10TH INTERNATIONAL WORKSHOP ON INTEGRATED DESIGN ENGINEERING
IDE WORKSHOP | 10.-12. SEPTEMBER 2014 | GOMMERN

TRACEABILITY – A FACTOR OF INTEGRATION AND A METHOD TO DEAL WITH COMPLEXITY

Neven Pavković[1], Tomislav Martinec[1], Mario Štorga[1]

[1]University of Zagreb, Croatia

Keywords: Traceability, Visualization of Networks of Dependencies, Complexity, MDM

ABSTRACT

The paper aims to summarize several important issues in researching of modelling and implementation of traceability frameworks in design engineering area. These issues are mainly focused to methods of relationships generation and to visualization methods and techniques. We argue that a well defined and established traceability framework could be an integration factor in engineering design environments, primarily through improvement of design communication and information flow. Secondly, through efficient visualization and browsing mechanisms, we propose how a traceability framework could be based on existing matrix methods developed to deal with complexity. An extended Multiple Domain Matrix (MDM) is proposed combined with general diagramming tool, IBIS tool and tool for linking files (documents).

1 INTRODUCTION

The increased complexity of product development process, especially in large-scale projects, generates situations with which existing tools and methods are not able to deal with. Huge networks of complex dependencies and design communication in large teams are very difficult to be managed [KNV14]. The aim of this paper is to propose an approach where an implementation of traceability could significantly contribute to:

- dealing with complexity through efficient visualization and browsing methods and tools for large networks of dependencies and
- overcoming current problems in design product development process integration through improving the quality of design communication.

Traceability should enable understanding the semantic relationships that exist within and across life cycle of information objects containing information fragments about requirements, concept explanation, design details, component description, production specification or maintaining procedures. These semantic relationships could help engineering designers to understand the existing information and reuse them in the right context. Research literature describes the impact of poor traceability practices on project efficiency. A decrease in system quality, increase in the number of changes, loss of knowledge due to turnover, erroneous decisions, misunderstandings, and miscommunication are some of the common problems that arise due to lack of or insufficient traceability of engineering information [HK07].

Based on our previous research on situations that occur in medium and large scaled projects in industry, we distinguish two main directions of traceability:

1. Looking forward—guiding: where traceability process is planned and organized, followed by assigning identification to information objects, activities, participants, locations, and resources, and exchanging it among participants. Here the participants should find the answers, e.g., the overview of design process, the knowledge about information needs, the availability of information and documentation, and most important, the relationships (linkages) between all identified items.

Especially in complex products implemented traceability model should be able to provide the answers like: what objects, parameters, etc. are affected if a particular change is to be made - who are the persons responsible for those objects and parameters, etc.

2. Backtracking—management of the design history should allow participants to follow the evolution of design items from its origins, through its development and specification, to its deployment and realization, and through periods of ongoing refinement and iteration in any of these phases. Also, tracing of the design history should improve understanding of the design routes by linking designed items to justifications, important decisions, and the assumptions behind them. By tracing designed items back to their sources, the impacts of later changes in any product feature can be identified before a product is redesigned.

We argue that an implementation of traceability in engineering design frameworks could significantly contribute to the quality of design communication. Creation of new channels of communication may be also viewed as a facilitation of design engineering integration. This may be valid for all levels of communication interfaces: designer to designer, multidisciplinary team, team and company (organization), and interfaces of collaboration in an innovation network.

2 RELATED WORK

2.1 Software traceability

Traceability in software engineering has got more attention of researchers than in engineering design. Several models and methodologies were developed, mainly focused on requirements traceability and related issues – [MXC08], [RJ01]. An example of comprehensive research projects in this area is the "MOST" project (http://most-project.eu). Schwarz et al. [SEW10] present the approach that supports the definition of metamodels for traceability information, recording of traceability information in graph-based repositories, identification and maintenance of traceability relationships using transformations, as well as retrieval and utilization of traceability information using a graph query language. A roadmap of research and practices related to software traceability together with open issues is presented in Spanoudakis [SZ05]. This paper summarizes research work in area of software traceability and presents a very useful discussion on manual, semi-automatic and automatic generation of traceability relations.

