ECO TRACING - A SYSTEMS ENGINEERING METHOD FOR EFFICIENT TRACELINK MODELLING

Rainer Stark[1,2] and Asmus Figge[1]
(1) Technische Universität Berlin, Germany (2) Fraunhofer IPK, Germany

ABSTRACT
Using expertise and combining functionalities from different domains has led to a significant increase of information engineers have to deal with. It is hardly possible to identify influenced components in activities like change requests. A model containing tracelinks between the elements of involved partial models as an essential part of PLM based Systems Engineering helps to overcome this deficit. The main obstacle for a broad introduction of traceability is the significant workload involved in creating tracelink models as every element combination has to be examined for dependencies. This calls for an approach to support developers in creating tracelink models more efficiently.
The presented approach *Eco Tracing* allows developers to significantly *Eco*nomize modelling effort. In order to do so, the method uses the hierarchical structure of many models and a top-down analyzing approach to exclude element combinations prior user examination. Furthermore *Eco Tracing* allows choosing the desired level of detail flexibly while modelling. *Eco Tracing* is a promising approach helping to establish traceability in product development by reducing modelling effort significantly.

Keywords: traceability, dependency modelling, product development, mechatronics

1 INTRODUCTION AND MOTIVATION
A challenge many companies face today is the increasing complexity of products. Complexity is defined as a property of a system depending on the amount of system elements, the number of relations between them and the multitude of its possible states [1]. Regarding mechatronic product development there are numerous sources for complexity, such as the product's augmented, cross-domain functionalities.
Embedding and integrating functionalities from different domains is a major source of innovation [2], but on the other hand it has led to a significant increase in information that developers have to deal with. It is impossible to oversee all implications of activities like change requests since influenced components from other domains cannot be identified. Thus, changes in complex systems are very difficult to handle [3]. This is especially challenging since changes occur very often: usually more than 50% of system's requirements get modified before it is eventually put into service [4]. Nonetheless there is little tool and methodological support for that challenge.
During product development several partial models are created (e.g. function structure) all describing the final product. Between some elements of these partial models there is a dependency with regards to their content – e.g. function "cool passenger compartment" satisfies requirement "provide means for the regulation of passenger compartment temperature". An approach to deal with the mentioned challenges is the continuous linkage between all partial models. Changes can be propagated efficiently and implications can be detected easily based on dependencies between partial models. These dependencies can be represented and documented through tracelinks, which mainly contain information about the linked source and target model elements. The degree to which a relationship can be established between those models is called traceability [3]. Especially in software engineering traceability is viewed as a measure of system quality and process maturity and is mandated by many standards such as MIL-STD-498, IEEE/EIA 12207, and ISO/IEC 12207. But also in Systems Engineering tracelink models can be of significant assistance. For example by automatically filling in information into QFD or FMEA spreadsheets helping to ease the use of established but infrequently used quality and reliability methods.
The main obstacle for a broad introduction of traceability in system development is the significant workload involved when creating and maintaining tracelinks as well as the discrepancy between the persons modelling and using tracelinks [3]. In 2000 the US Department of Defence spent about 4% of

their system development costs on traceability [5], emphasising the necessity to have a good and efficient traceability strategy [6].
The situation described calls for an approach to support product developers with methods and tools helping to reduce the necessary effort for primarily building up and controlling complex systems and dependencies between their comprising models. For this reason, the approach *Eco Tracing* presented in this paper focuses on how to use hierarchical characteristics to model tracelinks in an efficient way. It has been integrated in the software prototype Model Tracer, which is a tool for tracelink modelling [7].
In this paper, the state of the art and research is described in section 2 and the shortcomings in the area of tracelink identification are discussed in order to illustrate the motivation for *Eco Tracing*. Section 3 introduces the Model Tracer, which is a tool for tracelink modelling. During the project ISYPROM the *Eco Tracing* method was integrated into the Model Tracer and tested in an industrial environment. In section 4, *Eco Tracing*, its basic principles, its effectiveness as well as its implementation are detailed. Section 5 provides an overview over the application of the *Eco Tracing* method in context of a systems engineering example. Section 6 concludes the findings presented in this paper and hints at further research in the area of tracelink identification and maintenance.

2 STATE OF THE ART AND RESEARCH

Procedure models are supposed to help developers coping with complexity in product development. Cooperation between different domains or departments is promoted through their application [8]. In most design steps models are created (e.g. requirements model, functional model, structural model), which represent different development perspectives on the same system. These models are developed based on each other. In this way, e.g. functions of a system are derived from the requirements, which have been defined before. Since this happens implicitly, dependencies between partial models are usually not documented [9].
A way to cope with this problem is to establish traceability between different partial models by explicitly linking the information contained in the models. Maurer proposes the use of matrices for the connection of information. He differentiates between Design Structure Matrices (DSM), Domain Mapping Matrices (DMM) and Multi Domain Matrices (MDM), which are a combinatorial advancement of the first mentioned. Goal of his studies is to analyse, control and improve the dependencies of complex systems. For this purpose he suggests a number of analysing techniques that allow for the identification of connected structures by rearranging matrices or the tracing of impact chains. For visualisation purposes he uses matrices or strength based graphs, if the number of connected elements becomes too large. Both visualisations can be derived from each other and are included in the commercial software Loomeo [10], [11].
The Change Prediction Method (CPM) tool developed at the Engineering Design Centre in Cambridge combines different visualisation techniques that help understanding a system. The CPM tool aims at decision assistance by providing effective and intuitive information visualisation regarding the prediction of change propagation [12].
METUS by ID-Systems provides functionalities to specifically model functions, subfunctions, components and modules of a product as well as dependencies between their elements. These tracelinks are used by several analysis functions in order to e.g. optimize costs or the weight of the system in focus. To reduce the work to model and analyse the system, a PLM integration has been implemented, which enables METUS to acquire product information [13], [14].
ToolNet, which was developed in a research project by DaimlerChrysler and EADS, completely follows this approach of acquiring information from existing sources. Partial models are created in the established authoring software while only tracelinks between the models are created and stored in ToolNet [15].
A complementary approach is followed by Reqtify. In this approach, tracelinks are not managed in an additional application but stored in the original documents with the help of references to elements of other documents. Reqtify is then used to visualise these tracelinks, perform several analyses and to export reports [16].
All described traceability approaches aim at an improvement of system's understanding and model consistency during the development. This means that there are a lot of ways to use the already modelled tracelinks in a beneficial way. But none of the presented approaches propose any effective solutions on how to identify the dependencies between the partial models' elements in order to model

