goals is the first step toward developing a computational system that allows functional design intent to be captured and archived during conceptual design for use later in the design process.

Future work involves expansion of the underlying information database of the IDEAS application and connection to existing PLM software systems in order for broader adoption to be feasible. Longitudinal and cross-sectional studies are being developed in order to explore the impact of using the IDEAS application on student learning of the engineering design process.

Acknowledgements

We would like to thank Dr. Seth Orsborn at Bucknell University for providing the component icon images used in the Form Follows Form iPad application and University of Louisville students Jonathan Miller, Udit Bajaj, Shamir Patel, Kyle Underwood, and Camden Knight for their work in developing the first prototype of the IDEAS application.

ORCiD

Matt R. Bohm http://orcid.org/0000-0002-9598-633X
Robert L. Nagel http://orcid.org/0000-0003-4539-7964
Marie K. Riggs http://orcid.org/0000-0002-0882-9476

References

[1] 10303-1:1994, I.: Industrial Automation Systems and Integration Product Data Representation and Exchange - Overview and Fundamental Principles, International Standard, 1994.

[2] Allen, R.; Nidamarthi, S.; Regalla, S.; Siriram, R.: Enhancing Collaboration Using an Internet Integrated Workbench, Proceedings of the 1999 ASME Design Engineering and Technical Conference, DETC99/DAC-8573 1999.

[3] Attaluri, V.; McAdams, D. A.; Stone, R. B.; Crescenzo, A. D.: Visual Representations as an Aid to Concept Generation, Proceedings of the 2006 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2006-99572 2006, http://dx.doi.org/10.1115/DETC2006-99572.

[4] Bauer, M.; Siddique, Z.; Rosen, D. W.: A Virtual Prototyping System for Design for Assembly, Disassembly, and Service, Journal of Agile Manufacturing, 2 (2), 1999, 119–138.

[5] Benami, O.; Jin, Y.: An E-Documenting Approach to Conceptual Design, Proceedings of the 2000 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC/CIE-14649 2000.

[6] Bohm, M.; Stone, R.: Product Design Support: Exploring a Design Repository System, Proceedings of the 2004 ASME International Mechanical Engineering Congress, IMECE2004-61746 2004, http://dx.doi.org/10.1115/IMECE2004-61746.

[7] Bohm, M.; Stone, R.; Simpson, T.; Steva, E.: Introduction of a Data Schema: To Support a Design Repository, Computer-Aided Design, 40(7), 2008, 801–811, http://dx.doi.org/10.1016/j.cad.2007.09.003.

[8] Bohm, M.; Stone, R.; Szykman, S.: Enhancing Virtual Product Representations for Advanced Design Repository Systems, Journal of Computer and Information Science in Engineering, 5 (4), 2005, 360–372, http://dx.doi.org/10.1115/1.1884618.

[9] Bohm, M.; Stone, R.: Form Follows Form: Fine Tuning Artificial Intelligence Methods, Proceedings of the 2010 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2010-28774 2010, http://dx.doi.org/10.1115/DETC2010-28774.

[10] Bohm, M.; Vuchovich, J.; Stone, R.: An Open Source Application for Archiving Product Design Information, Proceedings of the 2007 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2007-35401 2007, http://dx.doi.org/10.1115/DETC2007-35401.

[11] Bohm, M.; Vucovich, J.; Stone, R.: Capturing Creativity: Using a Design Repository to Drive Concept Innovation, Proceedings of the 2005 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC 2005-85105 2005, http://dx.doi.org/10.1115/DETC2005-85105.

[12] Bohm, M. R.; Stone, R. B.; Simpson, T. W.; Steva, E. D.: Introduction of a Data Schema: The Inner Workings of a Design Repository, Proceedings of the 2006 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 2006, http://dx.doi.org/10.1115/DETC2006-99518.

[13] Bohm, M. R.; Vucovich, J. P.; Stone, R. B.: Using a Design Repository to Drive Concept Generation, Journal of Computer and Information Science in Engineering, 8(1), 2008, http://dx.doi.org/10.1115/1.2830844.

[14] Bryant, C.; Stone, R.; McAdams, D.; Kurtoglu, T.; Campbell, M.: Concept Generation from the Functional Basis of Design, Proceedings of the International Conference on Engineering Design, 2005.