2.2 Visualization

Efficient visualization (and manipulation) of large networks of relations is arguably the primary condition for successful implementation of traceability in industrial practice.

Diagrams augment cognition [SEW08]. As such, a good diagram augments the capacity of the diagram's user to achieve goals. Visualization literally "makes visible" (or "evident") things that might not otherwise be so [SEW08] - authors made a review of existing diagramming tools and they concluded that:

- Simplicity is important. The simpler the tool – even though its scope may be limited as a result – the easier it is to use, and the more likely users are to adopt it willingly and "naturally."
- Network hypergraphs are essential. The richly interrelated information elements typical in early designing are highly coupled, and representing those relationships is essential.
- Diagram layout is essential. A proper layout for a diagram can actually simplify it without loss of semantics.

Based on their findings the authors argue that there is no existent tool fully suitable to engineering design support purposes and that a new framework for diagramming tools must be developed. By making information structures organized, modern visualisations provide means for user to interactively navigate and uncover the information engineers are looking for [KT05]. It is presumed that the user is often being unaware of the precise information location by which the information can be obtained or possesses incomplete specification relating the information necessary to perform

search. Both of the latter could be the cases in the product development of the complex technical systems involving large data and information sets and multitude of stakeholders generating and interpreting information. In [MP14] we argue that diagrams are convenient for both fast recording and retrieving of particular tracing context on design episode level, and consider diagram networks as the basis of well-established traceability on project level. A computer-based diagramming tool was used to test the methodology. It features basic node-link creation, formatting and arrangement, predefined IBIS nodes, image import, hyperlink embedding, ontology support and search mechanisms.

3 MODELS AND METHODS FOR ESTABLISHING TRACEABILITY

From current research results it could be concluded that the achievement of engineering information traceability in modern, highly automated product development environments is still very difficult. There are many reasons for that. The current engineering design environments could not be supportive of traceability procedures because people communicate and exchange engineering information across organizational and discipline boundaries, so they reuse existing information in new and unpredictable contexts and often information is translated from one format to another, during which information loss occurs. Those facts make the development of suitable and efficient models and methods for establishing and supporting traceability very complex and challenging.

Several current research projects are focused on the development of an integrated product and process approach supporting the modelling of traceability in order to handle today's rising complexity eg. [KNV14] and [CWW14]. In [KNV14] authors argue that it is necessary to include sociotechnical meta-model. Cycle-oriented traceability based on well defined templates of particular subprocesses is proposed in [CWW14].

Generally traceability could be viewed as a generation of a network of relations between various engineering objects (EO) where objects are considered as documents (or "information carriers"), abstract notions from various domains (e.g. functions, requirements, changes, design tasks), "physical" objects like elements of product structure (components) and finally employees. Based on research findings focused to current traceability practice in industry it is arguably obvious that it is impossible and unnecessary to establish a "full network" of all existent traceability relations, because of huge number of EOs that exists in any sociotechnical system on levels of granularity that could satisfy practical needs. Therefore it is necessary to focus the further research to models and methods that will primarily be able to detect and manage a subset of beneficial relations for practical needs, both for guiding and backtracking.

According to [SZ05], despite the wide recognition of its importance and numerous years of research, effective traceability is still rarely established in contemporary industrial settings. It is very difficult to automate the generation of traceability relations with clear and precise semantics that could, adequately and cost-effectively, support the types of analysis necessary to deliver the benefits of traceability. Spanoudakis and Zisman [ZS05] emphasize that most of the existing approaches, environments and tools assume either that traceability relations should be identified manually or offer traceability generation techniques which cannot identify relations with a rich semantic meaning. In the former case, the cost of identifying traceability relations manually clearly outweighs the expected benefits of traceability and makes organisations reluctant to enforce them, unless there is a regulatory reason for doing so. In the latter case, the lack of a clear and precise semantics make the asserted relations of little use and do not provide the benefits of using traceability as described above. Therefore, the relevant techniques are not widely adopted in industrial settings.

Manual creation of traceability relations is difficult, error-prone, time consuming and complex, [SZ05], [KNV14], [MSB11a]. Therefore a compromise must be found which will provide satisfactory level of traceability functionality (benefits) to engineers, but at the same time which will not require significant additional efforts to be developed, implemented and managed. Mainly in the area of software traceability, several approaches which support automatic or semi-automatic generation of traceability relations have been proposed [SZ05].

