Res Eng Design (2011) 22:223–243
DOI 10.1007/s00163-011-0108-7

ORIGINAL PAPER

Early design interference detection based on qualitative physics

Valentina D'Amelio · Magdalena K. Chmarra · Tetsuo Tomiyama

Received: 7 December 2009 / Accepted: 15 March 2011 / Published online: 3 April 2011

Abstract Developing multi-disciplinary products presents cross-disciplinary problems that are difficult to predict and to solve. Unfortunately, those cross-disciplinary problems are often discovered only at a later stage of the design through physical prototypes and can lead to modification of the conceptual design of a product. This is extremely costly and time consuming. This paper describes a new software tool, a Design Interference Detector (DID), which based on qualitative reasoning infers possible problematic physical phenomena that may appear in a design. However, qualitative reasoning techniques often reveal a shortcoming of generating too many negligible solutions. This is a burden to the designer and makes qualitative reasoning practically unusable. Therefore, we developed two filtering methods that filter out such negligible solutions and highlight only potential cross-disciplinary problems. DID with these filtering methods aims particularly at supporting redesign of complex multi-disciplinary products. The paper analyzes advantages and limitations of the filtering methods through a case study.

Keywords Multi-disciplinary design · Qualitative physics · Complexity management · Conceptual design

1 Introduction

To deal with increasing complexity of modern products, we need product development methodologies such as the V-model by Stevens et al. 1998 (Fig. 1). In the V-model, product design starts with requirements analysis, which considers possible conflicting needs of various stakeholders. Then the system design begins, which derives system specifications from system requirements and chooses main concepts of the product. The overall system concept is quickly decomposed into subsystems, and eventually, components are designed. Then, components and subsystems are integrated and tested through prototypes. Subsystems should clearly be defined and understood because they belong to one small module of a product. Although each subsystem is supposed to behave as specified, designers can still be surprised by unpredicted phenomena that deteriorate the performance of the product during the integrated system test. This is due to neglected or overlooked interactions among subsystems.

This type of problems could be hard to solve or even hard to detect. As a consequence, reaching a final satisfactory product can be time consuming and cost inefficient because of additional iterations between designing and prototyping in the development process. In order to reduce these interactions, we need to detect such unpredicted phenomena as early as possible in the design process. In our previous paper (D'Amelio and Tomiyama 2007), we proposed the concept of the Design Interference Detector (DID) that performs an early verification of the system design in order to reduce the probability of having design failures caused by unpredicted phenomena (Tomiyama et al. 2007). This paper reports the software implementation of DID with a focus on inferring algorithms of unpredicted phenomena.

DID is based on qualitative reasoning, which is explained in Sect. 2.3. One advantage of qualitative reasoning is that it does not require precise information in describing a physical system. Therefore, qualitative reasoning is appropriate for the early design phases (i.e. in

V. D'Amelio (✉) · M. K. Chmarra · T. Tomiyama
Faculty of Mechanical, Maritime, and Materials Engineering,
Delft University of Technology, Mekelweg 2,
2628 CD Delft, The Netherlands
e-mail: valentina.damelio@gmail.com

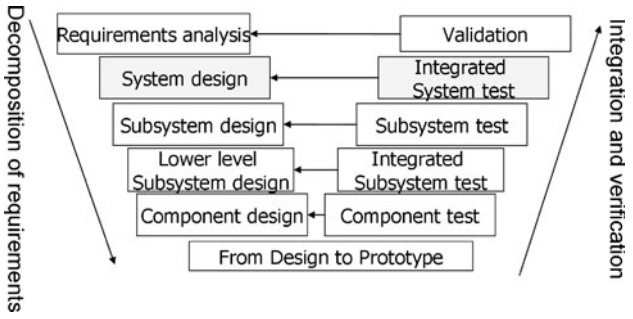


Fig. 1 V-Model representing the product development process

conceptual design) where available knowledge is incomplete or primarily qualitative (Will 1991). Another advantage is that it allows building a model of a fairly complex system in a short time. However, qualitative reasoning presents drawbacks as well. One of them is that it generates a large number of feasible solutions that are difficult to verify for a single designer. The majority of these solutions can be neglected because their magnitude is very small or because they are spurious. Spurious phenomena appear due to the lack of information about the system.

This paper proposes two methods, the contrast and interaction finding methods based on heuristics to filter out negligible solutions generated during qualitative reasoning about a physical system. Figure 2 depicts the computational process of these two methods and can be used as a navigation guide throughout this paper.

The contrast method assumes that most of design activities in industry are modification-based design in which a new product is not designed from a scratch but from an existing design. It supports the redesign (modification design) phase by comparing old and new designs of a product in order to understand the consequences of design changes. By referring to Fig. 2, the design starts with an old design of a product (Process 1) built with the Function Behavior State (FBS) model (Umeda et al. 1996) (Sect. 2.2). Second, the designer makes a new product in a redesign process by adding, removing, and changing structural and/or behavioral knowledge (Process 2- Sect. 3.3). When the new design is finished, DID automatically infers all the possible physical phenomena that can appear in both new and old designs (Process 3—Sect. 2.4) and classifies these inferred physical phenomena into three classes PP+, PP−, and PP= (Process 4). PP+ class contains phenomena that appeared in the new design due to new components or new combination of components in the system. PP− class contains physical phenomena that are negligible in the new design. PP= represents phenomena that occur to both old and new designs. This categorization helps the designer to identify the occurrence of physical phenomena that can potentially lead to unpredicted problems.

The interaction finding method supports the system design (see Fig. 1) where known subsystems are integrated in an unknown combination. This is described in Sect. 3.4. Differently from the contrast method, in the interaction finding method, the design starts with combining product modules (Process 1 of Fig. 2), after which the verification of the design based on the detection of unpredicted phenomena is performed, and finally the classification of physical phenomena into the classes PP+, PP−, and PP= (Process 4). PP+ class represents phenomena that derive from the interaction of the different subsystems.

In the discussions, we argue the limitations of this research and point out future work. The conclusions provide the reader with the main points of this paper and describe the relevance of the filtering methods for the conceptual design of a product.

2 Background

DID is implemented in a software tool called KIEF (Knowledge Intensive Engineering Framework) (Yoshioka et al. 2000, 2004, Yoshioka 2008) and uses the Function Behavior State (FBS) modeler (Umeda et al. 1996) and Physical Feature Reasoning System of KIEF (Yoshioka et al. 2000, 2004). The Physical Feature Reasoning System is based on Qualitative Process Theory (Forbus 1984; Barr et al. 1989). All these tools and methods used by DID are explained in the following subsections.

2.1 Knowledge intensive engineering framework

Knowledge Intensive Engineering Framework (KIEF) is a knowledge intensive framework implemented in Smalltalk. KIEF consists of several modelers, a central metamodel to match information from those modelers, and a knowledge base that contains physical concepts such as entities, relations, and physical phenomena. Definitions of these concepts are based on the ontology described in (Yoshioka et al. 2000, 2004).

- Entity—an atomic physical object or physical component.
- Relation—a physical relation among entities, which illustrates how entities are connected to each other.
- Attribute—a concept attached to an entity, which has a value to indicate the state of that entity.
- Physical phenomenon—a concept that designates physical laws or rules that govern behaviors.
- Physical law—relationships among attributes.
- Physical feature—a product building block that consists of a set of causally related physical phenomena and mechanical elements (Kiriyama et al. 1992a) that are described by entities and relations among entities.

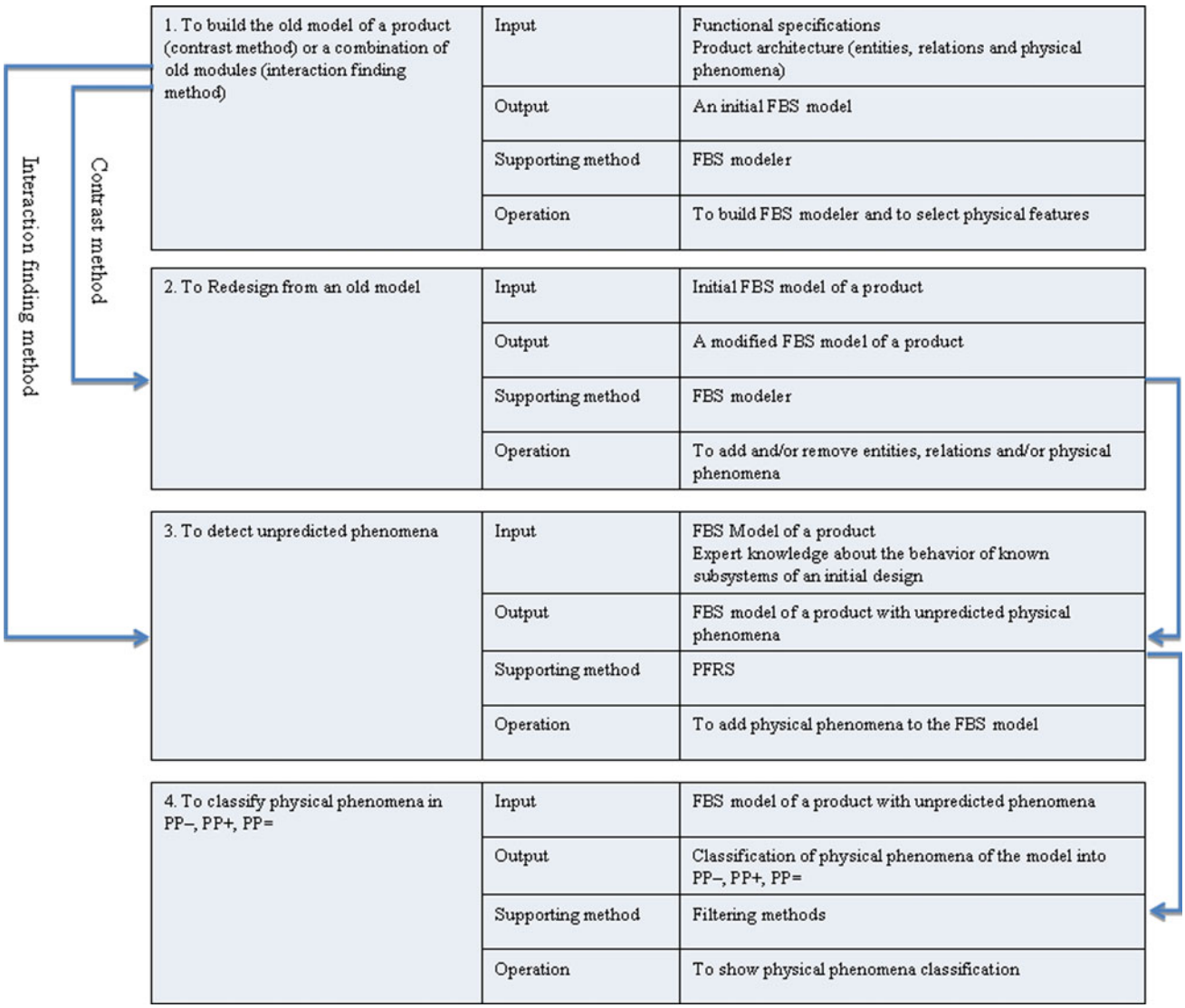


Fig. 2 High-level computational process to classify physical phenomena

- Function—a representation of behavior through human recognition, which is expressed in terms of *what the device aims to do*. Function constitutes a bridge between human intention and physical behavior of a system (Umeda and Tomiyama 1995; Erden et al. 2008).
- State—a set of qualitative values of system parameters (Kuipers 1994).
- Behavior (state transition)—a set of qualitative states that the system visits over time (Kuipers 1994).

2.2 The function behavior state model

The Function Behavior State (FBS) model of a product can be built by combining the low-level information of functions with the behavioral information described in physical features (Umeda et al. 1996). The FBS model is intuitive to use while designing provides a systematic way of structuring knowledge and its compatible with qualitative reasoning that is used as reasoning engine for this work.

