Design Interference Detector-A Tool for Predicting Intrinsic Design Failures

V. D'Amelio, T. Tomiyama
Department of BioMechanical Engineering, Faculty of Mechanical, Maritime and Materials Engineering
Delft University Technology, Delft, the Netherlands
v.damelio@tudelft.nl

Abstract
Intricate failures at the system integration phase of the design are mostly generated by interactions of product modules or/and engineering domains. Modules of the product are tested independently and when their integration is performed, design failures are detected. This significantly delays the machine development. This paper aims at introducing a software tool able to predict phenomena which generate destructive couplings of engineering domains and product modules. An example shows how these couplings influence the system behaviors. The analysis is conducted at the conceptual deign level and makes use of qualitative data to reason out behaviors of the product.

Keywords:
Mechatronics, Unpredicted problems, Qualitative Physics, Verification, Integration

1 INTRODUCTION

Market is nowadays invaded by very complex machines. A product is complex when it has to fulfill a large amount of functionalities. Current mobile phones for instance are not anymore just devices to transmit and receive sound but may support many additional services and accessories, such as SMS for text messaging, email, packet switching for access to the Internet, and MMS for sending and receiving photos and video. Printers are generally scanners and copiers together; they can deal with many paper formats and materials maintaining a high quality. Costs, environmental issues, energy consumptions and high performances play other important roles in the development of products. Numerous technologies, skills and competencies are then required to make a product competitive. Mechatronics solutions are adopted by designers because of their efficient way to combine machine functionalities. Suh defines complex a product in which functional requirements are coupled [1]. In this sense mechatronics solutions lead to increase product complexity and therefore to a complicated product development process (PDP). Even if Paul and Beitz [2] provide a systematic way to get from the conceptual to the detailed design to avoid as many design failures as possible, still years are required to deliver products. To have a better overview of the product, standard modularization methods are used as for instance Design Structure Matrix [3]. On one hand these methods allow a better subdivision of tasks among engineers; on the other hand they can generate a huge number of integration problems when the system is considered as a whole.

Modules of the machine are designed and tested independently of each other and only at the end of the PDP a total test is performed. Unpredicted problems are the result of the total test and they are difficult to troubleshoot and to solve. Unpredictable problems are unexpected behaviors or physical phenomena that occur within a domain or by interactions of domains. They are generated by destructive couplings of engineering domains which are undesired and unpredictable interactions.

A fault-tree analysis (FTA) [4] is not sufficient to understand causes of problems. Indeed, the machine architecture is also influenced by unpredictability given by unknown connections between components. Furthermore, FTA and FMEA (Failure Mode and Effect Analysis) deal with problems coming from deterioration of the system and faults which are consequences of anomalous circumstances, while unpredictable problems are generated intrinsically by the design of the system.

This article introduces a software tool, the Design Interferences Detector (DID) which is able to predict unpredicted problems of mechatronics solutions at the conceptual design level.

In order to understand capabilities of the DID, the article summarizes main phases of the PDP referring to well-known design methods and it specifies where the DID is located into the PDP. Then the article focuses on the architecture of the DID which is implemented as an extension of KIEF (Knowledge intensive engineer framework) [5]. An example shows how a total test of the machine can be roughly but efficiently performed by using a qualitative analysis. Then section 4 shows which behaviors are predicted when the DID is used for integrating sub-modules. Results are discussed in the conclusions together with limits and future work on this research.

2 PLACING THE DID INTO THE PRODUCT DEVELOPMENT PROCESS

The V-model represents the system development lifecycle and it was developed by Stevens in 1998 [6]. Figure 1 shows the main design phases to generate a product based on the V-diagram [7]. Although the V-model is mainly used for software creation, it is adopted also in product design. All the design and verification phases are mapped in figure 1.

The left side of the model describes decomposition of requirements and creation of system specifications. Suh in his Axiomatic Design [1] provides a methodology to valuate if functional requirements and specifications are adequate. He states that functions requirements must be decoupled or uncoupled. This method cannot deal with cases in which the complexity increases unexpectedly during the course of design due to undesired and unpredictable interactions among subsystems.

On the right branch of the model the integration and the verification of the system are performed. In integration testing the separate modules are tested together to expose faults in the interfaces and in the interaction between integrated components.

Many different verification and validation techniques are used to determine if the system fulfills specifications and if its output is correct for each prototype test. Among the other techniques it is important to mention the Functional testing, which consists of proving all the functions of the system which are defined in the requirements; the Structural Testing, which uses architectural information of a system to verify the operation of individual components; Random testing, which can detect peculiar faults; Fault injection in which system is observed while working under fault conditions; and Risk Analysis which identifies consequences of obstacles and possibility of occurring [8].