In survey written by Spanoudakis and Zisman [ZS05] the authors organise the semi-automatic traceability generation approaches into two groups: (a) *pre-defined link group*, that is concerned with the approaches in which traceability relations are generated based on some previous user-defined links, and (b) *process-driven group*, that is concerned with the approaches in which traceability relations are generated as a result of the software development process. Proposals of approaches to support automatic generation of traceability relations use information retrieval (IR) techniques, traceability rules, special integrators, and inference axioms.

At this point a main research question emerges:

Which kind of traceability model framework would enable a cost effective and beneficiary implementation of automated and semi-automated generation of traceability relations?

All previously listed research findings and our own experiments made in [MSB11b] directed us towards the idea (proposal) of developing of a "hybrid" model of traceability framework that will comprise and integrate various approaches and methods. The intention is to use the most appropriate method(s) for each identified issue – e.g. relation generation, network visualization, template generation, modelling of processes and their cycles, etc., always from the primary viewpoint of reducing the efforts required in practical industry application.

Further idea is to identify and classify most common (and important) traceability problems and issues in engineering design practice, and for each of them to find and develop a focused (specialised) approach and/or method of traceability relations generation and visualization.

In such an approach firstly we could distinguish traceability relations and EOs from the dynamic point of view. Product structure and/or product architecture (or at least their elements) could be considered as relatively static data structures (on higher levels of granularity) for majority of engineering design environments. For example in automotive industry there is a high extent of mechatronic systems' reuse [KNV14]. Product structures for complex products could contain large sets of EOs and relations (especially for mechatronic systems). These structures (at least subassemblies and/or modules) do not change significantly over time, (on higher levels of granularity), therefore we assume that it could be cost-effective to build a template structure for them in form of diagrams. Such an approach could be considered as a semi automated method, because engineers would reuse and update templates while generating sets of relations.

Generally, at the highest level of abstraction, traceability relations can be classified as relations between objects of the same domain and between objects from different domains.

Consequently we assume that the majority of the relations between different domains have a more dynamic character, but probably smaller sets of EOs will have to be linked. For such situations manual generation of relations and matrices as visualization method instead of diagrams seems to be more appropriate. There are many assumptions here that still have to be validated – this line of reasoning is mostly based on previous research findings presented in [PBF11] and [PTS12].

Design rationale may be viewed as traceability of design thinking and the decision process. We argue that a design rationale capturing method have to be an element of traceability framework. We consider that IBIS (Issue Based Information Systems) based diagrams proved to be presumably the most appropriate design rationale capturing method [AB13].

Finally, how those various approaches could be integrated and/or merged? Our proposal is to use an extended model of Multiple Domain Matrix (MDM) as the basic framework and a starting/basic interface. Firstly we will describe a developed prototype tool for building a network of interlinked diagrams, and then a proposal of extended MDM will follow.

3.1 Network of diagrams as one of the methods for establishing traceability

This chapter describes our research work [MP14] on establishing engineering information traceability using diagram tools as means of information and relation generation and recording. Information

displayed in diagrams is structured through the concept of nodes and links between the nodes. Every diagram node is an information container, which can include information about digital entities storage, displayed as hyperlinks to computer stored files. There is no limit in terms of file types that can be linked (CAD, spreadsheets, text documents...), including other diagrams. Adding links between diagram files creates a diagram network. Such a network allows users to cross boundaries of a single record and browse information spread in multiple design episodes.

A prototype of computer-based diagramming tool was built and used to test the methodology. It features basic node-link creation, formatting and arrangement, predefined IBIS nodes, image import, hyperlink embedding, ontology support and search mechanisms.

Several types of diagrams were introduced throughout the methodology and diagramming tool implementation on the ongoing project. These diagrams cover communication visualization, product structure and specification, and design rationale. Traceability relations between computer files is very important part of traceability framework, because files of any type are “carries” of product information- they represent generated product documentation. In [MP14] we proposed a methodology and interface for manual generation of relations between files. The visualization of file system content interrelations is realized in both diagram (graph) and matrix form. The network of interrelated files is created through an explorer-like interface, where one can establish and record relationships between selected explorer items (Explorer Tool on Figure 1). File browsers enable navigation through computer (server) content, and thus serve as Windows Explorer substitute. File system content can also be displayed as a matrix, where rows and columns represent the content of two or more different file system folders. Relationships can furthermore be visualized either manually by exporting node-edge files, or automatically with the developed diagram network visualization tool.