[15] Bryant, C. R.; Bohm, M. R.; Stone, R. B.; McAdams, D. A.: An Interactive Morphological Matrix Computational Design Tool: A Hybrid of Two Methods, Proceedings of the 2007 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2007-35583 2007, http://dx.doi.org/10.1115/DETC2007-35583.

[16] Bryant, C. R.; Stone, R. B.; McAdams, D. A.: Concept Generation from the Functional Basis of Design, Proceedings of the International Conference on Engineering Design, 2006.

[17] Byron, B. M.; Shooter, S. B.: A Review of Software Solutions for the Management of New Product Development and Product Family Planning, Proceedings of the 2005 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2005/DAC-84454 2005, http://dx.doi.org/10.1115/DETC2005-84454

[18] Campbell, M.; Kurtoglu, T.; Rahul, R.: A Stochastic Graph Grammar Algorithm for Interactive Search, Proceedings of the 2009 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2009-86804 2009, http://dx.doi.org/10.1115/DETC2009-86804.

[19] Chen, L.; Wang, T.; Song, Z.: A Web-based Product Structure Manager to Support Collaborative Assembly Modeling, Journal of Computing and Information Science in Engineering, 4(1), 2004, 67–78, http://dx.doi.org/10.1115/1.1666894.

[20] Chung, M. J.; Keyes, D.; Choi, Y.; Kwon, P.; Gu, H.; Behr, M.: A web-Based Framework for Design and Manufacturing a Mechanical System, Proceedings of the ASME Design Engineering and Technical Conference, DETC98/DAC-5599 1998.

[21] Dixon, J.; Libardi, E.; Luby, S.; Vaghul, M.; Simmons, M.: Expert Systems for Mechanical Design: Examples of Symbolic Representations of Design Geometries, Engineering Computing, 2(1), 1987, 1–10, http://dx.doi.org/10.1007/BF01200172.

[22] Fenves, S.: A Core Product Model for Representing Design Information, NISTIR 6736 2001, 1–38, http://dx.doi.org/10.1115/1.2830842.

[23] Fenves, S.; Foufou, S.; Bock, C.; Sriram, R.: CPM2: A Core Product Model for Product Data, Journal of