the tracelinks in the first place. Even though especially the manual creation and the effort necessary to achieve high connection quality (avoidance of wrong and missing dependencies) still pose high challenges for their application [3], [10]. Present suggested approaches for this problem, like the use of interdisciplinary workshops or the collection of existing data (e.g. QFD, TRIZ) still imply high efforts. Only in software development there are some approaches to generate tracelinks automatically, e.g. as the result of model transformation or the statistical interpretation of change histories in order to identify dependencies between items [3]. But these approaches either have not yet been adapted to challenges posed in mechatronic system development or can't be easily applied to it because of different procedures (e.g. no model transformations). For this reason it is necessary to offer more methods and tools that help limiting the additional work effort in order to further integrate the creation of tracelinks in today's system development reality.

3 MODEL TRACER – A TOOL FOR DEPENDENCY MODELLING IN SYSTEMS ENGINEERING

In order to cope with the mentioned challenges, a prototypical software tool was developed – called Model Tracer. The general approach of Model Tracer provides means to define tracelinks between different partial models (Figure 1). Each qualitative tracelink represents a general dependency between a pair of elements from two different partial models [7].

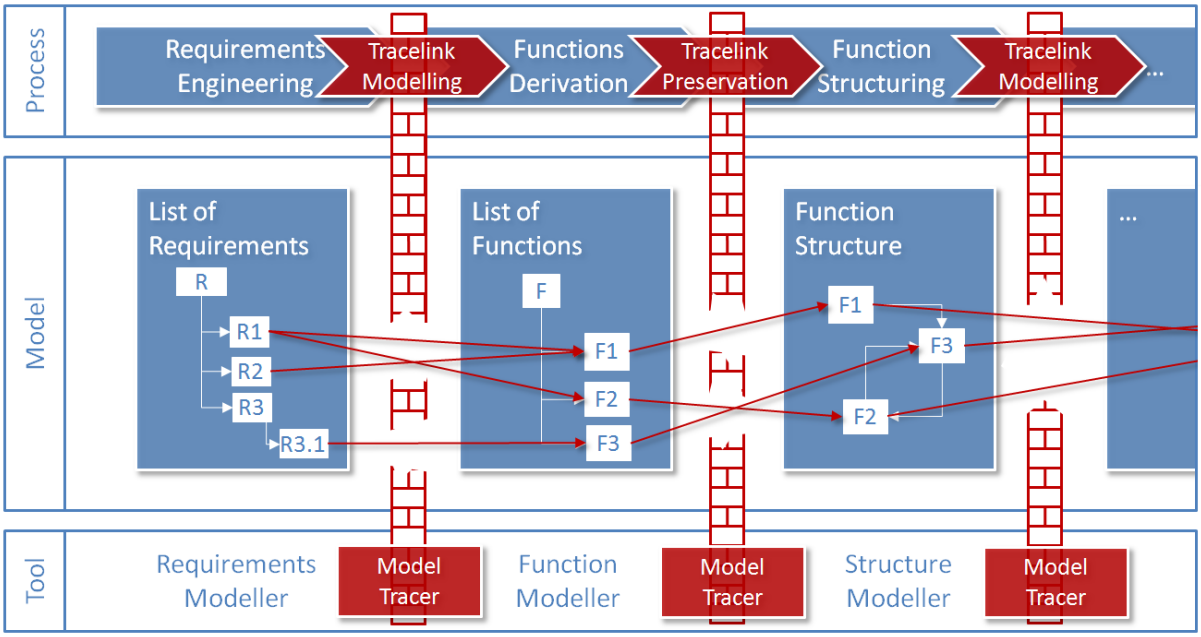

Figure 1: Process steps and models enhanced by tracelinks created with Model Tracer.

Since different partial models are not necessarily created with the same tool it is important to allow tracelink modelling while avoiding to introduce "just another tool" that aims at replacing well established authoring tools.

Therefore Model Tracer acquires all models from their specific authoring tools and visualises them in its own graphical user interface (GUI) as shown in Figure 2. During the project ISYPROM interfaces for common standards like ReqIF (Requirements Interchange Format) and PLMXML have been implemented to exemplarily prove feasibility. Basically the approach allows for importing and using any hierarchical model in Model Tracer if an interface is implemented. Model Tracer only saves information regarding the tracelinks and references to the connected models.