Going down on the left branch of the diagram each design level gives more functional and architectural details than the previous one. On the right side of the V-diagram (Figure 1) the system is tested at various levels, from components to subsystems and from subsystems to system level. Strong delay in the product development is given by failures which are found in this leg of the V-model. The design process of a complex product requires months while the verification process requires years.

A problem at the component level of the verification branch is easy to solve and to troubleshoot because it is about a sole system unit. Problems at the system level, where total test is performed, are not-trivial to troubleshoot and solve because they involve the entire system. Therefore, problems at the system level can lead to changes at the conceptual design of the product which is at the top level of the left diagram branch. The early design level and top verification level depend on each other but they are performed at the opposite extreme of the PDP. Years are in between the two development phases.

The distance between design level and correspondent verification level increases going up to the right branch of the diagram. It is risky to make mistakes at the conceptual design level because these mistakes will only appear at the end of the PDP. Failures at a high level of the prototype test correspond to flaws at an early level of design. Going backwards to an early design phase at the end of the PDP is obviously time consuming and cost inefficient. The ideal situation would be to have verification of the conceptual design as early as possible.

The DID wants to create the shortcut between early design level and high level verification as it is shown in figure 2. The objective is to recognize design failures given by integration of subsystems well in advance. This will save money in prototyping and increase the design quality by using alternative ideas instead of repairing design mistakes. Compensation of design mistakes turns out in adding pieces of software and hardware to the system to bring the system output to nominal values. This leads to changes at the conceptual design. By means of the DID the iterative trial-and-error process will be minimized and the best design among those proposed can be determined by analyzing several design solution at an early stage.

Even if the article emphasizes the application of the DID methodology at the system level, the software and the methodology can be used for subsystem and lower level subsystems test. At the conceptual design phase the system can be modeled and behaviors can be reasoned out just using rough information by means of Qualitative Physics (QP) [9]. This is what the next section is going to explain by introducing the DID architecture and methodology.

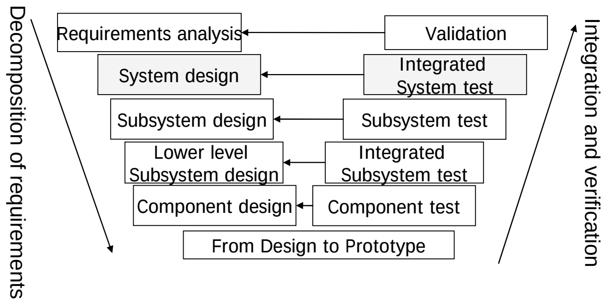

Figure 1: PDP without DID.

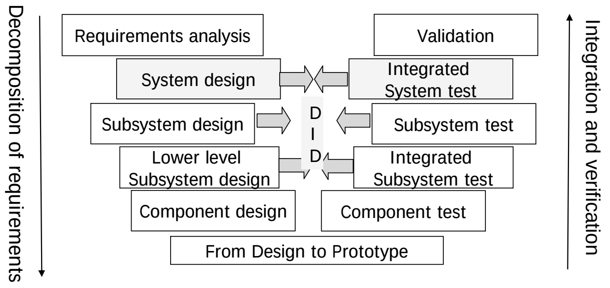

Figure 2: PDP with DID.

3 DID ARCHITECTURE AND METHODOLOGY

In order to reason system behaviors and unpredictable problems which can arise at a late phase of the design DID employs Function Behavior State (FBS) model as representational scheme [10] and QP based reasoning system [9] as the reasoning engine of the system.

The FBS model incorporates functional, behavioral and architectural information of the product. The architecture of the system consists of entities, which are physical components of the product, and of physical relations among entities, which denote the static structure of the product. The behavioral information consists of physical phenomena, which are physical laws or rules that govern behaviors, and of states of entities, which are values of parameters associated to the entities such as ‘rotational speed’ or ‘pressure’.

Qualitative Physics reasons qualitatively about the behavior of physical systems. This branch of artificial intelligence perfectly matches the FBS model in the way in which knowledge is structured. Qualitative Process Theory (QPT) is part of QP and it was developed by Ken Forbus at Massachusetts institute of technology [11]. The goal of QPT is to understand the commonsense reasoning about physical processes. Processes in QPT are the only source of change in physical situations. Examples of physical processes include ‘boiling’, ‘motion’, ‘acceleration’ and ‘rotational transmission’.

The idea of using QPT for prediction of design failures came out from two considerations: at the conceptual design level no precise details have been specified; in troubleshooting, monitoring and diagnosis there are no precise mathematical models of failure modes and humans operate with less detailed model [11]. Designers seldom use numbers but more approximations and rough values for the system state.

Steps to take toward the detection of unpredictable design problems are schematically represented in figure 3.

The first step consists in the construction of the primary FBS model. The model can be mono-disciplinary (Model 1D) or multidisciplinary model (Model nD), which represents mechatronics products. The qualitative engine reasoned out behaviors which can possibly happen to the design object including predicted and unpredicted ones. The behavior is represented in term of state transitions.