The development of the project explorer environment was started mainly to integrate diagrams into project documentation, but the application was further upgraded with other useful features and is still in development phase. Two main objectives were set at the start of the development:

- Allow users to manually link diagrams with computer-stored files and display these links in the explorer interface
- Facilitate diagram creation with templates since the tested diagramming tool doesn’t support template importing

New development objectives were additionally set, including:

- File to file (or directory) linking, using the same principle as in diagram to file linking
- File enrichment using attributes
- File status association and status display in the explorer interface
- Automatic visualization of created links in an interactive diagram form

The environment is conceived as a central tool for the creation of diagram networks. The diagramming tool, now a part of the environment, is supported with automated diagram storage and template selection. Three main tools were developed within the environment (Figure 1):

- **Explorer Tool** - Serves as the file explorer. User can browse the computer/server file system and create relations between computer-stored files and folders. The Explorer Tool also handles documents statuses, ontologies and diagram templating. It also drives the diagramming and visualization tools. File icons in the explorer are automatically modified depending on whether the files are linked or associated with a status.
- **Manual Diagramming Tool** - Used to manually create diagrams such as Issue Based Information System (IBIS), system architecture and function breakdown structure diagrams. Diagrams can be created either from scratch or from prepared templates. The tool supports different node types, customization, hyperlinks and image placement.
- **Visualization Tool** - Visualizes all established traceability links. The tool was developed to automatically generate diagram networks for the file selected in the Explorer Tool. Each file, diagram, ontology element or directory that is in any way linked with the selected file is

represented in the form of a diagram node. Traceability links between files are represented as diagram links.

Although the creation of relationships in-between the content of the file system can result in a well-established traceability of project documentation, it is limited to a single domain - computer-stored files. In order to manage complex engineering data it is required to cover and trace elements from multiple domains.