A screen display of an FBS model on KIEF is shown in Fig. 3. While the details of the figure will be explained in the following sections, it is now sufficient to notice how the knowledge is divided among functions, which forms the function layer, and physical phenomena, entities and relations, which belong to the behavior layer of the FBS model.

2.3 Qualitative reasoning and qualitative process theory

Qualitative physics is a branch of Artificial Intelligence that qualitatively deals with reasoning about the behavior of physical systems (Barr et al. 1989). Qualitative physics

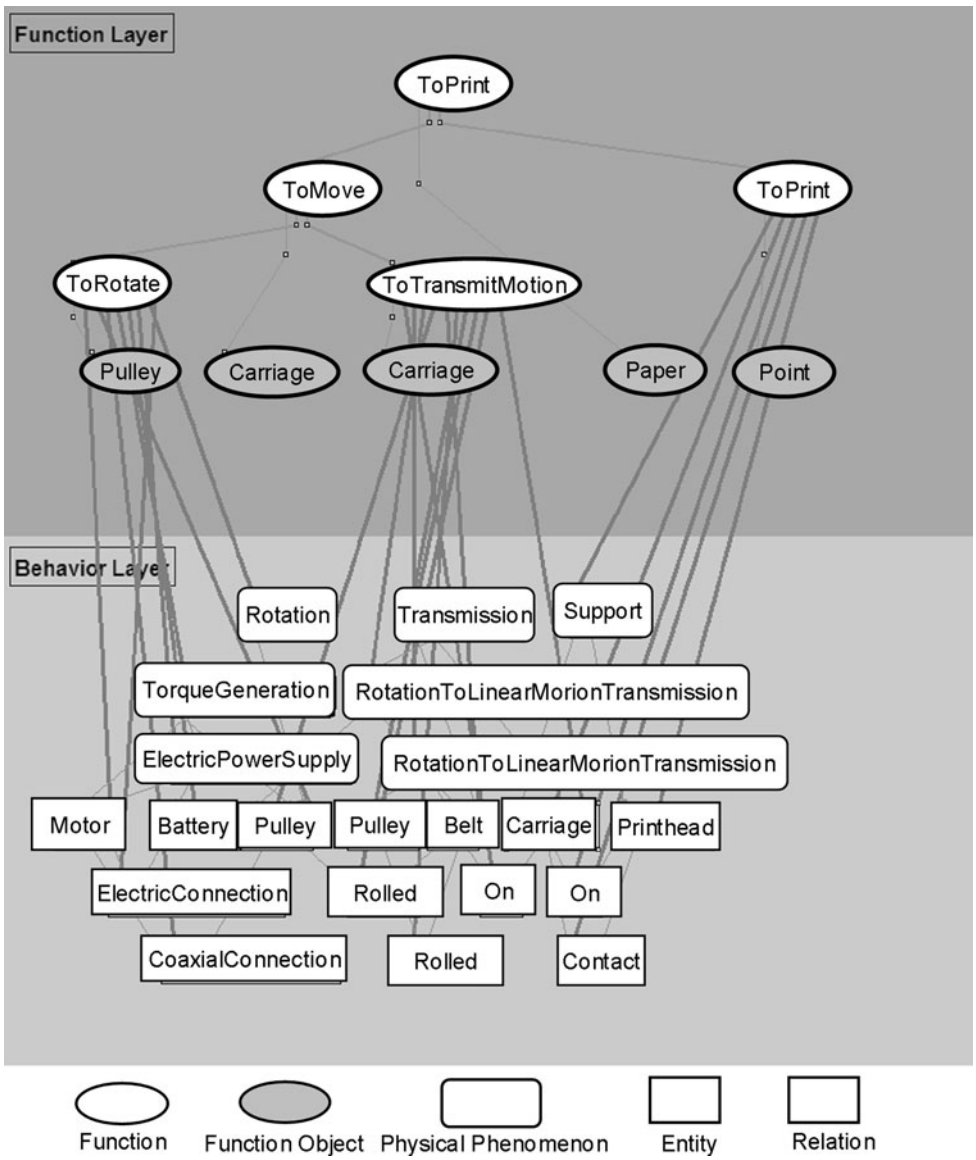


Fig. 3 FBS model in KIEF

employs qualitative reasoning to infer knowledge about a system. This is useful to describe roughly what will happen to a system. Due to the nature of knowledge required to perform qualitative reasoning, qualitative physics can mostly be used in situations where only incomplete knowledge is available including conceptual design.

Among qualitative physics techniques, DID uses QPT developed by Ken Forbus (1984) to understand commonsense reasoning about physical processes. QPT organizes domain theories around the concept of physical phenomena (processes) (Forbus 1984). Processes are the sole mechanisms of change in a system (Forbus 1993) and are the focus of our analysis. A physical situation is modeled as collections of objects (entities), their relationships (relations), and processes (physical phenomena). There are four operations that QPT can perform:

1. Given a physical situation, to decide which instances of processes can exist in that situation.
2. To determine which process instances are active by examining whether conditions are satisfied.
3. To determine which change can be caused by the active processes.
4. To predict behavior over time.

2.4 Physical feature reasoning system

Physical Feature Reasoning System (PFRS) is a simplified version of QPT and uses solely architectural knowledge to derive physical phenomena that may occur to the design object by matching patterns (prerequisites) against physical features in the knowledge base (Yoshioka et al. 2004;

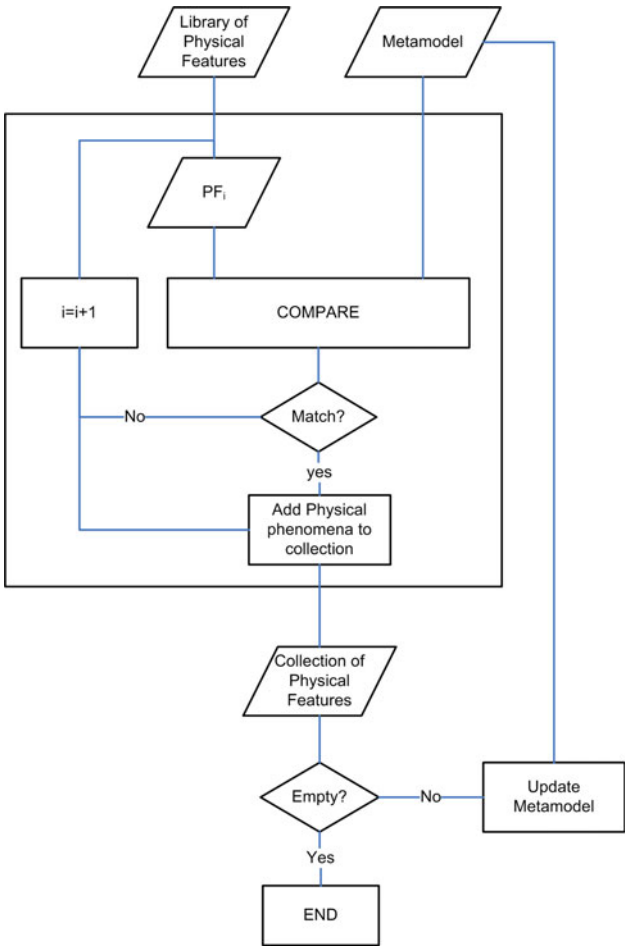


Fig. 4 PFRS algorithm

Kiriyama et al. 1992b). The pattern matching is performed by a pattern-directed inference system (Forbus and De Kleer 1993).

A simplified representation of the PFRS algorithm is shown in Fig. 4. The input data of the algorithm consist of the product description and the library of physical features. Figure 5 shows examples of four physical features. In the figure, thick borders indicate *consequence*, while the thin ones indicate *prerequisites*. '*Consequence consists of physical phenomena that are invoked by the physical feature'. 'Prerequisites are conditions about entities and relations, and physical phenomena that are used to check their conditions. They are needed to see if a physical feature happens or not*' (Yoshioka et al. 2004). A prerequisite can consist of entities, relations and physical phenomena, while a consequence consists of physical phenomena only. For example, the second physical feature *Heat Generation On Source* in Fig. 5 indicates that every time electric power (prerequisite) is applied to a conductive entity (prerequisite), heat is generated (consequence). In this example, a physical phenomenon (heat generation) needs another physical phenomenon (electric power supply) to be invoked. The electric power supply phenomenon represents a *causal link* for heat generation.

The algorithm starts by selecting one physical feature (PF) from the library (see Fig. 4). PFRS needs to scan all the physical features to verify possible occurrence of a new phenomenon. After that, the prerequisite of the PF$_i$ in Fig. 5 is compared with the model (Compare). If no match between the prerequisite and the model is found, then a next PF is selected from the library. When there is a match, one or more physical phenomena (consequences—thick border) are added to a *collection of physical phenomena*. These steps are repeated until all the PFs from the library have been compared with the model. At the end of the comparison, the collection of physical phenomena is added to the model. A new comparison is made between the updated model and the library. This process is repeated until there are no more physical phenomena found (the collection of physical phenomena is 'empty'). At this point, the model is completely updated.

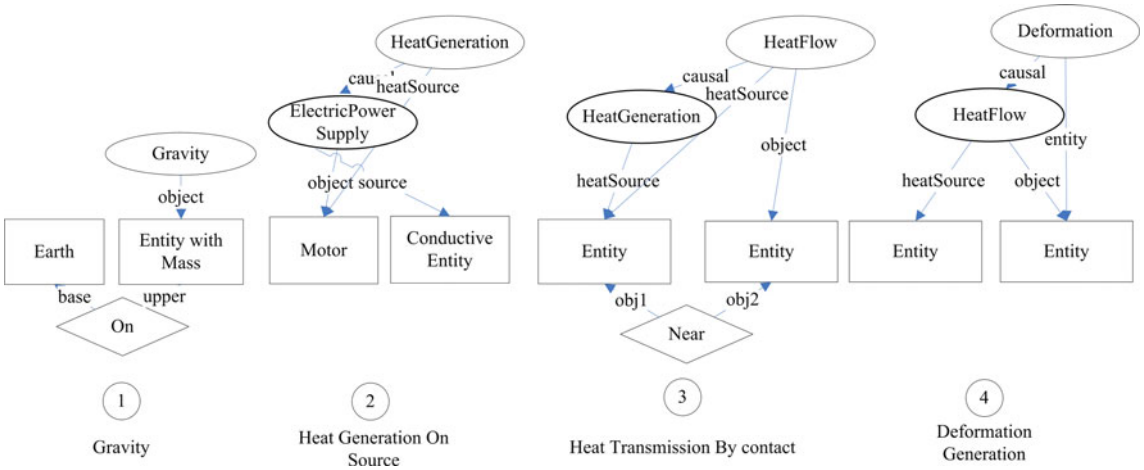


Fig. 5 Four examples of physical features

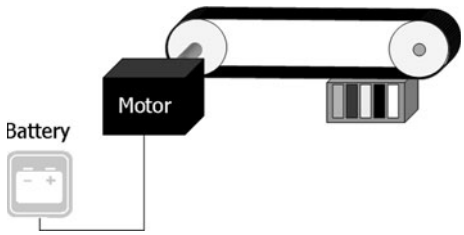


Fig. 6 Concise schematic representation of an inkjet printer

To reduce the number of iterations during the comparisons of the updated model with the library, the list of PFs in the library is reduced only to the ones that contain at least one causal link to physical phenomena. In this way, the PFs without causal link do not influence subsequent comparisons. Some physical phenomena depend only on structural knowledge (physical feature 1 in Fig. 5), while others depend on structural (entities and relations) and behavioral knowledge (physical phenomena) (physical feature 2, 3, 4 in Fig. 5). PFRS does not change the structural knowledge but it does change behavioral knowledge. Therefore, after the fist scan of all physical features, only physical features that can be influenced by the new behavioral knowledge are scanned again. These are the ones with causal links. For instance, the physical feature `gravity` of Fig. 5 after the first scan of PFRS, where the physical phenomenon gravity can be added to the model, does not have any other possibility to include new knowledge into the system. The other physical features indicated in Fig. 5 contain causal links between physical phenomena. For instance, the occurrence of `ElectricPowerSupply` in the physical feature `Heat Transmission By Contact` generates in the first scan `HeatGeneration`, in the second scan `HeatFlow` and in the third scan, `Deformation`.