When the product is multidisciplinary it is necessary to integrate the engineering modules before reasoning behaviors. This is performed automatically by the DID that adds Interfaces between modules. These Interfaces are represented by extra physical phenomena and extra connections among components.

The amount of physical phenomena reasoned out by the qualitative engine can easily explode for a complex system such as mechatronics product. A filtering method will be used to constraint the amount of possible behaviors reasoned out by the software [12].

All the operational blocks which are necessary to generate system behaviors are resumed in figure 4 [13]. Physical features consist of entities, relations and physical phenomena [14]. Physical features can be thought as high level building blocks of the product. The designer selects physical features from the database and combines them into an FBS model. Attributes of entities are generated by a direct influence of physical phenomena and they are related to each other by the indirect influence of physical laws. Attributes are assigned to entities by physical phenomena. The physical rules create the network among attributes.

QPT engine combines all the information of the product. State transitions and causal connections among attributes are then automatically generated.

KIEF works as a knowledge base and reasoning system for the DID. It reasons out the possible physical phenomena occurred on the designed object by using the physical feature reasoning system (PFRS) facility. The PFRS is based on pattern matching technique. The integration of physical features (product building blocks) is performed by comparing the FBS model, which is created by the designer, and physical features, which are stored into the database. This step is necessary to predict unpredictable phenomena in the product design.

The DID connects modules of the product by introducing hidden relations among components and it integrates building blocks at the behavioral level (physical phenomena) and at the architectural level (physical relations) as well. This will be better clarified in the next section.

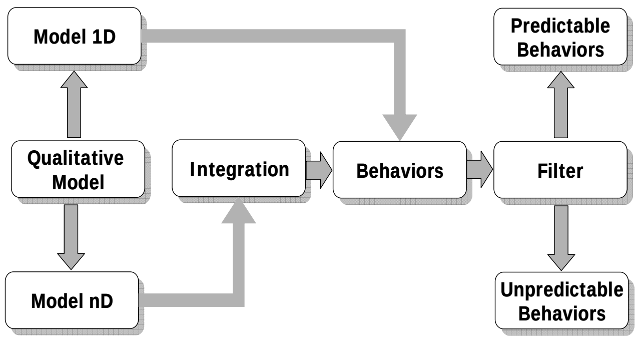

Figure 3: DID architecture.

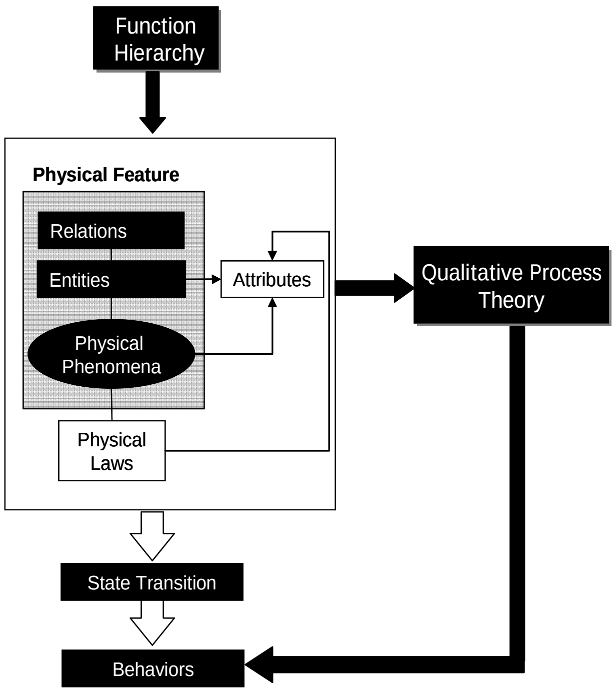

Figure 4: DID methodology.

3.1 The DID extension for KIEF

DID makes use of the PFRS and Qualitative reasoner of KIEF in order to reason out unpredicted physical phenomena which lead to unpredicted behaviors of the product.

The PFRS generates unexpected physical phenomena. The physical feature reasoning system has been modified in the DID to generate also unpredicted physical relations of components. Unpredicted relations can cause further unpredicted physical phenomena.

These connections are often implicit and the designer can omit their importance. An example can be found in the concept of ‘distance’. For instance the phenomenon ‘heat transfer’ connected to the source ‘heater’ will affect the destination ‘entity’ when their relation is ‘Near’. Relations represent a condition for the phenomenon to be generated. For activating the condition, the DID will ask to the designer to verify if the connection exists or not by generating unambiguous queries.

Especially when the system is very modular, it is likely for subsystems to be implicitly coupled. The DID keeps the attention of the designer on these unknown relations, which are important because they can activate further physical phenomena in the system. Next section shows unpredicted phenomena and unpredicted relations of product components by means of the example of the engine top of an inkjet printer.