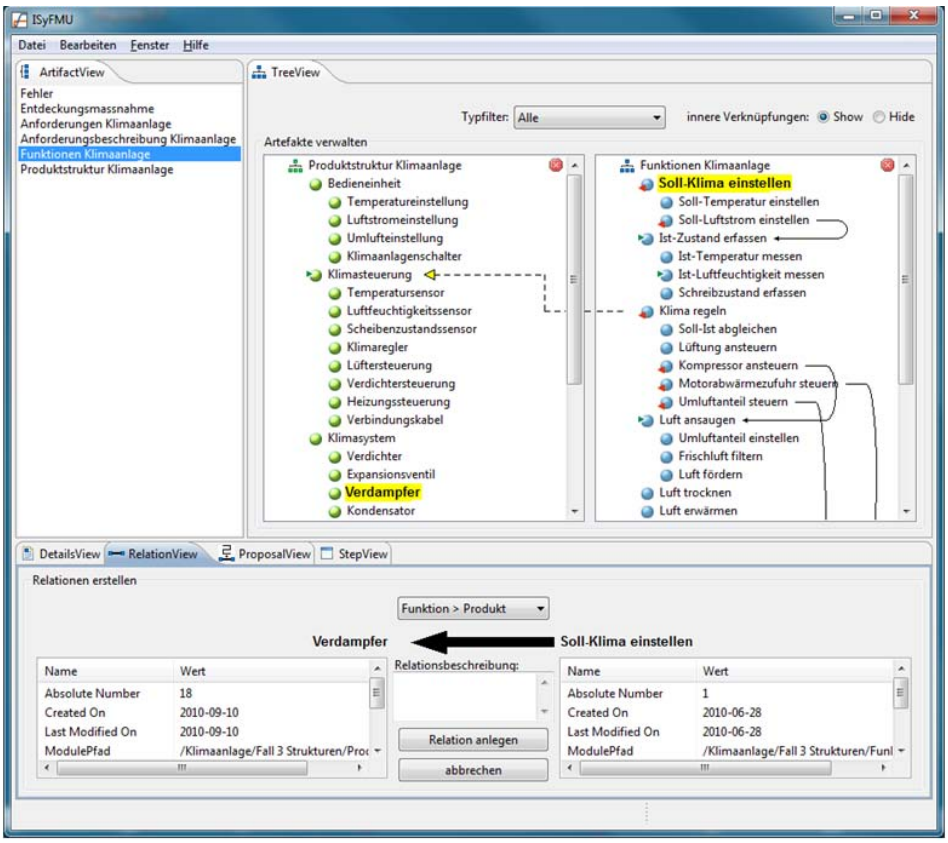

Figure 2: GUI of Model Tracer

With every start of Model Tracer, partial models are loaded from their proprietary databases. Modified and new elements are highlighted which helps guaranteeing actuality of data. As mentioned before, especially the laborious identification of dependencies is one of the biggest challenges and the main hindrance for an industrial application of methods to connect information.

4 ECO TRACING - A SYSTEMS ENGINEERING METHOD TO MODEL TRACELINKS EFFICIENTLY

To increase efficiency in tracelink modelling in the course of Systems Engineering two basic categories have to be considered. There are technological approaches using diverse algorithms to identify dependencies between elements; this is used for example in model transformation or automatic tracelink recovery [3]. On the other hand there are methodical approaches which help users to create tracelinks efficiently. The latter can be especially useful when initially modelling dependencies. For that activity a structured procedure is needed to make sure all dependencies are identified. A methodical approach to *Eco*nomize tracelink modelling is introduced in this paper: *Eco Tracing*. In Section 5 details on how to apply *Eco Tracing* in Systems Engineering will be discussed.

4.1 Basic Principles for Eco Tracing

The analysis of other approaches in the field of system development reveals that traceability is rarely used in industry yet. A major reason for its lack of application is the high effort needed for the manual creation of tracelinks [12], [10]. A widely used approach makes use of interdisciplinary workshops where all possible combinations of elements of both partial models have to be considered individually. For each pair of elements a decision regarding their dependency has to be made [10]. Since many combinations of elements do not feature a dependency, their examination is unnecessary and thus causing additional costs. This means the biggest challenge is to identify the majority of elements, which do not have to be analysed, in advance. Furthermore, a high flexibility in choosing the necessary level of detail is desirable when modelling tracelinks. This flexibility gives users the opportunity to decide in which areas to model in detail and where to stop at a high hierarchy level to avoid high efforts.

With this in mind, characteristics of nested hierarchical structures, which many models feature (e.g. requirements, functions or assemblies), in combination with a consequent top-down-approach can be used to exclude independent element combinations: In nested hierarchical structures, parent elements consist of or contain child elements (e.g. assemblies consist of parts) [17]. This leads to the conclusion that no child element depends on a certain element, if the dependency was dismissed for its parent element (which consists of its child elements).

Figure 3: If there is no dependency between R1 and F2, no dependency exists between R1 and all child elements of F2.

Figure 3 is illustrating this conclusion: if there is no dependency between requirement R1 in the list of requirements and function F2 in the list of functions, there will be no dependency between R1 and sub-functions F2.1 and F2.2 either.
This conclusion has been applied to requirements models, functions structures and product structures successfully. Since it is the main principle *Eco Tracing* is based on, it is recommendable to verify its applicability for any new model prior application.
In the following section *Eco Tracing* is described in detail.