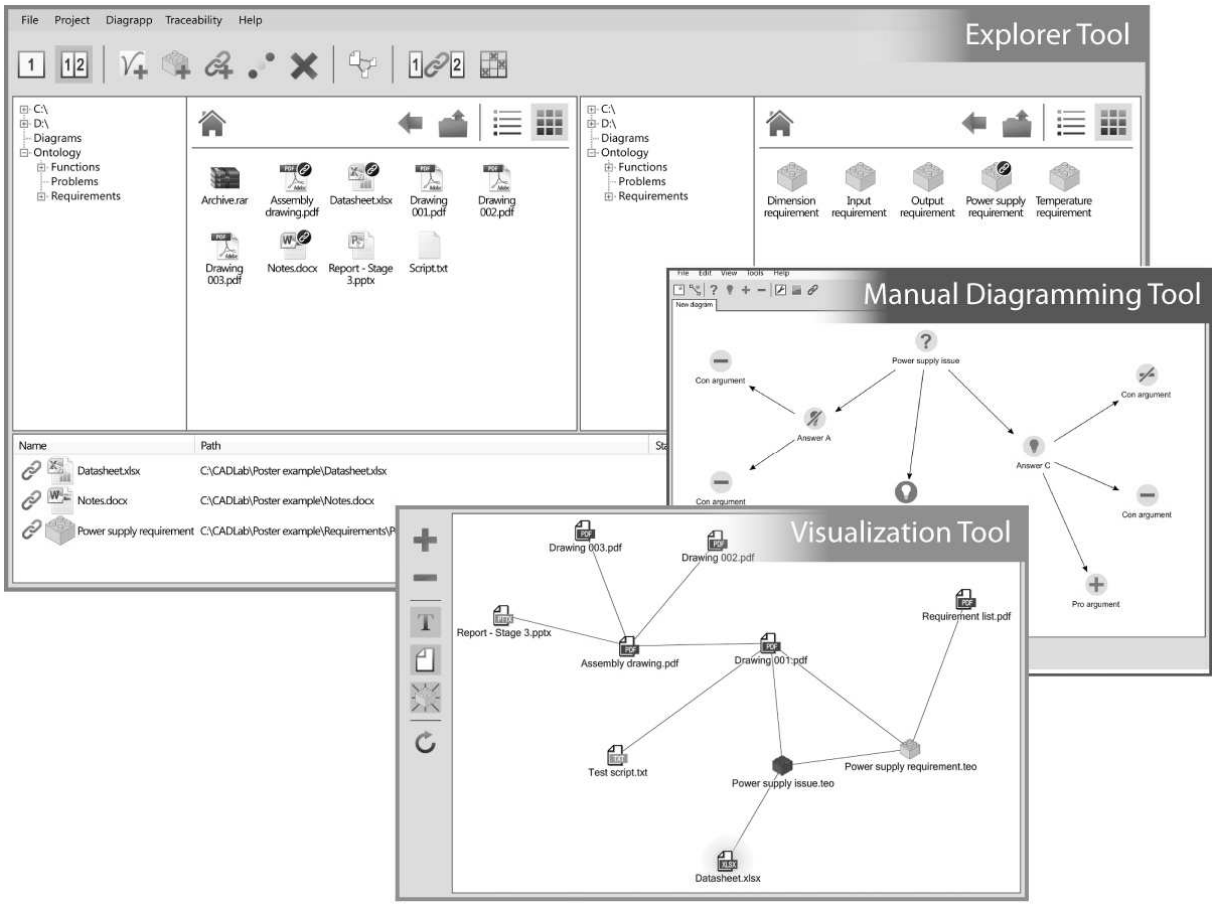

Figure 1: Components of tool for file (documentation) linking and for diagram network manipulation

3.2 Extended Multiple Domain Matrix as the basis for traceability framework

Several methodologies exist for dealing with data complexity in product design, including the application of graph theory and matrix-based approaches [LMB09]. Since the matrix-based approaches to complexity management are widely applied, we decided to use them as the basic architecture of the traceability framework. Nowadays, the large variety of matrix-based methods in engineering can be classified by the quantity of the types of elements involved. Whereas some approaches focus on the representation and analysis in between elements of the same type (e.g., dependencies within product components), others consider linkages between two types (e.g., dependencies between customer requirements and product functions) [M07]. According to [LMB09] there are four types of general matrix systems. If relations within elements belonging to the same type (domain) are examined, the related matrices can be defined as intra-domain. A commonly applied approach of an intra-domain matrix is the Dependency Structure Matrix (DSM). Relationships between file system content in our research were mapped and stored in form of a square intra-domain matrix. Matrices combining different elements belonging to different domains are referred to as inter-domain matrices. For example, components and functions of a product can be considered as elements belonging to two different domains [LMB09]. Some applications make use of combinations of intra- and inter-domain matrices, while some further include computations of some subsets by

information stored in other subsets. Such an approach is called the Multiple-Domain Matrix (MDM) [LMB09].

MDM is a square matrix comparable to a DSM containing system elements in identical order on both axes. In contrast to a DSM, different types of system elements are included and grouped in domains; the MDM can be subdivided into DSMs and DMMs (Domain Mapping Matrices) according to the inherent domains. The MDM possesses features of a common DSM; in fact, it represents a DSM on a higher level of abstraction: If the domains are considered as single elements, the areas of the DMM subsets represent the matrix cells that can store dependencies between these elements. Applying this logic, the areas of the DSM subsets are located on the matrix diagonal and can represent self-reflexive dependencies [LMB09].

To further extend our proposed traceability framework, we need to establish relations between engineering information stored as documentation in files with engineering objects (EOs) from other different domains. Of course it is also necessary not only to relate documents and EOs, it is equally important to establish and record relation between EOs. A schematic view of such approach is presented on Figure 2. EOs from different domains are represented with different symbols and colours, while relations are represented with different types of lines, similarly as in [LMB09].