To make pattern matching and search faster, PFRS uses the assumption-based truth maintenance system (ATMS) algorithm. ATMS is a problem solving facility to help inference engines conveniently and efficiently manipulate assumptions (De Kleer 1986).

2.4.1 An example of pattern matching in PFRS

The aim of this section is to explain how the matching mechanism works. To do this, we use a simple example. The case of an inkjet printer (see Fig. 6) has been chosen to demonstrate the way PFRS algorithm works. A print head is incorporated in a carriage, which moves forward and backward along a path indicated by a pulley-belt mechanism. The inkjet printer is driven by a DC motor, which is powered by a battery. The FBS model of this inkjet

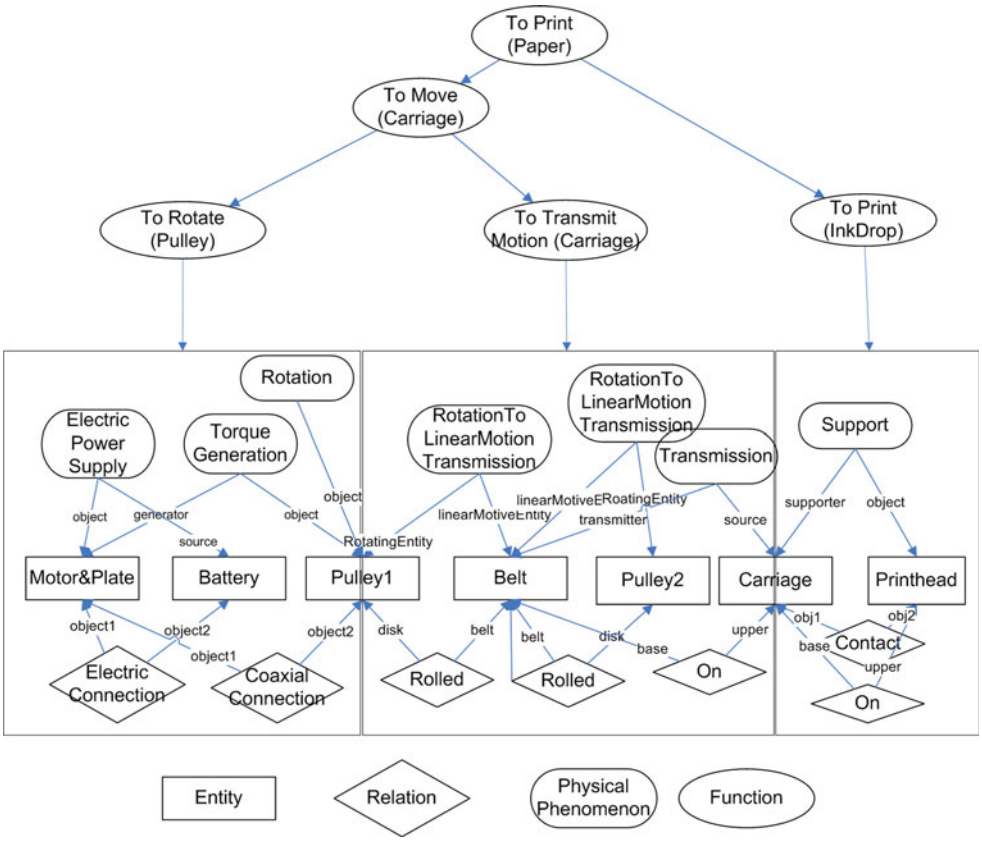


Fig. 7 Starting FBS model of an inkjet printer

printer is shown in Fig. 7 and consists of seven entities (*Motor&Plate, Battery, Pulley1, Belt, Pulley2, Carriage and Printhead*), seven relations (*ElectricConnection, CoaxialConnection, Rolled, Rolled, On, Contact* and *On*), and seven physical phenomena (*ElectricPowerSupply TorqueGeneration, Rotation, RotationToLinearMotionTranmission, Transmission* and *Support*). Three physical features related to three functions (*To Rotate(Pulley), To Transmit Motion to (Carriage), To Print(Ink Drop)*) of the printer are used to build the model. This model can be translated into the following assertions:

- *(ElectricConnection Motor&Plate (object1) Battery (object2)).*
- *(CoaxialConnection Motor&Plate (object1) Pulley (object2)).*
- *(Rolled Pulley (disk) Belt (belt)).*
- *(Rolled Belt (belt) Pulley (disk)).*
- *(On Belt (base) Carriage (upper)).*
- *(On Carriage (base) PrintHead (upper)).*
- *(Contact Carriage (obj1) PrintHead (obj2)).*
- *(ElectricPowerSupply Motor&Plate (object), Battery (source)).*
- *(TorqueGeneration Motor&Plate (generator), Pulley (object)).*
- *(Rotation Pulley (object)).*
- *(RotationToLinearMotionTrasmission Pulley1 (RotatingEntity), Belt (linearMotiveEntity)).*
- *(RotationToLinearMotionTrasmission Belt (linearMotiveEntity), Pulley2 (RotatingEntity)).*
- *(Transmission Belt (transmitter), Carriage (source)).*
- *(Support Carriage (supporter) PrintHead (object)).*

For example, the first assertion means that there is an electric connection between a motor and plate and a battery. Then, the physical features in Fig. 5 are considered. They can be transformed in the following set of *rules*:

- *(rule (On Earth (?base) EntityWithMass (?upper)). Assert! (Gravity (EntityWithMass (object)))).*
- *(rule (ElectricPowerSupply Motor (?object) ConductiveEntity(?source). Assert! (HeatGeneration (Motor (heatSource)))).*
- *(rule (Near Entity (?obj1) Entity (?obj2)) (HeatGeneration (Entity (?heatSource)). Assert! (HeatFlow Entity (heatSource) Entity (object))).*
- *(rule ((HeatFlow (Entity (? heatSource)) (Entity (?object))). Assert! (Deformation (Entity (entity)))).*

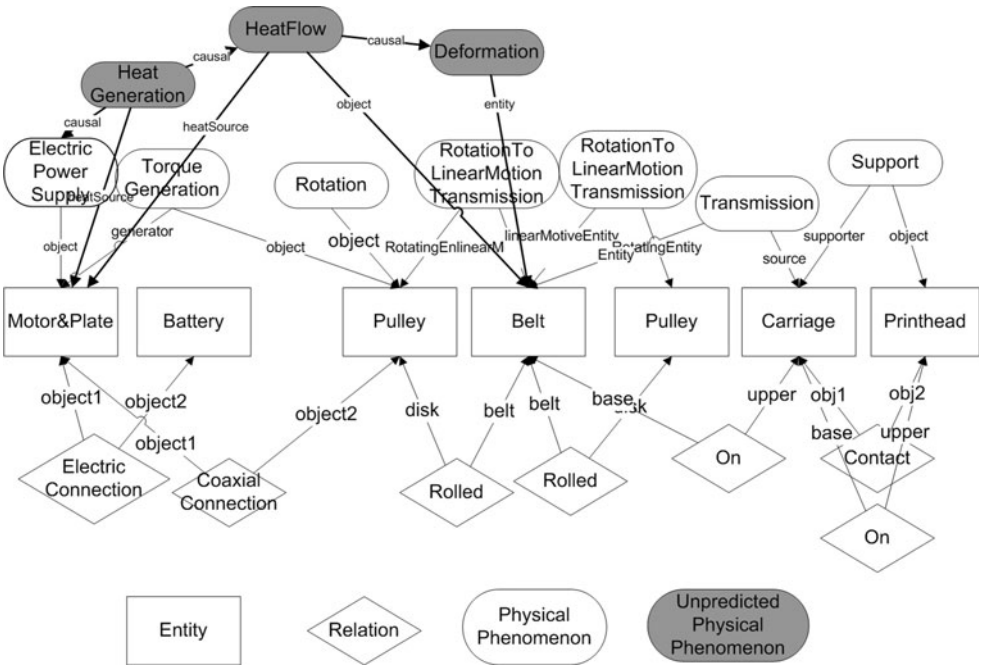
For example, the first rule means that when an entity with mass on the Earth, gravity force applies to this entity. Assertions and rules in PFRS are made based on class indexing. This means that assertions and rules are indexed with their classes, corresponding to sets whose elements can match. Therefore, a new assertion needs to be tested only against rules indexed under its class and the other way around a new rule needs to be tested against assertions in its own class.

For instance, the prerequisites of the second rule *((ElectricPowerSupply Motor(?object) ConductiveEntity(?source))* match the eighth assertion *(ElectricPowerSupply Motor&Plate (object), Battery (source)).*

Motor is included in *Motor&Plate*, and *ConductiveEntity* is a superclass of the entity *Battery*. Therefore, the variable *Motor(?object)* is bound to *Motor&Plate (object)* and *ConductiveEntity (?source)* is bound to *Battery (source)* and the consequence of the rule is executed. This means that *HeatGeneration* becomes also part of the assertions and *Motor(heatSource)* is translated into *Motor&Plate (heatSource).*

The inkjet printer model is one input to the PFRS algorithm. The other input is the library, which includes the physical features presented in Fig. 5. The result of the pattern matching between rules and assertions performed by the PFRS algorithm is shown in Fig. 8. The figure does not include the functional description of the model since functions do not contribute in inferring new physical phenomena. The new physical phenomena are indicated by oval shapes with gray background. Figure 8 shows that the motor of the printer can generate heat, which might result in deformation of the belt. By extending the database to more than the four physical features of Fig. 5, it is possible to infer other phenomena that are not mentioned in this example. The database must include general physics knowledge as well as a collection of knowledge obtained through previous design failures. By using this collection of knowledge about design failures, the designer can avoid to make the same mistakes again.

Figure 9 illustrates examples of failure knowledge expressed as physical features. The first physical feature represents a situation when wear of a bearing generates noise in the system. The second and third physical features express that friction and electrostatic charges are generated between these two entities whether a carriage moves on the



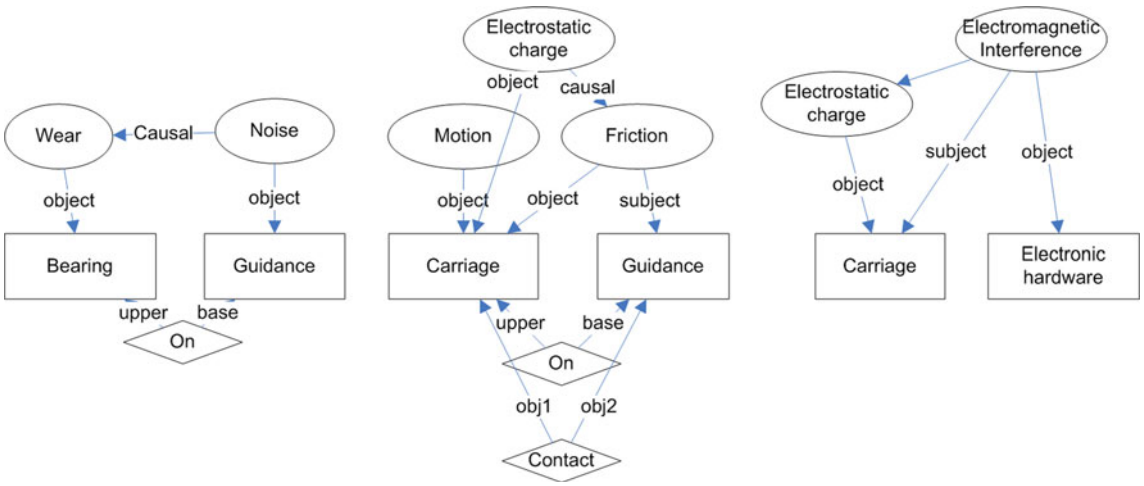


Fig. 8 Inferred model of the inkjet printer

Fig. 9 Example of failures knowledge expressed by physical features

guidance. Electrostatic charge causes electromagnetic interference that can affect the electronics in the system. Therefore, as soon as a new problem is encountered, the designer must document the new knowledge by including that in the physical features knowledge base.