Computer and Information Science in Engineering, 8 (Special Issue on Engineering Informatics), 2008, http://dx.doi.org/10.1115/1.2830842.
[24] Fox, E.; Akscyn, R.; Furuta, R.Leggett, J.: Digital Libraries, Communications of the ACM, 38(4), 1995, 23–28. http://dx.doi.org/10.1145/205323.205325.
[25] Friedenthal, S.; Moore, A.; Steiner, R.: A Practical Guide to SysML: Systems Model Language, 2008.
[26] Grantham Lough, K.; Stone, R. B.; Tumer, I. Y.: The Risk in Early Design (RED) Method: Likelihood and Consequence Formulations, Proceedings of the 2006 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2006-99375 2006, http://dx.doi.org/10.1115/DETC2006-99375.
[27] Grantham Lough, K.; Stone, R. B.; Tumer, I. Y.: The Risk in Early Design Method (RED), Journal of Engineering Design, 18(1), 2007, http://dx.doi.org/10.1080/09544820701684271.
[28] Hirtz, J.; Stone, R.; McAdams, D.; Szykman, S.; Wood, K.: A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts, Research in Engineering Design, 13(2), 2002, 65–82, http://dx.doi.org/10.1007/s00163-001-0008-3.
[29] Hutcheson, R. S.; McAdams, D. A.; Stone, R. B.; Tumer, I. Y.: A Function-Based Methodology for Analyzing Critical Events, Proceedings of the 2006 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC 2006-99535 2006, http://dx.doi.org/10.1115/DETC2006-99535.
[30] Hutcheson, R. S.; McAdams, D. A.; Stone, R. B., Tumer, I. Y.: Function-Based Behavioral Modeling, Proceedings of the 2007 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2007-35337 2007, http://dx.doi.org/10.1115/DETC2007-35337.
[31] Hutcheson, R. S.; McAdams, D. A.; Stone, R. B.; Tumer, I. Y.: Function-based Systems Engineering (FuSE), Proceedings of the International Conference on Engineering Design, 620 2007.
[32] Isenhour, P.; Begole, J.; Heagy, W.; Shaffer, C.: Sieve: A Java-Based Collaborative Visualization Environment, IEEE Visualization '97 Late Breaking Hot Topics, 1997, 13–16.
[33] Jayaram, S.; Jayaram, U.; Kreitzer, K.: Preserving Design Intent in Data Integration Between Virtual Prototyping and CAD Systems, Proceedings of the ASME Design Engineering and Technical Conference, DETC98/CIE-5708 1998.
[34] Karne, R. K.; Dandekar, S.; Poluri, S.; Chen, G.; Baras, J.; Nau, D.; Ball, M.; Lin, E.; Trichur, V.; Williams, J.: Web-IT-Man: A Web-Based Integrated Tool for Manufacturing Environment, Proceedings of the ASME Design Engineering and Technical Conference, DETC98/CIE-5524 1998.
[35] Katwyk, K.; Cheng, H.: XLINKAGE: A Web-Based Analysis and Simulation Tool for Planar Mechanical Systems, Proceedings of the ASME Design Engineering and Technical Conference, DETC97/DAC-3863 1997.
[36] Kopena, J.; Cera, C. D.; Regli, W.: Conceptual Design Knowledge Management and the Semantic Web, Proceedings of the 2005 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2005/CIE-85310 2005, http://dx.doi.org/10.1115/DETC2005-85310.
[37] Kurtoglu, T.; Campbell, M.; Bryant, C.; Stone, R.; McAdams, D.: Deriving a Component Basis for Computational Functional Synthesis, Proceedings of the International Conference on Engineering Design, 2005.
[38] Kurtoglu, T.; Campbell, M. I.: Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping, Journal of Engineering Design, 20(1), 2009, 83–104, http://dx.doi.org/10.1080/09544820701546165.
[39] Lough, K. G.; Stone, R. B.; Tumer, I.: Implementation Procedures for the Risk in Early Design (RED) Method, Journal of Industrial and Systems Engineering, 2(2), 2008, 126–143.
[40] Ma, W.; Lei, Z.; Rong, Y.: FIX-DES: A Computer-aided Modular Fixture Configuration Design System, International Journal of Advanced Manufacturing Technology, 14, 1998, 21–32, http://dx.doi.org/10.1007/BF01179413.
[41] Miles, L.: Techniques of Value Analysis and Engineering, 2 1972.
[42] Murdock, J.; Szykman, S.; Sriram, R.: An Information Modeling Framework to Support Design Databases and Repositories, Proceedings of ASME Design Engineering and Technical Conference, DETC97/DFM-4373 1997.
[43] Nagel, R.; Vucovich, J.; Stone, R.; McAdams, D.: A Signal Grammar to Guide Functional Modeling of Electromechanical Products, Journal of Mechanical Design, 130(5), 2008, 051101-1-10, http://dx.doi.org/10.1115/1.2885185.
[44] Nagel, R.; Vucovich, J.; Stone, R.; McAdams, D.: A Signal Grammar to Guide Functional Modeling of Electromechanical Products, Journal of Mechanical Design, 130(5), 2008, 051101-1-10, http://dx.doi.org/10.1115/1.2885185.
[45] Okudan, G.; Medeiros, D.: Facilitating Collaborative Design: A Review on Design Representations and Workstations, Proceedings of the 2005 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2005/DAC-85124 2005, http://dx.doi.org/10.1115/DETC2005-85124
[46] Pahl, G.; Beitz, W.; Schulz, H.; Jarecki, U.: Engineering Design: A Systematic Approach, 2007, http://dx.doi.org/10.1007/978-1-84628-319-2.
[47] Pratt, M. J.: Introduction to ISO 10303 - The STEP Standard for Product Data Exchange, Journal of Computing and Information Science in Engineering, 1 (4), 2001, 138–165.
[48] Rodenacker, W.: Methodisches Konstruieren (Methodical Design), 1971.
[49] Shaffer, J.; Kopena, J.Regli, W.: Web Service interfaces for Design Repositories, Proceedings of the 2005 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2005/CIE-85368 2005, http://dx.doi.org/10.1115/DETC2005-85386.
[50] Shooter, S.; Keirouz, W.; Szykman, S.; Fenves, S.: A Model For Information Flow In Design, Proceedings of the

ASME Design Theory and Methodology Conference, DETC2000/DTM-14550 2000.