4.2 The Method Eco Tracing

As mentioned before *Eco Tracing* is a top-down-approach, meaning that users are starting every analysis at a high hierarchical level and work their way down to lower levels. In the example illustrated in Figure 4 the examination of dependencies following the *Eco Tracing* approach would therefore start with the combination of A1 and all parent elements in model B (B1, B2, and B3). If no dependency is identified between A1 and B1, no tracelink is set (illustrated by a red cross in Figure 5). Since parent elements contain or consist of their child elements, no dependencies from A1 to any of B1's child elements are possible. That is why these combinations do not have to be analysed at all. To maintain this knowledge for later reference all combinations from A1 to every child element of B1 are marked with a flag meaning they have been examined for dependencies and dismissed (illustrated by a red cross in Figure 5).

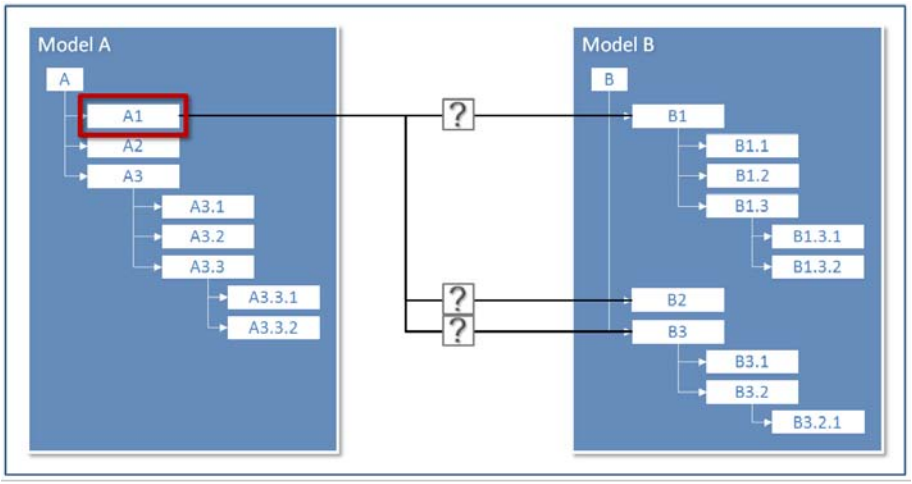

Figure 4: Examination for dependencies.

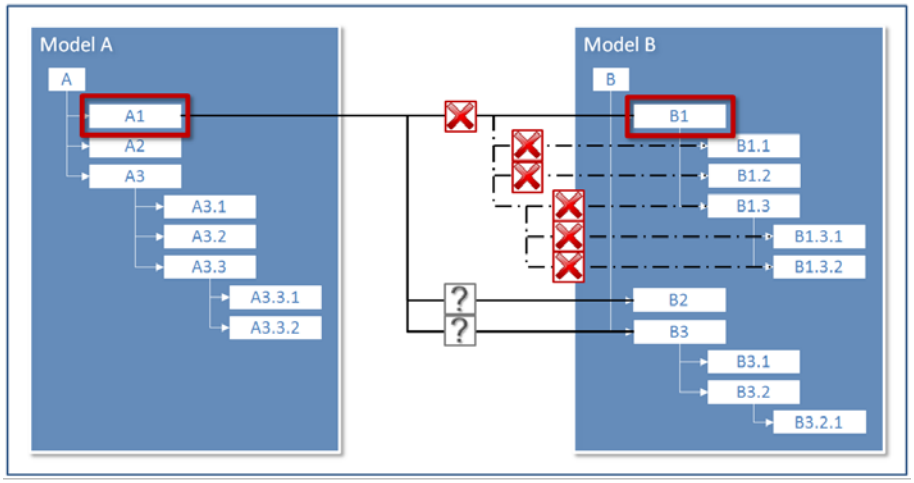

Figure 5: Flagging dismissed dependencies.

If a dependency is approved, a tracelink is set for this combination illustrated by a green check mark in Figure 6. The conclusion that "no child element depends on a certain element, if the dependency was dismissed for its parent element" is now inverted: it is very likely that at least one child element of B3 features a property that has led to the affirmation of the dependency to A1. Therefore all possible combinations of A1 and all direct child elements of B3 (B3.1 and B3.2) are flagged (illustrated by green flags) meaning at least one dependency exists within those combinations (Figure 6).
With the help of the flags it is possible to differentiate between:

- Combinations of elements which have been examined and were rejected,
- Combinations of elements which have not been examined yet,
- Combinations of elements where dependencies are likely but no detailed examination has been performed.

This differentiation of states is especially important as it enables users to discontinue the examination for dependencies at any time, allowing them to continue at a later time or to finish at any level of detail.

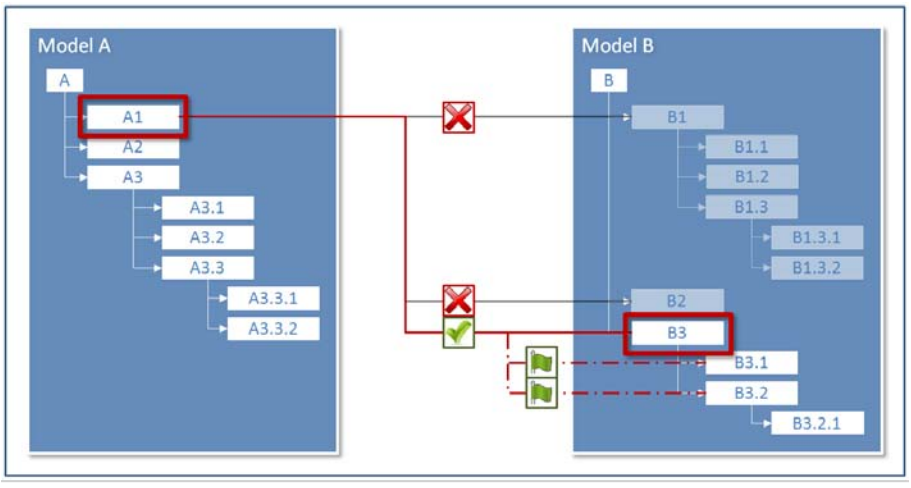

Figure 6: Setting a tracelink and automatically flagging its child elements.