2.5 Similar past research

The goal of this study is to develop methods into a software tool to detect and to classify physical phenomena while preserving all the relevant information necessary for redesign tasks. Before going to the filtering methods, this section shows similar past research that deals with conceptual design and redesign issues.

Case-based reasoning is a general paradigm of problem solving that recalls and reuses previous design experiences that can help with new situations (Maher and Gómez de Silva Garza 1997). DID makes use of heuristics obtained through past design experiences to support designers in problem solving. In this sense, DID developed aspects of case-based design in order to generally support the representation of design cases. For instance, DID is based on the FBS model (Umeda et al. 1996) (see Sect. 2.2) that represents design cases based on function decomposition.

Function decomposition allows the designer to reuse knowledge to combine subfunctions in order to create a design case. The same knowledge used to create a design case is used by PFRS to predict unpredicted phenomena and to find possible failures of the design (unpredicted problems) (see Sect. 2.4). This means that the same database is used both to create a design case and to verify the design. The filtering methods use expert knowledge to discard phenomena that can theoretically appear in a design case but practically do not appear (known by experience) (see Sect. 3). The filtering methods also use expert knowledge at the system level to integrate known hierarchical parts or subcases of a design. Therefore, case-based reasoning is relevant to this research, and it is used both for design, redesign, and verification of the design.

Goel analyzed the task of design adaptation and redesign. Adaptive design is described as a method where a design is adapted to another to develop new functionalities (Goel and Stroulia 1996). Redesign is an essential task to recover a system from design failures (undesired phenomena) (Goel and Chandrasekaran 1989). Failures are detected by diagnosis in design adaptation and in redesign (Goel and Stroulia 1996). For instance, this is done by detecting in the existing device components that affect the desired new functionality. Failures can also be detected in a realistic environment where undesired behavior (unpredicted phenomena) is discovered. Also in this situation, the goal is to detect the structural elements in the structure, which, if adapted, could generate the desired behavior. Once causes of failures are detected, the designer changes the adapted design. When this design fails again. new adaptations (redesign) are made until the design accomplishes the desired functionalities. Therefore, Goel analyzed device behavior and finds unpredicted and undesirable behaviors, and repairs the device from the failures. The repair subtask of redesign takes as input the desired functions, the proposed structure, the undesirable behaviors and their structural causes; it produces as output a modified structure that accomplish the desired functions without the undesirable behaviors. However, the verification of the design in Goel's case is performed on physical prototypes, and redesign is not intended for innovation purpose but for failure recovery. Furthermore, diagnosis can be costly to compute, and a better organization of the device model is needed so that only the relevant information is examined. Therefore, it is still necessary to have other verification techniques to detect undesired behaviors early in the design and a filter to constrain information to the relevant ones. However, once those possible design failures are detected, Goel's methods can be used to redesign with corrective or compensatory solutions (Goel and Chandrasekaran 1989).

Change management (revision control) manages changes in information. For example, Clarkson (Clarkson et al. 2004) discusses mathematical models to predict the risk of change propagation in terms of likelihood and impact of change. At an abstract level, the filtering methods are also methods for change management in design engineering due to the fact that they can track changes from an original source. However, differently on DID, change management does not use change information to detect design failures cased by those changes.

Kitamura tries to capture intentions of the designer by building a functional ontology. He intended to make the search in the functional hierarchies traceable for *functional understanding task* (Kitamura et al. 2002). For instance, he provides an operational method for bringing a gap between the behavioral and structural levels and the functional level, which is useful for redesign. However, redesign consists of removing and adding new functional structure in the system, and this task can lead on one hand to new unpredicted phenomena that are not present in previous designs and on the other hand to disabling intended functionalities in the new design. His method does not use qualitative physics-based reasoning that detects such behavioral changes.

Davis makes a system diagnosis reasoning based on structure and behavior (Davis 1984). The purpose of his analysis is to troubleshoot and diagnose complex systems when a symptom of malfunction is given. To do so, all the different kinds of paths of interactions among components need to be identified and handled. However, including all the possible paths, candidate generation again becomes indiscriminate since every component could somehow be responsible for the observed symptom. On the other hand, neglecting some paths can make the troubleshooting unreliable. The technique that Davis used for solving this dilemma is based on categorization of failures, which are organized from the most likely to less likely categories and it is based on experienced knowledge about failures. However, this strategy is applied to a static system where any phenomenon is known, controlled, and established; it does not consider the occurrence of unpredicted phenomena that enter into the system to change and to increase paths of interactions. Moreover, Davis works with diagnosis from symptom-fault rules rather than model-based diagnosis from design description.

Falkenhainer and Forbus proposed compositional modeling techniques for organizing domain models in order to determine which subset of knowledge to apply for a given task and, therefore, to filter behaviors relevant to a task (Falkenhainer 1991). However, using the compositional modeling techniques brings the risk of excluding useful knowledge to predict side effects.

Liem discussed in (Liem et al. 2008) an approach toward an automated model algorithm that uses causality to explain the system's behavior. In their algorithm, they use

clusters and causal information to reduce the space of possible behaviors. This approach is useful to eliminate redundancies in the system but cannot eliminate negligible behaviors that are the focus of this paper.

3 Methods to classify and filter unpredicted physical phenomena

3.1 Classification of physical phenomena

The core of filtering methods introduced in this study (see Sect. 3.2) is based on the selection and classification of physical phenomena. The following classification of physical phenomena will be used in the rest of the paper (D'Amelio and Tomiyama 2007):

- Desired phenomena—intended phenomena that the designer wants to realize with the design (see Sect. 3.2).
- Undesired Phenomena—phenomena that disturb the product behavior, in other words, side effects.
- Predicted phenomena—desired phenomena, which are predicted by the designer. Predicted phenomena are known effects (i.e. known from previous designs). In the redesign case, where a product changes from an old to a new version, predicted phenomena can be part of PP= or PP+ classes depending on whether they belong to both product versions or only to the new design (see Sects. 3.2 and 3.3).
- Unpredicted phenomena—These are unexpected phenomena, which can be either desired or undesired, depending on whether they include additional desired functionalities to the design or whether they disturb the behavior of the system (see Sect. 3.2). Unpredicted phenomena are part of PP+.
- Negligible phenomena—Physical phenomena that are insignificant in a product. These phenomena belong to the PP—class.

The combination of those four types of phenomena is aggregated in the following manner:

- Predicted problems—undesired and predicted behaviors that can appear in a product. The designer is aware of these problems, and (s)he must control their intensity during embodiment and detailed design.
- Unpredicted problems (destructive phenomena)—undesired and unpredictable interactions that result in undesired and unpredictable behaviors. The designer is not aware of these problems during design but they can appear at the prototype phase. They are added to the design model by using the PFRS.
- Constructive phenomena—desired and predictable phenomena that result in desired and predictable behaviors. These are the result of design decisions.
- Forgotten phenomena—desired and unpredicted phenomena. For instance, the designer may overlook to include a phenomenon in the product model. They are added automatically to the design model by using PFRS.

The above definitions are summarized in Table 1.

KIEF is able to infer unpredicted phenomena starting from predicted phenomena. However, it is not possible for KIEF to distinguish between desired and undesired phenomena because this distinction depends on the design context and on the intention of the designer. Moreover, the distinction between desired and undesired phenomena goes against the 'No function in structure' principle, which asserts that 'the laws of the parts of the device may not presume the functioning of the whole' (De Kleer and Brown 1984).

3.2 Filtering physical phenomena

The goal of the filtering methods is to identify changes in the system behavior due to changes in system architecture. Thus, it is necessary to make another classification of physical phenomena, viz. PP−, PP+, and PP=.

PP− class comprises the list of negligible physical phenomena. $PP-_{design}$ are PP– instances that are removed from the model by the designer during the phase of redesign (see Process 1 in Fig. 2). $PP-_{design}$ instances appear when the designer removes an entity from an old design dragging out also its interconnected physical phenomena. $PP-_{causal}$ are PP− instances that are automatically filtered out by PFRS (see Process 2 in Fig. 2).

$$PP- \equiv PP-_{design} \cup PP-_{causal}$$

Furthermore, $PP-_{design}$ belongs to the old model and not to the final model. $PP-_{causal}$ belongs neither to the old model nor to the final model. $PP-_{causal}$ are potential unpredicted phenomena because patterns of their physical features match the new design but the filtering method discards them by using heuristics information of the old design. The next sections show examples of $PP-_{design}$ and $PP-_{causal}$.

Table 1 Physical phenomena classification

Phenomena classification	Predicted phenomena PP= or $PP+_{design}$	Unpredicted phenomena $PP+_{causal}$	Negligible phenomena PP−
Desired phenomena	Constructive phenomena	Forgotten phenomena	$PP-_{design}$ (See Sect. 3.2)
Undesired phenomena	Predicted problems	Unpredicted problems	$PP-_{causal}$ (See Sect. 3.2)

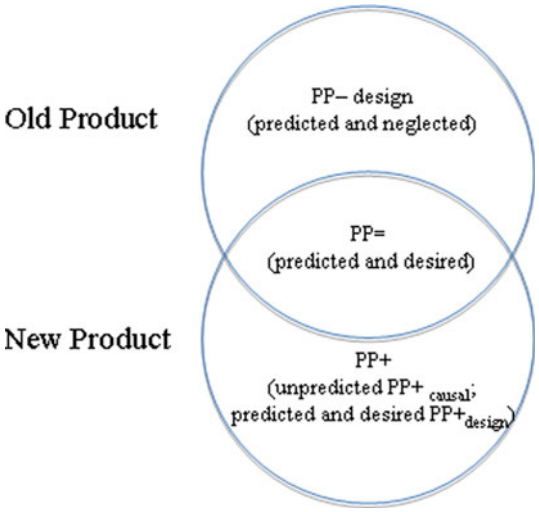


Fig. 10 Comparison scheme between old and new product

PP+ is a list of physical phenomena added by the designer during the redesign phase (PP+ $_{design}$) (see Process 1in Fig. 2) or automatically by the PFRS (PP+ $_{causal}$) during the phase of verification (see Process 2 in Fig. 2). Therefore:

$$PP+ \equiv PP +_{design} \cup PP+_{causal}$$

PP+ can represent both unpredicted and predicted phenomena that are not included in the old design. Therefore, the designer knows a number of phenomena associated with new components but he/she can still be surprised by unpredicted phenomena associated with these new components. Thus, PP+ can belong only to the new design, and it is used to keep track of changes, to understand consequences of design changes and interactions among machine building blocks (physical features).

The PP= class is used as a checklist to determine whether all desired physical phenomena are present in the final design.

The difference between old and new design in terms of PP−, PP+, and PP= is shown in Fig. 10. Both PP+$_{causal}$ and PP−$_{design}$ can cause anomalous system behaviors. The first because unpredicted phenomena can generate unpredicted problems, and the second because the designer could have erroneously removed some intended phenomena while redesigning without caring to put them back in the new model.

Therefore, due to the potential problems that PP+$_{causal}$ and PP−$_{design}$ may cause, the designer should further investigate their effects. For instance, additional quantitative tests should be performed at the later design phases in order to understand when these effects are negligible or not (i.e. embodiment and detailed design). The advantage is that when the designer knows potential problems, he/she can keep them under control in more detailed phases.