[51] Shooter, S. B.; Simpson, T. W.; Kumara, S. R. T.; Stone, R. B.; Terpenny, J. P.: Toward an Information Management Infrastructure for Product Family Planning and Platform Customization, Proceedings of the 2004 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2004/DAC-57430 2004, http://dx.doi.org/10.1115/DETC2004-57430.

[52] Stahovich, T. F.; Raghavan, A.: Computing Design Rationales by Interpreting Simulations, Journal of Mechanical Design, 122 (1), 2000, 77–82, http://dx.doi.org/10.1115/1.533547.

[53] Stock, M.; Stone, R.; Tumer, I. Y.: Going Back in Time to Improve Design: The Function-Failure Design Method, Proceedings of the 2003 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2003/DTM-48638 2003, http://dx.doi.org/10.1115/DETC2003/DTM-48638.

[54] Stone, R. B.; Tumer, I. Y.; Van Wie, M.: The Function-Failure Design Method, Journal of Mechanical Design, 127 (3), 2005, 397–407, http://dx.doi.org/10.1115/1.1862678.

[55] Summers, J.; Maxwell, D.; Camp, C.; Butler, A.: Features as an Abstraction for designer convenience in the Design of Complex Products, Proceedings of ASME Design Engineering Technical Conference, DETC2000/CIE-14642 2000.

[56] Svensson, D.; Malmqvist, J.: Integration of Requirement Management and Product Data Management Systems, Proceedings of ASME Design Engineering Technical Conference, DETC2001/CIE-21246 2001.

[57] Szykman, S.: Architecture and Implementation of a Design Repository System, Proceedings of the 2002 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2002/CIE-34463 2002, http://dx.doi.org/10.1115/DETC2002/CIE-34463.

[58] Szykman, S.; Racz, J.; Sriram, R.: The Representation of Function in Computer-Based Design, Proceedings of the ASME Design Engineering Technical Conferences, DETC99/DTM-8742 1999.

[59] Szykman, S.; Sriram, R.; Regli, W.: The Role of Knowledge in Next-generation Product Development Systems, Journal of Computer and Information Science in Engineering, 1(1), 2001, 3–11, http://dx.doi.org/10.1115/1.1344238.

[60] Szykman, S.; Sriram, R.; Smith, S.: Proceedings of the NIST Design Repository Workshop, 1996.

[61] Tumer, I. Y.; Stone, R. B.: Analytical Methods for Mapping Function to Failure During High-Risk Component Development, Research in Engineering Design, 14 (1), 2003, 25–33.

[62] Van Wie, M.; Bryant, C.; Bohm, M.; McAdams, D.; Stone, R.: A general model of function-based representations, Artificial Intelligence in Engineering Design, Analysis and Manufacture, 19(2), 2005, 89–111, http://dx.doi.org/10.1017/S0890060405050092.

[63] Vucovich, J.; Bhardwaj, N.; Ho, H.-H.; Ramakrishna, M.; Thakur, M.: Concept Generation Algorithms for Repository-Based Early Design, Proceedings of the 2006 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 2006, http://dx.doi.org/10.1115/DETC2006-99466.

[64] Wang, L.; Ting, K.; Kosa, M.: XML-Based Integration of Design, Analysis and Manufacturing, Proceedings of the 2003 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2003/CIE-48231 2003, http://dx.doi.org/10.1115/DETC2003/CIE-48231.

[65] Weilkiens, T.: Systems engineering with SysML/UML: modeling, analysis, design, 2007.

[66] Xiao, A.; Choi, H.; Kulkarni, R.; Allen, J.; Rosen, D. W.; Mistree, F.: A Web-Based Distributed Product Realization Environment, Proceedings of the ASME Design Engineering and Technical Conference, DETC2001/CIE-21766 2001.

[67] Yang, Q., Lu, W. F.: A Web-Enabled Engineering Object Modeling Environment to Support Interoperability and Intelligent Services in Collaborative Design, Proceedings of the 2005 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, DETC2005/DAC-84240 2005, http://dx.doi.org/10.1115/DETC2005-84240.