After investigating all combinations of A1 and Bx the approach proceeds with the investigation of all green flagged combinations of A1 and Bx.x (in this case A1 – B3.1 and A1 – B3.2). When traversed all the way to the lowest user-chosen hierarchy, the investigation restarts with the next parent element in model A (in this case A2) from the beginning.

4.3 Eco Tracing vs. conventional Dependency Examination

In this section *Eco Tracing* is compared to the conventional way for dependency examination (e.g. workshops [10]) with the help of a simple example. This example is illustrated in Figure 7 and comprises two models with a total of 19 elements (white blocks) and 17 dependencies (red lines). For the purpose of calculation it is assumed that during the application of the conventional method all element combinations have to be examined, even though in praxis the analysis of some combinations could probably be dismissed by the experts.

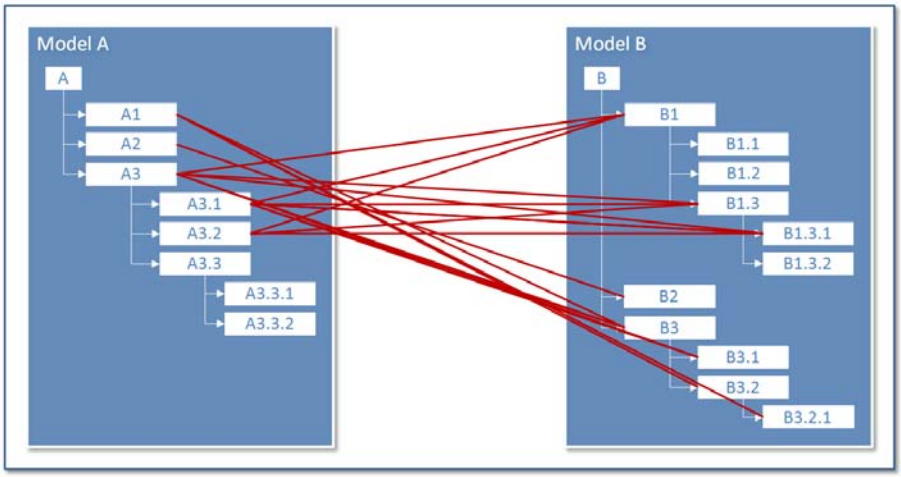

Figure7: Dependencies between model A and model B.

Table 1 shows the same example as Figure 7 in a different visualisation type and with some additional information. All rows contain elements of model A whereas the columns comprise model B. It is assumed all elements are of interest and are examined. If a dependency between two elements exists a '1' is entered in their common cell otherwise a '0'. All combinations examined with *Eco Tracing* are in white cells, automatically dismissed ones in grey cells.

The conventional way means to examine every combination of elements from model A with those from model B. Since there are eight elements in model A and eleven elements in model B, *88 combinations* have to be checked for dependencies.

If *Eco Tracing* is applied not all combinations have to be checked since several dependencies are already dismissed on a higher hierarchical level. That is why only *32 combinations* have to be checked (number of white cells in Table 1) which means the effort for examining *56 combinations* is saved.

Table 1: Comparison between Eco Tracing and a conventional method for dependency examination.

	B1	**B1.1**	**B1.2**	**B1.3**	**B1.3.1**	**B1.3.2**	**B2**	**B3**	**B3.1**	**B3.2**	**B3.2.1**
A1	0	0	0	0	0	0	0	1	0	1	1
A2	0	0	0	0	0	0	1	0	0	0	0
A3	1	0	0	1	1	0	0	1	1	0	0
A3.1	1	0	0	1	1	0	0	1	1	0	0
A3.2	1	0	0	1	1	0	0	0	0	0	0
A3.3	0	0	0	0	0	0	0	0	0	0	0
A3.3.1	0	0	0	0	0	0	0	0	0	0	0
A3.3.2	0	0	0	0	0	0	0	0	0	0	0

Under the assumptions made, the saving is about 60 %. But the effort which can be saved by *Eco Tracing* depends on the hierarchical structure of the models and in which hierarchy levels the dependencies exist. For example, if a parent element has few child elements, the effort possible to be saved is less than if it would have many child elements. Therefore no general conclusion about the effectiveness of the method can be quantified.

4.4 Eco Tracing Prototype

Eco Tracing as described in Section 4.2 is implemented as a plug-in of Model Tracer introduced in Section 3. The aim of *Eco Tracing* is to guide users in a wizard-like environment through the process of examining dependencies while reducing the necessary effort.

When starting *Eco Tracing* a selection dialog is displayed. In a first step parts of model A need to be selected. Based on this selection and under consideration of the flags that have been set before, a reduced view of model B is displayed. User support is given by excluding all elements of B that have already been examined or cannot have a dependency to one of the selected elements of model A, reducing the overall amount of information. Additionally users can further limit their selection within this reduced view of model B.

This selection dialog is especially helpful, when examining combinations of very complex models with a large quantity of elements. When for example dependencies between requirements and functions are examined it is possible to choose only the functional requirements, skipping the non-functional requirements and thus increase clarity.