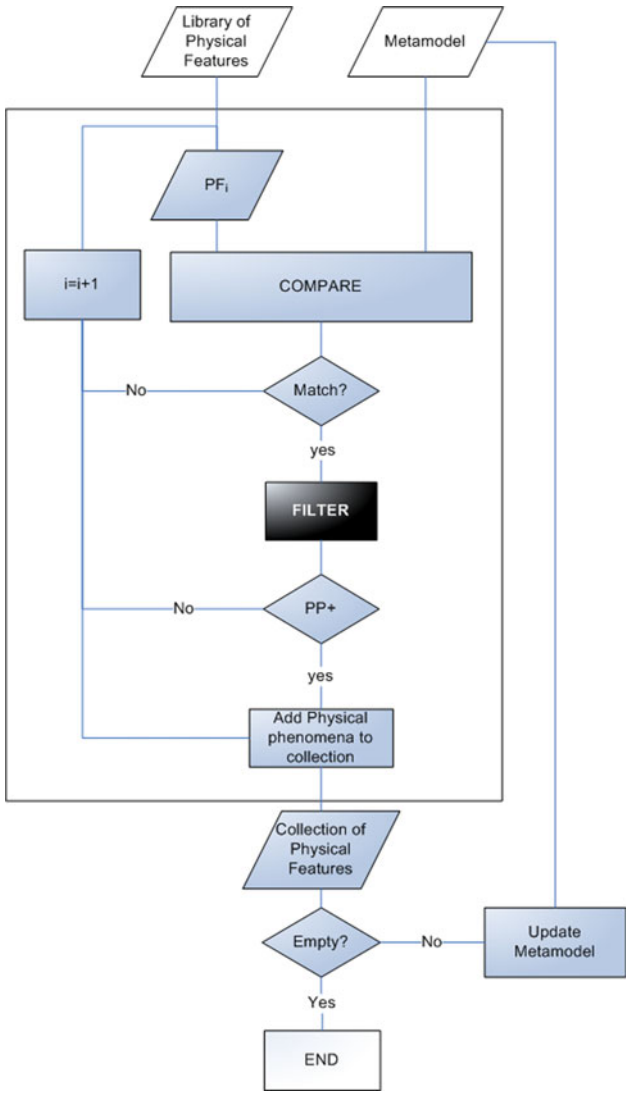


Fig. 11 Filtering module in PFRS Algorithm. Trapezoidal shape—input and output; rectangular shape—process; diamond shape—decision

For the purpose of distinguishing between PP−, PP+, and PP=, two filtering methods have been developed and implemented in KIEF by introducing a FILTER block in the PFRS algorithm (see Fig. 11). The two filtering methods are the contrast and the interaction finding methods. These two methods can be used separately or together during design tasks from an old product (with known behavior) to a new product (with uncertain behavior), and during the system phase of the design.

3.3 The contrast method

The contrast method is used in redesign, when concepts (entities, relations, or physical phenomena) are removed

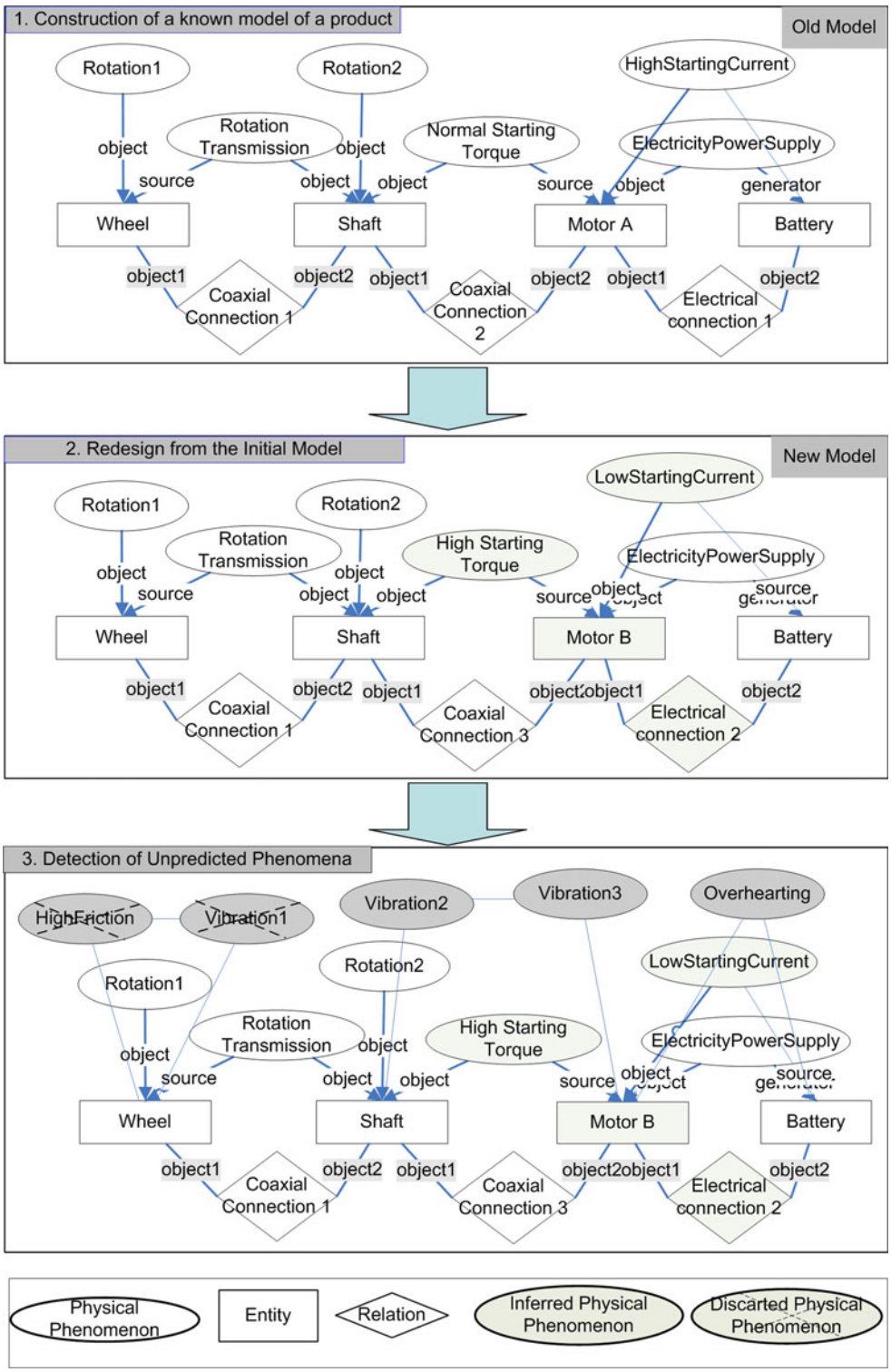


Fig. 12 Process 1: Old model of a product. Process 2: new model of a product. Process 3: new model of a product after using PFRS with the contrast filtering method

from or added to an old model. The goal of this method is to identify all the PP+ and PP− instances resulting from those changes. The contrast method assigns to the PP+ class new phenomena that are added to the system by the designer and the unpredicted physical phenomena, in which at least one prerequisite belongs to new components. The unpredicted physical phenomena, in which *all* the prerequisites are old concepts, are assigned to PP−. The PP= class contains physical phenomena that are predicted in the model.

Table 2 Classification of system concepts

Component−	Component=	Component+	PP−	PP=	PP+
Motor A	Wheel	Motor B	Normal starting torque (Shaft; MotorA) *PP−design*	Rotation (Wheel)	High starting torque (Shaft; MotorB) *PP+design*
Coaxial connection2 (Shaft; MotorA)	Shaft	Coaxial connection3 (Shaft; MotorB)	SupplyingElectricPower (MotorA; Battery) *PP−design*	Rotation transmission (Wheel; Shaft)	SupplyingElectricPower (MotorB; Battery) *PP+design*
Electrical connection1 (MotorA; Battery)	Battery	Electrical connection2 (MotorB; Battery)	High starting current (MotorA; Battery) *PP−design*	Rotation (Shaft)	Low starting current (MotorB; Battery) *PP+design*
	Coaxial connection1 (Wheel; Shaft)		Friction (Wheel; Shaft) *PP−causal*		Overheating (MotorB; Battery) *PP+causal*
			Vibration (Wheel) *PP−causal*		Vibration (MotorB) *PP+causal*
					Vibration (Shaft) *PP+causal*

An example of all the processes of Fig. 2 is shown in Fig. 12. Table 2 shows the result of the contrast filtering method for this example. The old model of an abstract product is shown in Process 1 in Fig. 12, and this model was known. For the sake of simplicity, the functional layer of the FBS model is not shown.

A new product (Process 2 in Fig. 12) is the result of the redesign task based on the old model. The new model is obtained by removing an entity (`Motor A`) from the old model and adding another entity (`Motor B`) together with two relations (`ElectricalConnection 2` and `CoaxialConnection 3`). In this example, `MotorA` and `MotorB` refer to two motors with different characteristics. `MotorA` supports the phenomenon `NormalStartingTorque` and a `HighStartingCurrent`, and `MotorB` supports the phenomenon `HighStartingTorque` and `LowStartingCurrent`. Motors with different characteristics generate different behaviors.

Removal of `Motor A` results automatically in the removal of a series of elements: `Electrical connection1` between `MotorA` and `Battery`, the `CoaxialConnection2` between `Shaft` and `MotorA`, and the `NormalStartingTorque` that applies to `Shaft` and `MotorA` , `Electric Power Supply`, and `HighStartingCurrent` (see Fig. 12). As a consequence, the phenomena `NormalStartingTorque`, `SupplyingElectricPower`, and `HighStartingCurrent` become negligible and are added to the PP− class. These phenomena are labeled $PP-_{design}$.

Additional physical phenomena are added to the model by the designer: `HighStartingTorque`, `SupplyingElectricPower`, and `LowStartingCurrent` (see Fig. 12). These phenomena belong to PP+.

Some phenomena exist both in the old model (Process 1, Fig. 12) and the modified model (Process 2, Fig. 12): `Rotation(Wheel)`, `Rotation(Shaft)`[1] and `RotationalTransmission(Wheel; Shaft)` (Table 2). These phenomena are automatically labeled class PP=.

In Fig. 12, Process 3 shows the result of applying PFRS together with the contrast filtering method on the new product. Three new physical phenomena (indicated with the white background) are inferred by PFRS and included in the PP+ list. The new phenomena result from the matching of the model in Process 2 of Fig. 12 with the physical features described in Fig. 13.

The complete classification of the components and physical phenomena is listed in Table 2. By definition, the behavior of an old design or of its components is entirely predicted (known). If this old design or some of its components are included in a new model, the parts are still entirely known. Unpredicted physical phenomena cannot be attach to the known components of the new design and are automatically discarded and recognized as negligible by the filter.

In the example, since the prerequisites of `Vibration (Wheel)` (prerequisite: `RotatingEntity (Wheel)`) and `Friction (Wheel; Shaft)` (prerequisites: `RotatingEntity(Wheel)` and `RotatingEntity(Shaft)`) also belong to the old model, they are automatically discarded by the filter of PFRS.

The prerequisite of `Vibration(MotorB)` is `MotorB`, which is a new node of the model. Therefore,

[1] *Rotation (Wheel)*, and *Rotation (Shaft)* represent the same phenomenon applied to different entities (respectively to *Wheel* and *Shaft)*.