4 EXAMPLE OF THE TOP ENGINE OF AN INKJET PRINTER

This section represents the FBS model and the behavior of the top engine of an inkjet printer which is used to accurately position the print-head, and consequently the ink, over the paper. Results are shown for the model built by the designer (primary model) as well as the model including unpredicted phenomena (reasoned model).

4.1 FBS representation of the primary model and of the reasoned model.

The engine top of an inkjet printer is the module that allows the ink to be accurately placed on the paper. The main function of this artifact is to print dots of ink. This function consists into shooting a stream of ink forcefully forth from a nozzle while print head moves along the guidance. The print head is supported by a carriage. A motor (generally stepper motor) moves the carriage back and forth across the paper. A belt is used to attach the carriage to the motor.

The first abstraction model of this product is taken as an example to clarify all the concepts which are mentioned in the previous chapter.

In figure 5 the top engine of the inkjet printer is translated into an FBS model. The oval shapes represent functions of the product. Functions are decomposed into sub functions until physical features can be linked to them. In the example the three physical features: ‘carriage print head system’, ‘carriage drive’ and ‘rotation by motor’ are correspondently associated to the functions: ‘To print drop of ink’, ‘to transmit motion to the carriage’ and ‘to rotate pulley’. The components of the model are a print-head, a carriage, which supports the print-head, and a carriage drive. The carriage drive contains a motor, a battery and a pulley mechanism. Connections of the design object are represented, for instance, by ‘coaxial connection’ between motor and pulley, and by the ‘electrical connection’ between motor and battery, which are described in the physical feature ‘carriage drive’.

Figure 5 shows the physical features which are associated to the functions. The physical features are in relation to each other. Indeed the same component can appear in more that one physical feature. It is task of the system engineer to delegate components which are the same in one unique frame. In figure 6 components which are the same are delegated. The primary model is represented by the white blocks. The unpredicted physical phenomena and connections among entities are highlighted in the same figure with gray boxes. White and gray blocks together constitute the total behavioral model of the system which is the reasoned model.

Eleven unpredicted physical phenomena and two extra-connections such as ‘Heat Generation’, ‘Deformation’, ‘Rotation’, ‘Heat Flow’, ‘Belt Transmission’, and ‘Near’ are the reasoned results from the primary model.

The reasoned connection ‘Near’ generates the unpredicted phenomenon ‘Melting’ in the FBS model and change the model topology and behavior. This connection is automatically generated by the KIEF extension and it is an additional result in KIEF physical feature reasoning.

4.1.1 A filter for physical phenomena

The reasoned model suggests physical phenomena which can happen. No information is given on the probability of phenomena to happen. Furthermore, complex product can lead to a large amount of unpredicted phenomena. For these two reasons, it is necessary to have a method to filter reasoned phenomena based on their probability to occur.

The limit analysis and physical phenomena causal network analysis are two of the studies used to filter phenomena out of the broad range of possible physical phenomena.

The limit analysis establishes the condition for a phenomenon to be generated. Limit analysis is a basic operation in prediction for figuring out what kind of things might happen next [15]. For instance the phenomenon ‘Deformation’ will be activated just when ‘battery temperature’ turns to the value ‘hot’. Qualitative values of attributes become crucial to identify when a phenomenon is activated or not. Whether the system does not turn to the value associated to the phenomenon, the phenomenon is erased by the list of possible phenomena.

Filtering can be performed also looking to the causal network of phenomena. Physical phenomena can be connected to each other by a causal link. A phenomenon ‘B’ can be caused by another phenomenon ‘A’ and generates a further phenomenon ‘C’. The chain of phenomena is in this case A$\rightarrow$B$\rightarrow$C. The phenomenon C is the one with less probability of appearance because it requires first the appearance of both A and B. Among the 3 phenomena A is the most probable to happen.

The next section shows a comparison between the initial model and the reasoned model in terms of reasoned behaviors.