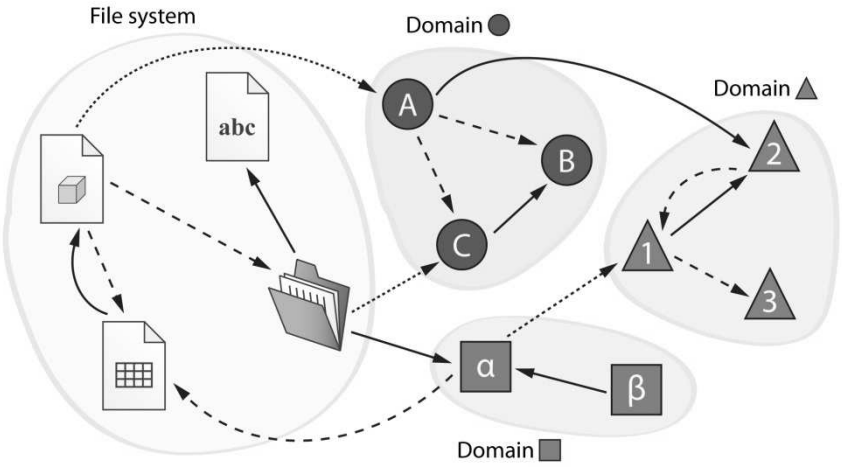

Figure 2: Network of engineering objects from different domains

Figure 3 is a matrix representation of diagram shown on Figure 2, where each relation is denoted with a mark in corresponding matrix cell. This is the Multiple Domain Matrix (MDM) as it is proposed in [LMB09] and in other relevant literature.

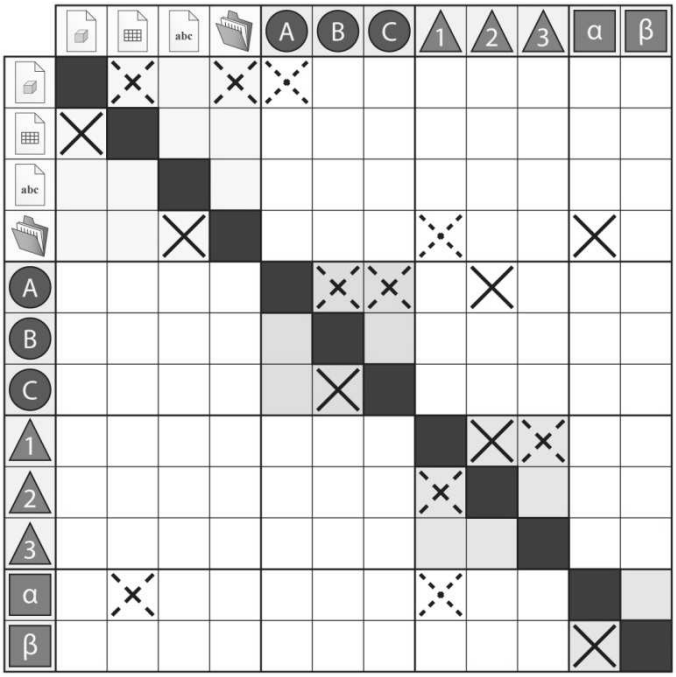

Figure 3: Representation of engineering objects and their relations in a MDM

A similar approach of MDM usage, focused to creation of knowledge maps of employees is proposed in [WSS13].

However, we think that for the purposes of traceability modelling, it is necessary to further extend the MDM model and especially the process of manipulation with matrix, due to several reasons:

- Huge number of traceability relations in any kind of industrial application will generate huge matrices, impossible to be manipulated and viewed as a whole – procedures and tools have to be developed that will enable hiding unnecessary areas and/or extracting and visualizing areas of current interest.
- Semantics of relations should be added, because this is very important in traceability. Additionally it would be beneficial if a cell would contain (or point to) more contents than just a mark of relation existence.
- Mechanisms (procedures) for generating and inserting predefined templates of selected matrix areas should be developed and implemented.

We argue that such an extended MDM model could open the further opportunities for development and implementation of semi-automatic generation of traceability relations. Also, with efficient mechanisms for manipulation of huge matrix, the matrix itself could serve as the basic interface for majority of operations in traceability framework.

An initial proposal of semantics of relations between a set of crucial domains for engineering design traceability is shown on Figure 4. We don't consider this set of domains as final, any particular design environment could build and adapt domains and relations according to its own needs.