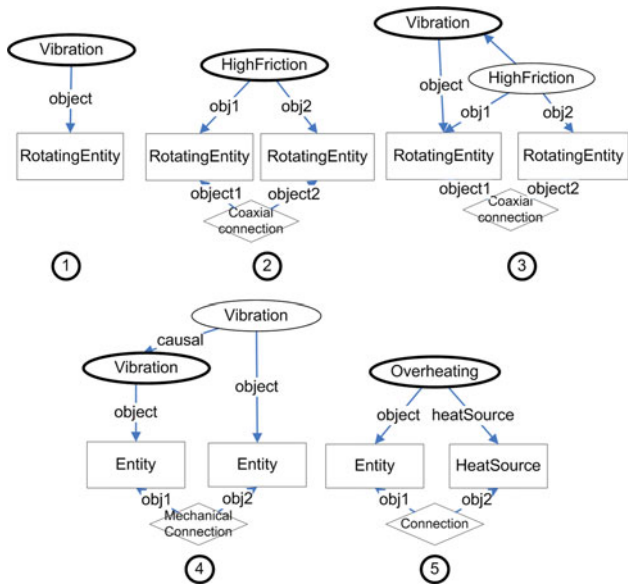


Fig. 13 Features used to derive the physical phenomena of Fig. 12

Vibration(MotorB) is an unpredicted phenomenon ($PP+_{causal}$). The prerequisites of *Vibration(Shaft)* are *Shaft* and *Vibration(MotorB)*, where *Vibration(MotorB)* is a new node of the model as well. Therefore, also *Vibration(Shaft)* is an unpredicted phenomenon that belongs to the class $PP+_{causal}$. *Vibration(Wheel)*, *Vibration(MotorB)* and *Vibration(Shaft)* represent the same phenomenon applied to different entities. *Vibration(MotorB)* and *Vibration(Shaft)* are causally connected.

3.4 The interaction finding method

The interaction finding method is used when the designer builds a new model by combining known subsystems in an innovative way, and this method can also be used together with the contrast method in the hybrid case of a design made of known subsystems and new components. Known subsystems refer to well-known pieces of technology (e.g. pulley mechanism, cooling systems, and generator). The goal of this method is to identify all the PP+ and PP− instances that emerge from the interactions of subsystems that we represent as physical features. In other words, the question here boils down to: Even though we combine known subsystems, can there something strange happen? The unpredicted physical phenomena that have prerequisites belonging to more than one building block are assigned by PFRS to PP+. The unpredicted physical phenomena that have *all* the prerequisites belonging to one building block are assigned by PFRS to PP−. The PP= instances are the *desired* physical phenomena, in which the prerequisites belong to one building block.

An example of the interaction finding method is shown in Fig. 14. Three different subsystems represented in different engineering domains (software, electronics and mechanics) are shown in the figures. The case shows how in the top engine of an inkjet printer, interactions among software, electronics, and mechanics occur. In Fig. 14, *CarriageElectronics* decodes all information to pass to the *PrintHead* (colors and dots); *FlexCables* provides the power acquired by the engine electronics to the carriage; *Carriage* supports and sets up the position of the print head on the paper. *SettingAccelerationPoints*, *Overpressure*, *Overheat*, *Heat Flow*, *Heat Generation*, and *IllDotsPositioning* represent the unpredicted interactions of various subsystems. PFRS infers phenomena by matching the present library of physical features with the current model and selecting the interactions of the software, electronics, and mechanics domains. By following the causal relations represented in Fig. 14, *Overheat* can be easily identified as the cause of *IllDotsPositioning* on the paper. Moreover, interactions of different blocks can be related to engineering domains that were absent in the original design (i.e. the thermal domain).

The interaction finding method is helpful to take decision at the system design level since all the knowledge coming from several subsystems has to be considered and integrated. This is done by inferring consequences of integrating knowledge from different engineering domains and/or different subsystems.

The complete classification of components and physical phenomena for the case of Fig. 14 is listed in Table 3. In the interaction finding method, the PP− are physical phenomena that are discarded by the filter in PFRS (i.e. *DataLoss*, *PowerLoss*, and *Deflecting*). Therefore, these are unpredicted physical phenomena, in which all the prerequisites belong to a single subsystem.

3.5 Case study

To show practical outcomes of PFRS combined with the two filtering methods, the inkjet printer of Fig. 7 is considered again. With the following cases, we highlight the advantages in using the two filtering methods together with PFRS in comparison with the use of PFRS alone.

Case 1 The model of Fig. 7 is considered new, and no filtering method is applied. In this case, physical phenomena are classified as in Fig. 15. None of the inferred physical phenomena is filtered out by the system. As a result, the designer must inspect a list of 39 PP+ instances to detect possible unpredicted problems.

Case 2 The model of Fig. 7 is considered as a combination of old modules, and hence, the interaction finding method is applied. This means that the physical features

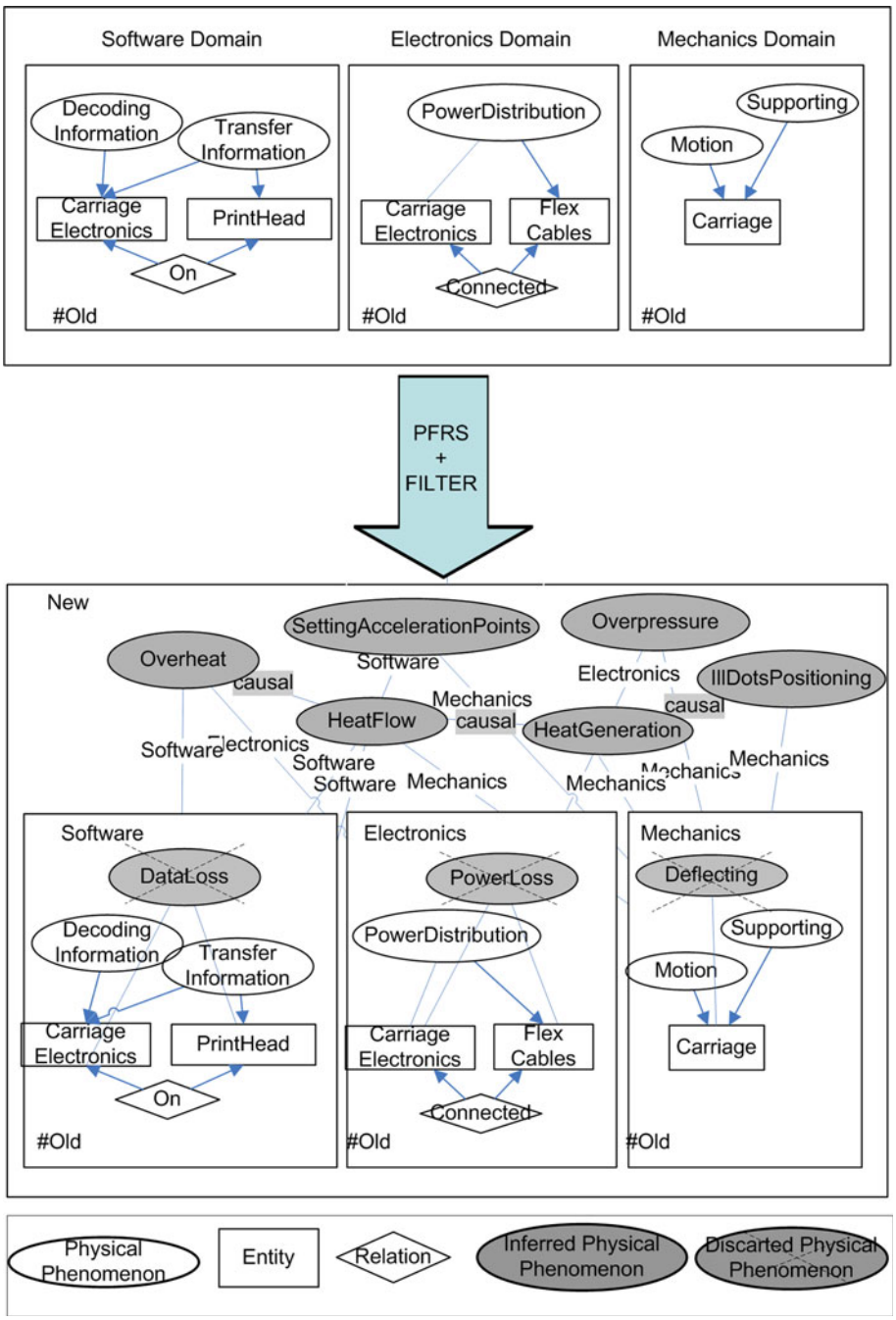


Fig. 14 Example of use of the interaction finding method in multi-disciplinary domains

Table 3 Classification of system concepts with the interaction finding method

Component−	Component=	Component+	PP−	PP=	PP+
	Carriage electronics		DataLoss	Decoding information	SettingAcceleration points
	PrintHead		PowerLoss	Transfer information	Overheat
	FlexCables		Deflecting	Power distribution	HeatFlow
	Carriage			Motion	HeatGeneration
	On			Supporting	IllDots positioning
	Connected				Overpressure

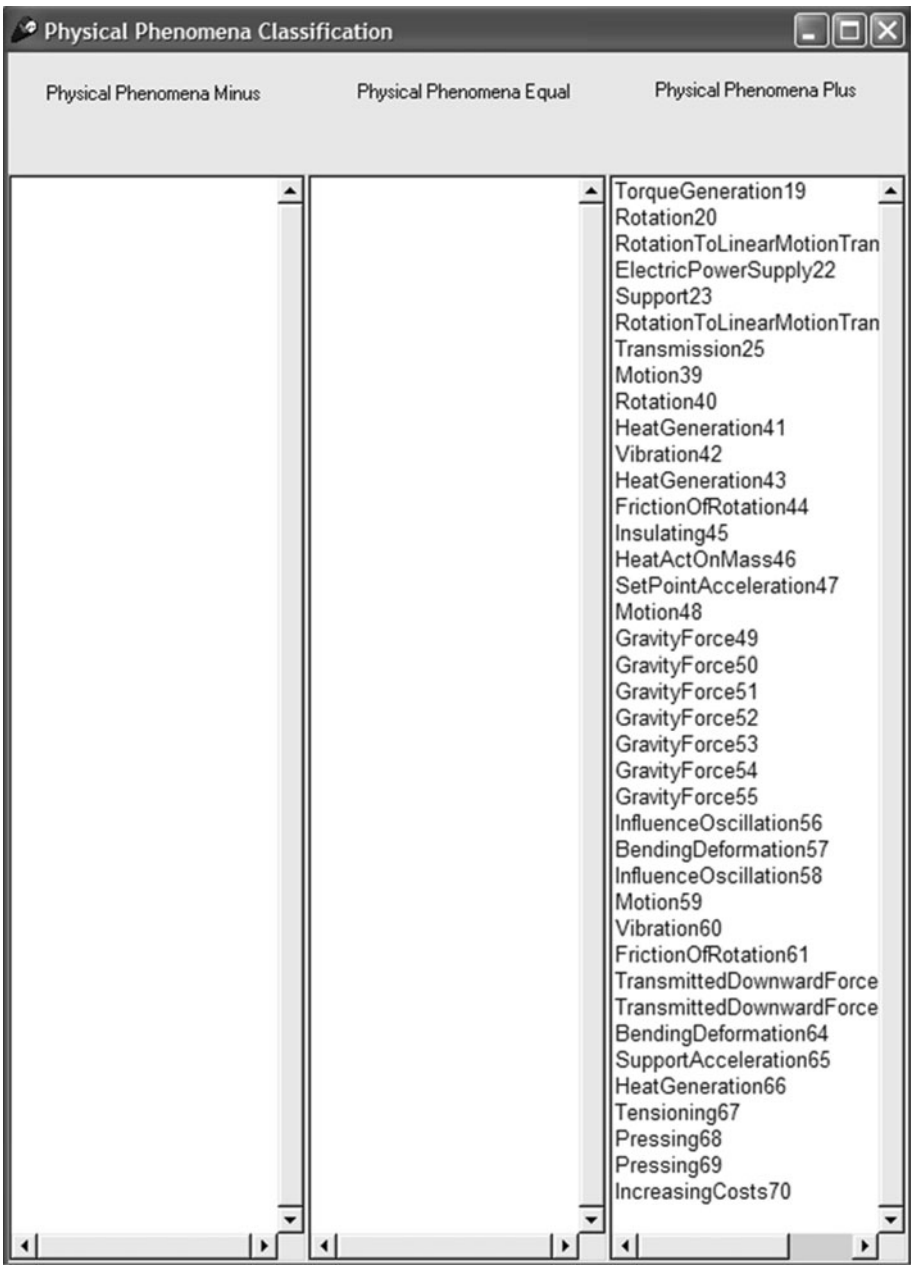


Fig. 15 Classification of phenomena without the use of any filter: new product

that constitute the model are considered as independent modules with unknown interactions. Four PP+ instances are shown in Fig. 16 and represent interactions of different modules.