Once a selection is made dependency examination begins. Figure 8 shows a screenshot of *Eco Tracing* in use. In the left and right column the selected models are displayed. The elements comprising the combination being examined at the respective moment are highlighted in green. For every combination users have three possibilities:

- Skip: Skipping the examination regarding a dependency between the two elements. All combinations of their child elements will also not be examined.
- Dismiss: If there is no dependency between the elements, the tracelink is dismissed. All combinations of their child elements will also be dismissed.
- Approve: If there is a dependency between the elements, a tracelink is set.

For motivational purposes a progress bar has been added visualising the rapidly growing ratio of already examined and overall combinations.

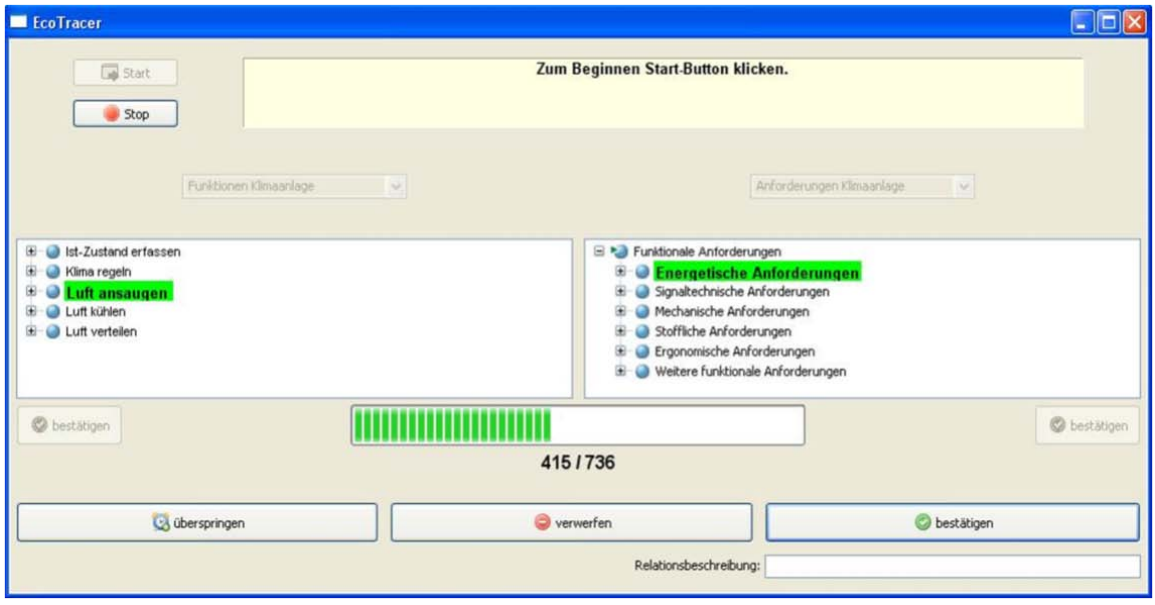

Figure 8: Examination of dependencies.

Eco Tracing provides users with the opportunity to economize their modelling efforts significantly. Numerous element pairs can be excluded from examination before users even get to see them. Users are also given the possibility to choose the level of detail for tracelink modelling on the fly - skipping entire parts of a model if desired. By allowing different levels of detail within one and the same model combination, dependency modelling can be realised as accurate as necessary with as little effort as possible. Through the guided procedure implemented in the *Eco Tracing* wizard users cannot get lost. They are systematically guided through all possible combinations of elements selected and always get provided with relevant context information. Once a decision has been made *Eco Tracing* saves this information to preclude redundant modelling work and to allow cancelling the process at any time.

5 APPLICATION IN SYSTEMS ENGINEERING

In Figure 9 an interdisciplinary system development process is illustrated. It is an adaption of the V-model according to VDI 2206 [2]. In the early phases of system development (represented by the left side of the V-model) a large amount of different models, determining the system's properties, is developed. Furthermore, designing is an iterative process and prone to numerous changes. That is why the potential for cost savings with the help of traceability, once the tracelinks are modelled, is especially high. *Eco Tracing* can efficiently provide tracelink information enabling continuity in early design phases, which are shortly described in the following paragraphs.

In the phase of *Product Planning* an alignment of the planned product with the business strategy is carried out. Business and system requirements are developed. During *System Architecture Reflection* the predecessor's system architecture is used as an initial point for discussions about the new product. Innovations and new systems are roughly integrated. These changes and plans for the new product are consolidated in the *Requirements* phase into new and changed high-level requirements. These are further detailed during the *Requirements Cascade* until they can be allocated to product functions. In order to satisfy all requirements it is necessary to add, remove, change and restructure functions during the *Development of Function Structure*. In the *Development or Adjustment of System Architecture* new solution elements for executing functions are added, out-of-date ones are removed or changed and system boundaries are defined. The last step of the *Continuous System Design*, is the *Partitioning of Models.* All models developed during preceding phases (requirements, functions, solution elements and behaviour models) are partitioned to the disciplines: mechanics, electronics / electrics (including software development), services and process & resources.