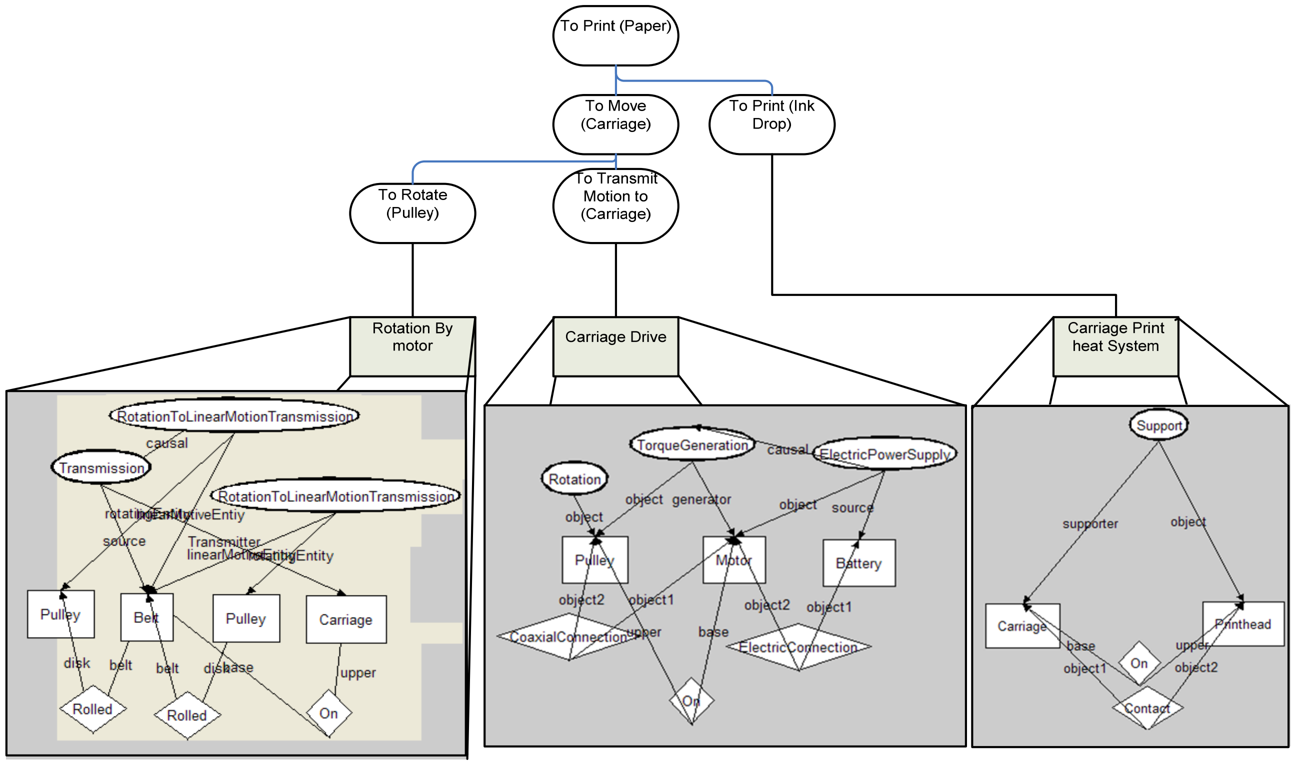

Figure 5: FBS representation of the primary model.