Case 3 Starting from the inkjet printer model in Fig. 17, the contrast methods are applied. This model refers to an inkjet printer plugged to an electric source, and the designer knows this design (old model). Imagine that the designer wants now to adjust the design to make the printer portable. To do so, the designer substitutes the wall-plugged motor with one connected to a battery. This transformation is made by removing module A from the model represented in Fig. 17 and by including module A' represented in Fig. 18. From these transformations, we derive the model of Fig. 18, whose component A' is totally new, and whose components B and C belong to the old model (Fig. 17) as well. In this last case, the classification of phenomena in PP−, PP=, and PP+ classes after running PFRS is presented in Fig. 19. The PP+ class consists of new phenomena. However, this case does not show unpredicted *problems* among the unpredicted phenomena. These unpredicted phenomena are forgotten phenomena that do not affect the functionalities of the design. Therefore, the designer has verified the robustness of the design

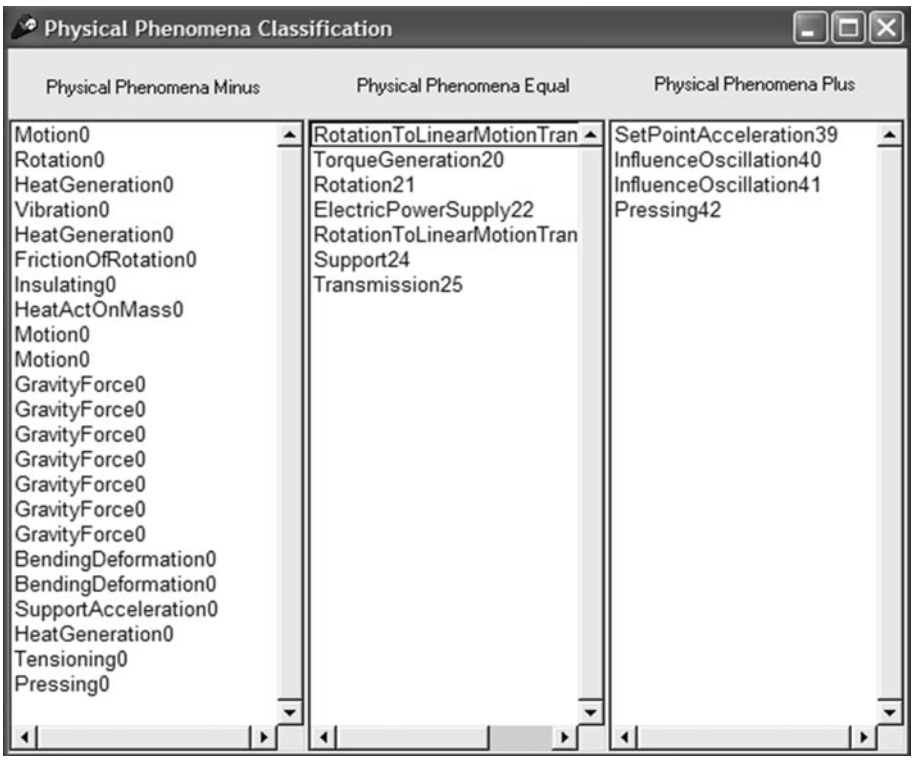


Fig. 16 Classification of phenomena with the use interaction finding method: old product

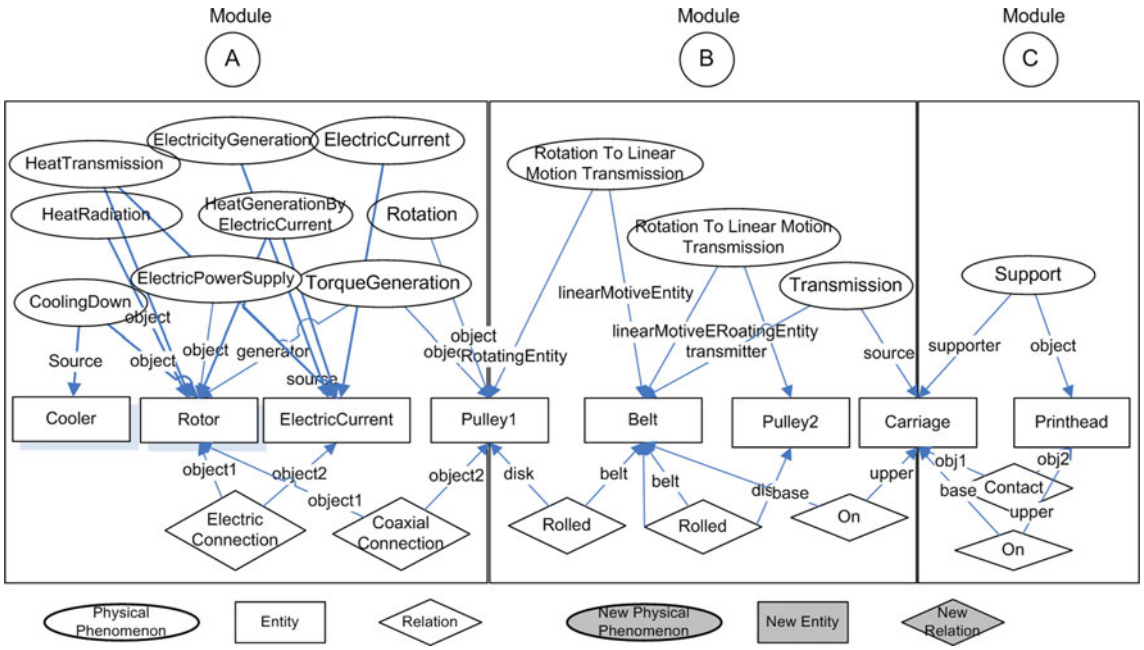


Fig. 17 Old model of a plugged inkjet printer

by inspecting unpredicted phenomena and can continue to the embodiment phase. Practically speaking, PFRS has inferred that the replacement of a plugged motor with a battery does not affect the functionalities of the inkjet printer.

PP$-_{\text{design}}$ instances (recognizable because starting with a number in PP− list) require special attention from the designer, since these phenomena were desired phenomena in the old product, but do not appear in the new product. It is necessary to check whether the change was intended by the designer or not.

It is also possible to apply both the methods simultaneously. This is the case of a design made by both known and unknown subsystems. The system can infer physical

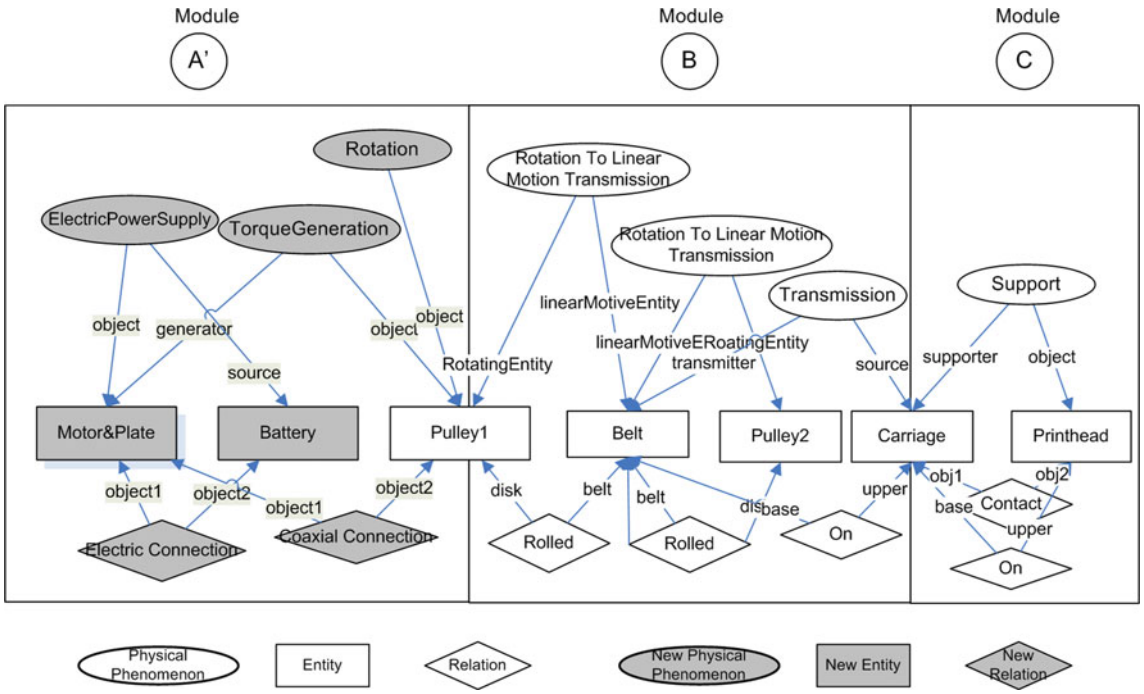
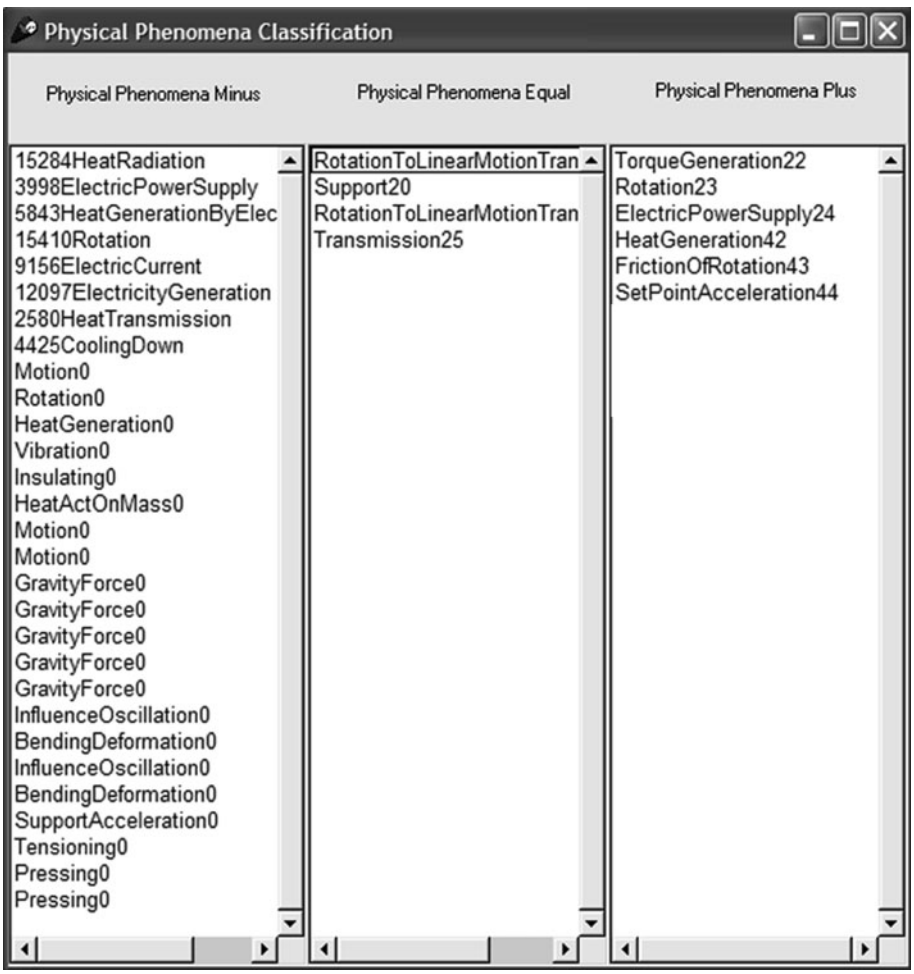


Fig. 18 New model of an inkjet printer

Fig. 19 Classification of phenomena with the use of contrast method



phenomena belonging to the singular unknown subsystems but also interactions between known subsystems.