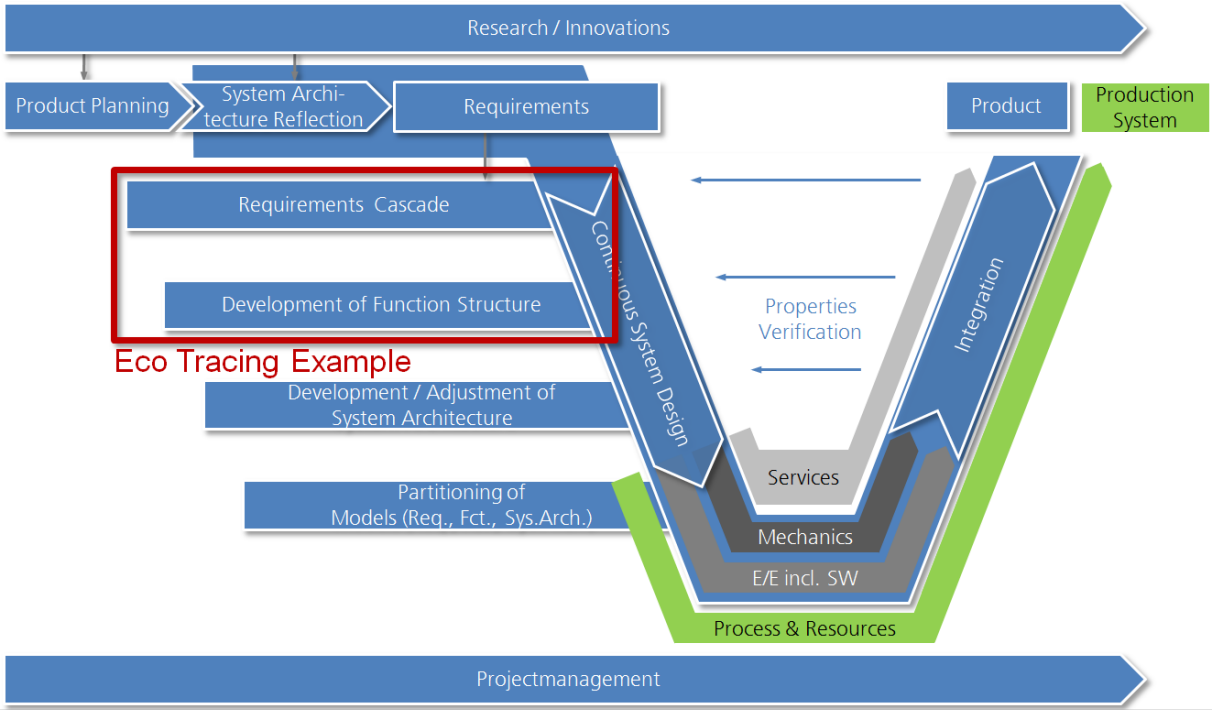

Figure 9: Allocating the described Eco Tracing example in a system development process.

During ISYPROM a hypothetical industry-like example was developed according to the interdisciplinary system development process in order to verify elaborated methods. In the following, this example is used to explain the application of *Eco Tracing*. The described region of the process is marked with a red rectangle in Figure 9. The chosen example deals with an engineering adaptation of an existing air conditioning system for a car. Cause for the adaptation is the ratification of EU directive 2006/40/EC. A high level requirement is added requesting the compliance of the air conditioning with 2006/40/EC. As all high level requirements it is further detailed causing several new and revised system level requirements. One of which is the requirement to exclusively utilize refrigerant fluids with a maximum global warming potential (GWP) of 150. All requirements are

added to the existing requirements model. Following, the new requirements have to be connected to the function structure to maintain traceability.
Starting *Eco Tracing* a user needs to select both partial models which need to be examined. In this case: requirements and function structure. The selection is narrowed down to the new and changed requirements, while the unchanged function structure with its functions and sub-functions is selected as a whole. The *Eco Tracing* wizard then leads the user through the examination process. The user has to decide for each combination if a dependency between the elements is existent. For example, a tracelink has to be set between the described system level requirement "GWP of 150 or less" and the main-function "cool passenger compartment". With the help of *Eco Tracing* only 12 of 30 main- and sub-functions contained in the function structure have to be examined in combination with this specific requirement. For five of these combinations tracelinks are established and seven are dismissed. Thus 18 combinations can be ignored, saving 60 % examination effort and accelerating the process. As mentioned before, this number strongly depends on the amount of levels in the models' hierarchy. That's why no general conclusion about the effectiveness can be made, but in several tests during ISYPROM the possible savings were between 60 % and 75 %.

6 CONCLUSION AND OUTLOOK

Recent projects have shown that traceability is of particular interest to industry when mechatronic systems are developed. The usage of Model Tracer aims to help developers to achieve effective cross-model traceability and complexity management.
In this paper a number of existing approaches to manage tracelinks have been presented, which provide sophisticated functions for an improved model consistency as well as understanding and analysis of systems. While they concentrate on the beneficial use of tracelinks, there is little support for their identification and modelling.
In order to cope with this deficit, the approach *Eco Tracing* has been detailed which utilises characteristics of nested hierarchies, often found in models in mechatronic system development. In doing so, many combinations of elements which usually have to be analysed during tracelink modelling can be excluded prior examination. The possible savings in effort highly depend on the amount of levels in the hierarchy. But several tests could show that the possible savings compared to a conventional method (where all element combinations have to be examined) were between 60 % and 75 %. An overview of the features of *Eco Tracing* and its benefits is provided in Table 2.