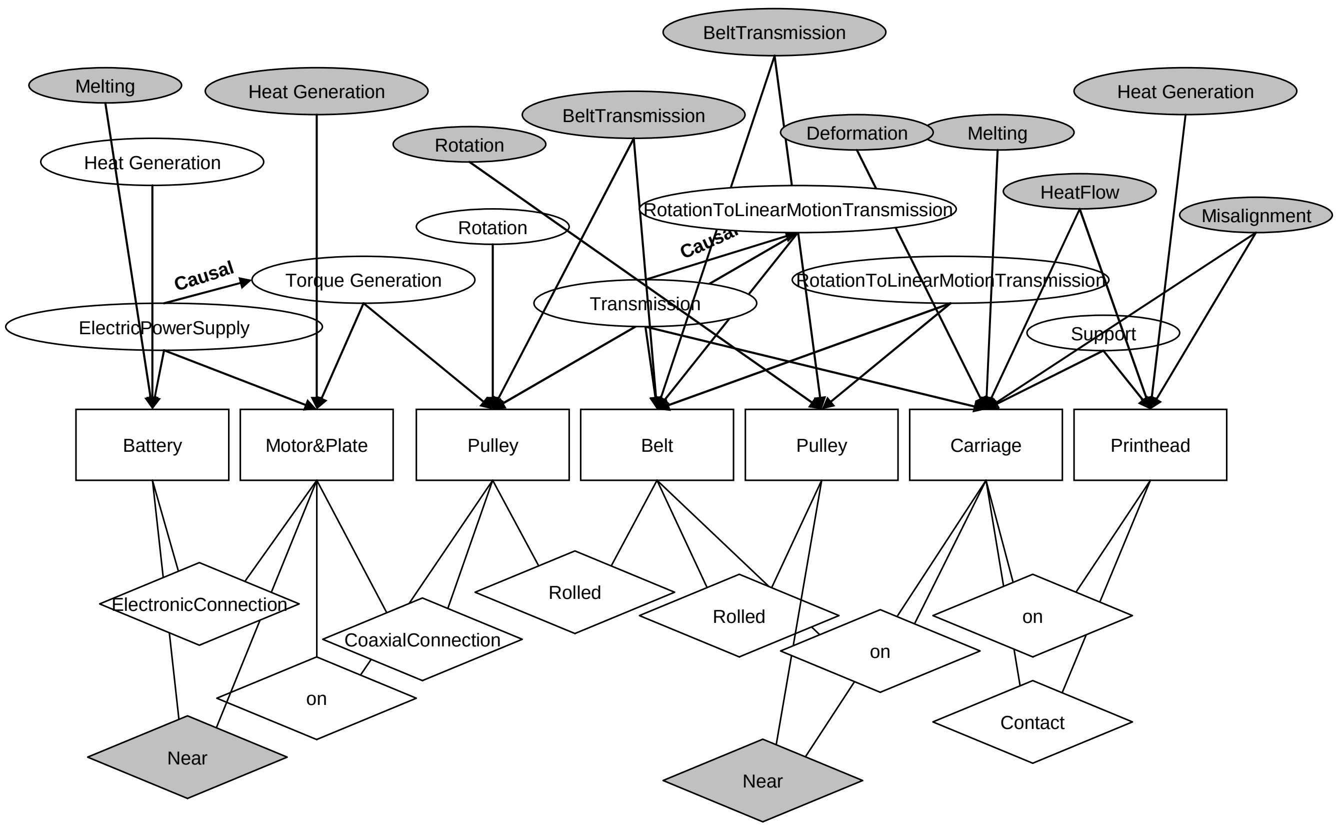

Figure 6: FBS representation of the reasoned model.

4.2 Behaviors of primary model and total behavioral model.

When the FBS model is complete, which means that all the unexpected physical phenomena are reasoned out, the model can be transferred to the qualitative physics reasoner. The task of this reasoning engine is to combine all the information which is represented in the FBS model. Sequences of state transitions are reasoned out which represent the behavior of the system.

The tables 1 and 2 graphically show behaviors of the initial model and of the reasoned model in terms of state transitions.

4.2.1 Results for the primary model

Just four states are reasoned out for the initial model by the qualitative physics reasoner. Behavior is represented by the sequence of state transitions over time. Values of attributes for each state are shown in table1. Attributes which are related to the primary model are shown in the first column of table 1. These attributes are generated by the physical phenomena introduced by the designer (figure 6 white oval blocks).

All values of attributes turn to the qualitative value ‘plus’ in the fourth system state. This is the expected result of the designer and it can be interpreted as the nominal behavior of the inkjet printer. The first three states represent the transition to the generation of the final state.

When the pulley torque is positive and all entities have positive velocity and acceleration, the print-head is moving without any problem along the guide. Nothing is specified about the print quality, but it indicates that no strange phenomena are disturbing the nominal value.

4.2.2 Results for the total behavioral Model

Table 2 takes into account just one of the reasoned behaviors coming from the reasoned model. Figure 7 shows the total amount of behaviors reasoned out for this example. 64 system states are generated. This represents a big transformation in comparison to the four states generated for the initial model. Each state transition represents a different system behavior. Therefore, it is evident from figure 7 that the real amount of behaviors generated by the DID for the reasoned model is much larger than in the initial model. Analyzing each behavior requires long time to the designer. The computational time for reasoning behaviors also increases exponentially for the reasoned model in comparison to the initial model.

It is important to pay attention to the value of the attribute ‘Print-head accuracy’ represented in table 2. This refers to the precision with which the ink is positioned on the paper. At the fourth state the value of ‘print head accuracy’ turns to minus while all the other attributes maintain plus or zero value. This means that print accuracy decreases while the other attributes are reaching the positive and nominal value. A low value of accuracy can represent an undesired and unexpected behavior of the system. This is an important result which was not reasoned out from the initial model.

Next step in the analysis is to detect the origin of the problem by looking into the parameter network of the reasoned model which is automatically generated by KIEF. The parameter network for the reasoned model of the inkjet printer is shown in figure 8. Result of the analysis is that print-head accuracy depends on the print-head displacement, which is connected to the displacement of the carriage, which depends on print-head temperature.

The parameter network suggests to the designer that decreasing the print head temperature leads to reduce the carriage displacement and finally to have a better print accuracy. In the example, ‘print head’, which incorporates a heater system to fuse ink, is the main source of ‘heat generation’. To avoid loosing accuracy in the print, the designer has to act on the temperature of the print-head. For instance he/she can add a pre-heater before the ink enters the print-head or use another kind of ink which has lower melting point. In this way the effect of the print head temperature on the carriage temperature will decrease and deformation leading to misalignments will not be generated.

In this section the article has showed how already with the trivial model of an inkjet printer, the designer can create new inventive solutions at an early stage.