4 Discussions

PFRS together with the filtering methods is helpful to verify the design against unpredicted phenomena and to organize the inferred information in classes. Furthermore, the filtering methods of DID can reduce the number of unpredicted physical phenomena in the qualitative inferred results. The case studies showed that unpredicted phenomena can decrease from 39 to 4 or 10 depending on the adopted filtering method. Among the inferred physical phenomena, there are some that can generate unpredicted problems in the design, for example, unexpected heat generation, friction of rotation, and oscillations when not controlled can deteriorate the system performances.

More work is needed to generate an adaptation plan to recover from unpredicted problems. As we mentioned in Sect. 2.5, the designer can use corrective or compensatory solutions to recover a system from failures (Goel and Chandrasekaran 1989). The FBS modeler of KIEF can support the use of corrective and compensatory solutions based on the functional description. For instance, a function can be associated with more than one working principle (for corrective tasks), and the designer can choose a particular change to perform from these alternatives. Therefore, it is possible to change working principles in redesign tasks. Furthermore, additional functions can be plugged into FBS to support compensatory solutions. However, these alternative solutions need to be analyzed again to verify their compensations and/or drawbacks. This analysis, to detect the best solution, can be difficult due to the amount of redesign possibilities. In order to avoid this time-consuming analysis, two solutions can be possible. One is the implementation of a synthesis algorithm to automatically find the best corrective or compensatory solutions. The other solution could be to equip DID with standard corrective and compensatory solutions to automatically recall and reuse previous design experiences to recover from failures. Although DID is not equipped with such tools to recover from failures, we suggest that to recover from erroneously discarded phenomena ($PP-_{design}$, due to the removal of entities), it is easier to use compensatory solutions because it can be necessary to integrate additional structure to the system in order to have more behaviors. However, compensatory solutions increase the system complexity due to the larger amount of components. Therefore, to recover from destructive phenomena, we recommend the use of corrective solutions.

A limitation of this work is in the knowledge base. Expert knowledge (e.g., design history, previous failures, and results of tests performed on products) and physical principles (physical laws and effects) are presented as physical features. Since human knowledge is huge, the number of physical features in the library is also huge. According to Lenat 1995, human knowledge is close to 10^8. This number refers to the number of axioms that are derivable by spanning human commonsense knowledge. Knowledge is represented by physical features that encompass both structural and behavioral knowledge. One problem in having a large amount of knowledge is that the higher the number of physical features is, the more computational effort has to be made by PFRS to match the library of physical features to the model. Moreover, a huge library can lead to the interference of phenomena of the relevant scope (i.e. anything can happen).

Knowledge on single entities (i.e. shaft, pulley, and motor) as well as knowledge about relations (i.e. coaxial connection, electrical connection) is universal knowledge that can be used and reused to build any physical system from scratch. This knowledge does not vary from library to library. Physical features consist of integration of knowledge that can be unique for specific applications. Integrating different entities in different manners makes a device behave differently. This knowledge is 'specific to a range of applications'. It can still happen that knowledge of various physical features is used for different devices because of similarity in functions. However, at least one function must differ between the devices to make them unique. The library of physical features also contains knowledge related about physical principles. Since physical principles are universal and their amount is computationally manageable (around 300 in physics domains (Hix and Alley 1958)), the part of the library that consists of physical principles provides a reusable and complete knowledge for any application.

In general, it is difficult to define a perfect size of the library because it depends very much on the application and on the physical system model to analyze. The library of KIEF in its current version includes around 300 physical features that represent physical effects and 320 physical principles. The library encompasses physical principles as well as specific knowledge about printer mechanisms and a collection of knowledge about failures that were exhibited by previous versions of the same type of inkjet printer. By using this collection of knowledge about failures, the designer can avoid to make same mistakes again. Examples of this knowledge have been illustrated in Sect. 2.4.

5 Conclusions

This paper discussed difficulties that are associated with mechatronic products from the viewpoint of design. To

solve these difficulties, the contrast filtering method and the interaction finding filtering method were introduced. Those methods are implemented in the PFRS of KIEF.

Filtering methods are based on the definition of predicted and unpredicted, desired and undesired as well as negligible physical phenomena. Starting from these definitions, physical phenomena were divided into three main classes: PP−, PP= and PP+. PP− instances represent discarded phenomena, PP= instances are predicted phenomena that appear both in the new and old design, PP+ are phenomena belonging only to a new design and they encompass unpredicted phenomena. The filtering methods are able to keep track of changes during the process of designing and/or redesigning, to understand the consequences of product evolution and product module integration. Then, in the PP+ class, the designer can detect design failures that warn of possible unwanted behaviors of the product. At this point, the design can still be reconsidered without constructing prototypes.

Two filtering methods were described, the contrast and the interaction finding methods. The designer can use them separately or together, in design tasks that assume the evolution of a product from configuration *A* to configuration *B* or the analysis of interactions among different design modules. The contrast method categorizes physical phenomena based on design modifications that constitute differences between the old and the new design. By assuming a product made by clearly defined and understood modules, the interaction finding method is capable to detect interactions of design modules. A case study (Sect. 3.5) has shown the profitable use of the Design Interference Detector (DID) for different design tasks: design from the scratch, redesign from an old to a new model, and interaction finding. In Case 1, filtering methods were not used that correspond to a situation where the designer does not have any expert knowledge to filter out phenomena (design from scratch). In Cases 2 and 3, the filters were relevant to reduce the list of attention points for the designer. The efficiency of the filters depends on the available knowledge about the system; the more knowledge, the less unpredicted phenomena. Consequently, the filtering methods reduce the large number of possible solutions generated by qualitative reasoning to a smaller number, which is easier to verify for a single designer.

In conclusion, filtering methods using heuristics exploit all the advantages of qualitative reasoning and reduce the difficulty related to the generation of too many negligible qualitative solutions. This result is relevant for the conceptual design because it allows qualitative reasoning, which is fast, does not require complete product knowledge, and can be used for early design verification. Therefore, PFRS with filtering methods constitutes a qualitative tool that is ready to be tested in industry to help in conceptual design tasks.

Acknowledgments The authors gratefully acknowledge the support of: the Dutch Innovation Oriented Research Program 'Integrated Product Creation and Realization (IOP-IPCR)' and the BSIK program of the Dutch Ministry of Economic Affairs, Agriculture and Innovation, to the project "Smart Synthesis Tools"; the Embedded Systems Institute (Eindhoven, The Netherlands) to the projects "Smart Synthesis Tools" and "OCTOPUS"; and to Océ-Technologies B.V. to the "Smart Synthesis Tools" and "Octopus" project.

References

Barr H, Cohen A, Paul R, Feigenbaum, Edward A (1989) The handbook of artificial intelligence. vol 4**,** Artificial intelligence

Clarkson PJ, Simons CS, Eckert CM (2004) Predicting change propagation in complex design ASME. J Mechani Des 126(5):788–797

D'Amelio V, Tomiyama T (2007) Predicting the unpredictable problems in mechatronics design, in: conference proceedings of the 16th international conference on engineering design –design for society, cité des sciences et de L'Industrie, Paris, August 28–30, The design society, Paper number 153, 10 pages, (CD-ROM)

Davis R (1984) Diagnostic reasoning based on structure and behavior. Artif Intell 24(1–3):347–410

De Kleer J (1986) An assumption based TMS. Artif Intell 28(1):6

De Kleer J, Brown JS (1984) A qualitative physics based on confluences. Artif Intell 24:7–83

Erden MS, Komoto H, Van Beek TJ, D'Amelio V, Echavarria E, Tomiyama T (2008) A review of function modeling: approaches and applications, Artificial intelligence for engineering design. Anal Manuf 22:147–169. doi:10.1017/S0890060408000103

Falkenhainer B, Forbus KD (1991) Compositional Modeling: finding the right model for the right job. Artif Intell 51:95–143

Forbus KD (1984) Qualitative process theory. Artif Intell 24:85–168

Forbus KD, De Kleer J (1993) Building Problem Solvers. Artif Intell, Cap. 12

Goel AK, Chandrasekaran B (1989) Functional representation of designs and redesign problem solving. In: Proceeedings of the 11th international joint conference artificial intelligence (IJCAI-89), San Matteo, CA: 1388–1389

Goel AK, Stroulia E (1996) Functional device models and model-based diagnosis in adaptive design. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Special Issue on Functional Representation and Reasoning 10:355–370

Hix CF, Alley RP (1958) Physical laws and effects. John Wiley &Sons, London

Kiriyama T, Tomiyama T, Yoshikawa H (1992a) Building a physical feature database for qualitative modeling and reasoning. In: Joskowicz L, Williams B, Cagan J, Dean T (eds) Working Notes of the Symposium: Design from Physical Principles, AAAI Fall Symposium Series, October 23–25, Cambridge. MA, USA, pp 125–130

Kiriyama T, Tomiyama T, Yoshikawa H (1992b) Qualitative reasoning in conceptual design with physical features. In: Faltings B, Struss P (eds) Recent advances in qualitative physics. The MIT Press, Cambridge, pp 375–386

Kitamura Y, Sano T, Namba K, Mizoguchi R (2002) A functional concept ontology and its application to automatic identification of functional structures. Adv Eng Inform 16(2):145–163

Kuipers B (1994) Qualitative reasoning/modeling and simulation with incomplete knowledge, p. cm.—Artif Intell

Lenat DB (1995) CYC: a large-scale investment in knowledge infrastructure. Communications of the ACM Volume 38, Issue 11 ISSN:0001-0782, pp 33–38

Liem J, Buisman H and Bredeweg B (2008) Supporting conceptual knowledge capture through automatic modelling, second international workshop on the induction of process models (IPM2008), pp 37–44

Maher ML, Gómez de Silva Garza A (1997) Case-based reasoning in design. IEEE Intell Syst 12(2):34–41

Nomaguchi Y, Tomiyama T (2002) Design knowledge management based on the model of synthesis, In: Proceedings of the fifth IFIP WG5.2 workshop on knowledge intensive CAD 62–81

Stevens R, Brook P, Jackson K, Anrnold S (1998) System engineering: coping with complexity. Prentice hall, London, ISBN 0-13-095085

Tomiyama T, D'Amelio V, Urbanic J, and ElMaraghy W (2007) Complexity of multi-disciplinary design, cirp annals-manufacturing technology, vol 56, no 1, ISSN 0007-8506, doi: 10.1016/j.cirp.2007.05.044, pp 89–92

Umeda Y, Tomiyama T (1995) FBS modeling scheme of function for conceptual design. In: Proceedings of working papers of the 9th international workshop on qualitative reasoning about physical systems, pp 271–278

Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T (1996) Supporting conceptual design based on the function-behavior-state modeler, AI for engineering design. Anal Manuf 4(10):275–288

Will P (1991) Simulation and modelling in early conceptual design: an industrial perspective. Res Eng Des 3(1):1–14

Yoshioka M (2008) Knowledge Intensive Engineering Framework: KIEF (formerly known as SYSFUND) Manual, http://www-kb.ist.hokudai.ac.jp/∼yoshioka/KIEF/manual.pdf, accessed in Oct 2010

Yoshioka M, Umeda Y, Takeda H, Shimomura Y, Nomaguchi Y, Tomiyama T (2004) Physical concept ontology for the knowledge intensive engineering framework. Adv Eng Inform 18:95–113