Table 2: Features and Benefits of Eco Tracing

Feature	Benefit
Economize Modelling Efforts	
Flag all child elements of rejected parent elements	Skip unnecessary examination of a huge amount of element combinations
Choose Level of Detail flexibly	
Top-down processing offers the opportunity to skip branches of the model if desired	User can choose the level of detail for tracelink modelling on the fly
Follow a guided Procedure	
Eco Tracing wizard	User is systematically guided through all possible combinations of selected elements
Conserve Knowledge	
Examined element combinations are flagged and those decisions are saved	Preclude redundant modelling work

"Each method for automatic relationship discovery provides one piece of the puzzle, but to obtain a complete picture, we still need to fit the pieces together and develop methods to integrate them to provide a complete solution." [3] Therefore the presented approach *Eco Tracing* is yet another piece of the puzzle towards a complete solution raising significance of traceability in industrial system development.
Future research will focus on aspects of assistance for dependency identification and maintenance e.g. by use of methods from semantic web applications in order to identify false and missing tracelinks.

The aim is to reduce the necessary effort to create and maintain dependency models and to gain a maximum benefit of them with respect to the overall quality of and a reduced development time for the developed system.

ACKNOWLEDGEMENTS

The research and development project ISYPROM is funded by the German Federal Ministry of Education and Research (BMBF) within the Framework Concept ”Research for Tomorrow’s Production” (funding number 02PC105x) and managed by the Project Management Agency Karlsruhe (PTKA). The author is responsible for the contents of this publication.

REFERENCES

[1] Ulrich, H. and Probst, G. J. B. *Anleitung zum ganzheitlichen Denken und Handeln: Ein Brevier für Führungskräfte*, 1995 (Paul Haupt, Bern).
[2] Verein Deutscher Ingenieure. VDI 2206: Design methodology for mechatronic systems, 2004 (Beuth Verlag, Berlin).
[3] Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., and Shaham-Gafni, Y. Model traceability. *IBM SYSTEMS JOURNAL*, 2006, 45(3), 515–526.
[4] Sutinen, K., Gustafsson, G., and Malmqvist, J. Computer Support for Requirements Management in an international Product Development Project. In *Design Engineering and Technical Conferences and Computers and Information in Engineering Conference, DETC’04,* Salt Lake City, September 2004, pp. 1-12.
[5] Ramesh, B. Implementing Requirements Traceability. *Cutter IT Journal*, 2000, 13 (5).
[6] Sutinen, K., Almefelt, L., and Malmqvist, J. Implementation of requirements traceability in systems engineering tools. In *Produktmodeller 2000*, Linköping, November 2000, pp. 313-330.
[7] Stark, R., Beier, G., Figge, A., Wöhler, T. Cross-Domain Dependency Modelling - How to achieve consistent System Models with Tool Support. In *European System Engineering Conference, EUSEC 2010*, Stockholm, May 2010, pp. 1-14.
[8] Lindemann, U. *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden*, 2007 (Springer-Verlag, Berlin, Heidelberg).
[9] Eigner, M. and Stelzer, R. *Product Lifecycle Management. Ein Leitfaden für Product Development und Life Cycle Management*, 2009 (Springer-Verlag, Berlin, Heidelberg).
[10] Maurer, M. S. *Structural Awareness in Complex Product Design*, 2007 (Dissertation, Technische Universität München).
[11] TESEON GmbH. *Loomeo*. http://www.teseon.com. Accessed December 20th 2010.
[12] Keller, R., Eger, T., Eckert, C. M., and Clarkson, P. J. Visualising Change Propagation. In *International Conference on Engineering Design, ICED’05*, Melbourne, August 2005, pp. 1-12.
[13] ID-Systems GmbH. *METUS - The System Design Method*. http://www.id-consult.com/metus-software/metus-methodik/. Accessed January 10th 2011.
[14] Tretow, G., Göpfert, J. and Heese, C. In sieben Schritten systematisch entwickeln. *CAD-CAM Report*, 2008, 8, 36–39.
[15] van Gorp, P., Altheide, F., and Janssens, D. Traceability and Fine-Grained Constraints in Interactive Inconsistency Management. In *Second ECMDA Traceability Workshop, ECMDA-TR 2006*, Bilbao, 2006, 1-13.
[16] Geensoft. *Reqtify. Effective Solution for Managing Requirements Traceability and Impact Analysis across Hardware and Software Projects Lifecycle*. http://www.geensoft.com/en/article/reqtify. Accessed December 17 th 2010.
[17] Allen, T.F.: *A Summary of the Principles of Hierarchy Theory.* http://www.isss.org/ hierarchy.htm. Accessed April 19th 2011.

Contact: Asmus Figge
Technische Universität Berlin
Chair of Industrial Information Technology
Pascalstr. 8-9, 10587 Berlin
Germany
Tel.: Int +49 30 314 26290
Fax: Int +49 30 393 0246
Email: asmus.figge@tu-berlin.de
URL: http://www.iit.tu-berlin.de

Prof. Dr.-Ing. Rainer Stark is head of the Division Virtual Product Creation and director of the Chair for Industrial Information Technology at the Technische Universität Berlin since 02/2008. After his studies in mechanical engineering (Ruhr University Bochum and Texas A&M University) he received the Dr.-Ing. degree from the Universität des Saarlandes Saarbrücken. During his industrial activities of many years he worked in different leading positions in the automotive industry. His research topics are the intuitive and context-related information modelling, intuitively usable and functionally experienceable virtual prototypes, the function-oriented Virtual Product Creation as well as development processes and methodologies for the product modelling.

Asmus Figge studied mechanical engineering with a specialisation in engineering design at Technische Universität Dresden. Since 2008 he works as a research fellow at the Chair for Industrial Information Technology at the Technische Universität Berlin. His research interests include traceability in systems engineering.