State Transition	1st state	2nd state	3rd state	4th state
Belt1 Position	0	0	0	1
Belt1 Velocity	0	0	1	1
Carriage Position	0	0	0	1
Carriage Velocity	0	0	1	1
Motor&Plate Voltage	0	1	1	1
Pulley1 Angular velocity	0	0	1	1
Pulley2 Angular acceleration	0	1	1	1
Pulley2 Torque	0	1	1	1

Table 1: Qualitative Behavior of the primary model.

	1st state	2nd state	3rd state	4th state	5th state	6th state	7th state	8th state
Battery Temperature	0	1	1	1	1	1	1	1
Belt Position	0	0	0	0	0	0	0	1
Belt Velocity	0	0	0	0	0	0	1	1
Carriage Dispacemet	0	0	0	1	1	1	1	1
Carriage Position	0	0	0	0	0	0	0	1
Carriage Temperature	0	0	0	1	1	1	1	1
Carriage Velocity	0	0	0	0	0	0	1	1
Motor&Plate Temperature	0	0	0	0	1	1	1	1
Motor&Plate Voltage	0	0	0	0	0	1	1	1
Printhead Accuracy	0	0	0	-1	-1	-1	-1	-1
Printhead Dispacement	0	0	0	1	1	1	1	1
Printhead Temperature	0	0	1	1	1	1	1	1
Pulley1 Angular acceleration	0	0	0	0	0	0	0	1
Pulley1 Angular velocity	0	0	0	0	0	0	1	1
Pulley2 Angular acceleration	0	0	0	0	0	1	1	1
Pulley2 Angular velocity	0	0	0	0	0	0	1	1
Pulley2 Torque	0	0	0	0	0	1	1	1

Table 2: Qualitative behavior of the reasoned model.

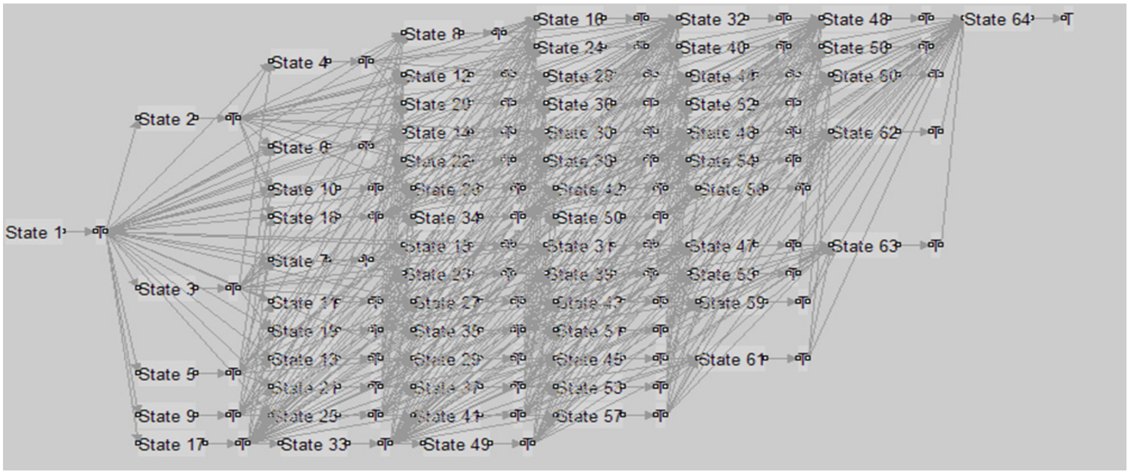

Figure 7: State transitions of the reasoned model.

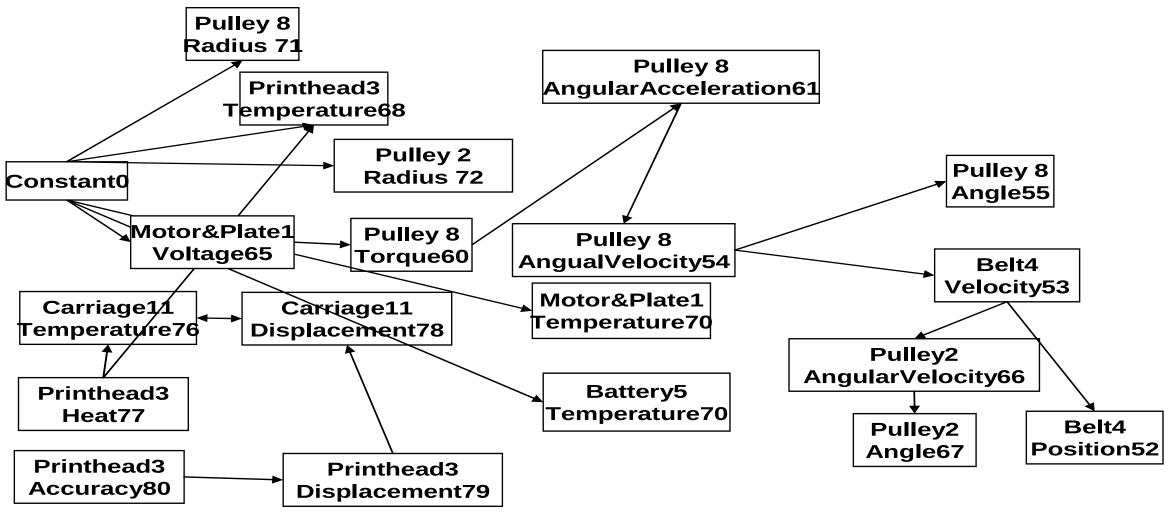

Figure 8: Parameter network connections.

5 CONCLUSIONS, LIMITS AND FUTURE WORK

This paper suggested a method and a software tool, the DID, for automatically detecting unpredictable problems in the conceptual design of complex machines such as mechatronics machines.

The DID generates a shortcut between early design stage and high level verification. The DID is an analysis tool which identifies design failures which are generated by integration of technologies. Finding and resolving problems at the conceptual phase increases the design quality, it helps to save money in prototypes reducing the amount of flaws at the engineering prototype phase.

Two different models of the top engine of an inkjet printer were analyzed and compared: primary model and total behavioral model. In section 4 differences between the two models in terms of behaviors were investigated. The analysis leaded to the suggestion of two alternative design solutions in order to avoid the quality of the print to decrease. Such predictions at an early stage of the design save months in the development of the product and they are cost efficient since fewer prototypes will be then needed in the verification phase. Unfortunately the simulation time and the reasoned behaviors increase exponentially with the amount of attributes. The risk is to generate an explosion of solutions as it happens in the reasoned model. This is why it is necessary to filter solutions. A brief introduction of filtering methods has been provided in section 4.

The analysis suggests the use of the filter to reduce the number of possible unpredicted phenomena before the qualitative physics reasoner is activated. In this way not only the number of behaviors but also the simulation time is reduced.

The software has reasoned significant results at the design level just by using qualitative information. The lessons learned by the designer concerning the product in terms of unpredicted phenomena and the solutions provided for the problems can be introduced into the database in terms of physical features. This allows the re-used of the reasoned results in subsequent sessions or in a new model. This avoids having the same unpleasant surprise again.

The next step for this research is to develop the two mentioned filtering methods to prioritize behaviors reasoned out by the system based on their significance and probability to appear.

6 ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the Dutch Innovation Oriented Research Program ‘Integrated Product Creation and Realization (IOP-IPCR)’ of the Dutch Ministry of Economic Affairs.

7 REFERENCES

[1] Suh, N. P., 1990, the principle of Design, Oxford Series on Advanced manufacturing.

[2] Pahl, G., Beitz, W., 1977, A Systematic Approach, Engineering Design Berlin, Springer-Verlag.

[3] Browning, T. R., 2001, Applying the design structure matrix to system decomposition and integration problems: a review and new directions, IEEE Transactions on Engineering Management, 48: 292-306.

[4] Chatterjee, P., 1975, Modularization of Fault Tree: A method to Reduce the Cost of Analysis, Reliability and Fault Tree Analysis, SIAM: 101-126.

[5] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y, Tomiyama, 2004, T., Physical concept ontology for the knowledge intensive engineering framework, Advanced Engineering Informatics, 18: 95-113.

[6] Stevens, R., Brook, P., Jackson, K., Arnold, S., 1998. System Engineering: Coping With Complexity, Prentice Hall Europe, ISBN 0-13-095085-8.

[7] Kari, L., Provisec, O., 2005, Challenging Global Competition: tune up your product development, VTT working paper 34, ESPOO.

[8] Tran, E., 1999, Verification/Validation/Certification, in Koopman, Topics in Dependable Embedded Systems, USA Carnegie Mellon University, http://www.ece.cmu.edu/~koopman/dess99/swtesting/index.html, accessed April 13, 2008.

[9] Kuipers, B., 1994, Qualitative Reasoning Modeling and Simulation with Incomplete Knowledge, p. cm. Artificial Intelligence.

[10] 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*, Analysis and Manufacturing, 10, No. 4: 275-288.

[11] Forbus, K.D., 1984, Qualitative Process Theory, 1984, Artificial Intelligence, 24: 85-168

[12] 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).

[13] D’Amelio, V., Tomiyama, T., 2008, Design at the Cross-Section of Domain, the 18th CIRP Design conference, April, 7-9, Enschede, the Netherlands.

[14] Kiriyama, T., Tomiyama, T., Yoshikawa, H., Qualitative Reasoning In Conceptual Design With Physical Features, 1992, Recent advanced in qualitative physics, Eds. B. Faltings, P. Struss, The MIT Press, Cambridge, MA: 375-386.

[15] Forbus, K. D., De Kleer, J., 1993, *Building Problem Solvers*. Artif Intelligence, Cap. 6-11.