	Documents (Files)	Requirements	Functions	Components	People	Design problem (Task)	...
Documents (Files)	Traceability of information fragments	Files documenting requirements	Files documenting functions	Files defining components	Organigrams, Workflows	Files contain design rationale	
Requirements		Requirement relations				Requirements impose design problems	
Functions		Functions correspond with requirements	Function Analysis Diagram (FAD)			Functions impose design problems	
Components		Components realize requirements	Components realize functions	Product structure		Component design problem	
People	People responsible for documents			People responsible for components	Team-based DSM	People working on problems (tasks)	
Design problem (Task)	Problems include files	Problems include requirements	Problems include functions	Problems include components	Problems include people	Task-based DSM	
...							

Figure 4: A proposal of crucial domain relations in design traceability framework

First step in adding relation semantics to MDM model could be a classification of traceability relations - a very good general proposal based on overview of several approaches could be found in [SZ05]. Thus, a class of relation could be indicated with a code e.g. "R2", as shown on Figure 5. Furthermore we think that in many cases would be beneficial if an additional content could be linked to each matrix cell. That may be comments, hyperlinks, etc. – that way a cell could be "expandable" (Figure 5.) pointing to any kind of information that may be of use for more detailed explanation of particular relationship. In such an approach we plan to treat a matrix cell as an information container, combined with a symbol that indicates generated (recorded) relation. A symbol (or its first digit) may be used for already developed matrix calculations.

Another approach to semi-automated generation of traceability relationships is to develop a templates of subprocess (scenarios) that could generate and/or record the relationships in matrix cells when a pre-planned situation (event) is triggered. The appropriate candidates may be the processes with cyclic character. Chucolowski et al. developed a data model and described a process sequence for traceability in engineering change management [CWW14]. Such processes should be focused on one particular area of MDM and should be precisely defined and modelled according to instances of MDM domains.

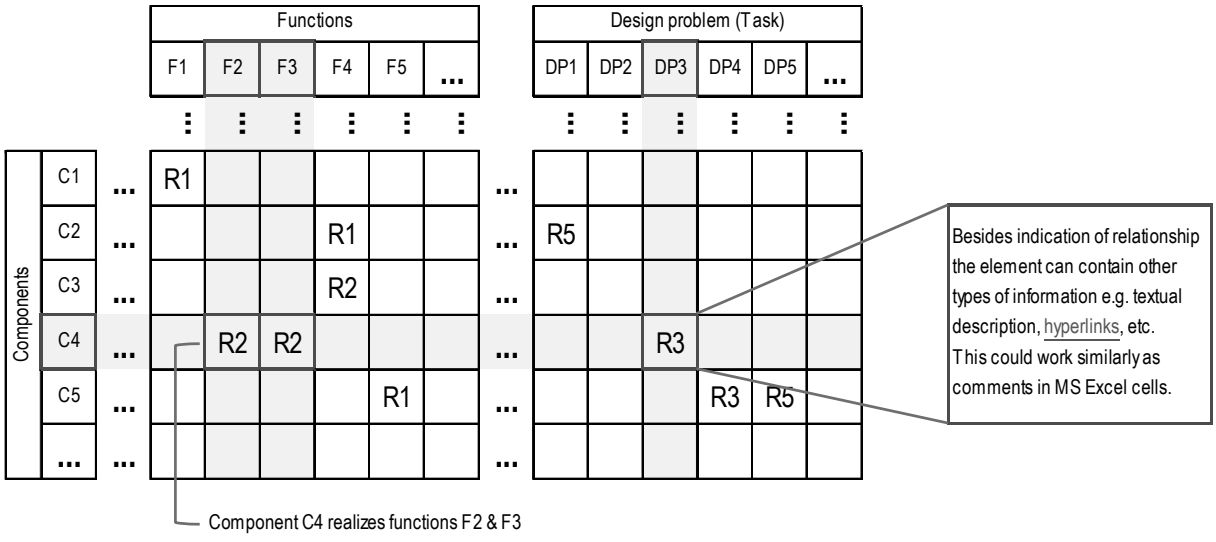

Figure 5: Extended semantics of MDM cells

The most important issue that has to be resolved for potential implementation of proposed MDM-based traceability framework in industrial practice is the manipulation with huge matrix. The interface and the visualization capabilities of the tool that will manage the huge matrix have to provide the following mechanisms (Figure 6):

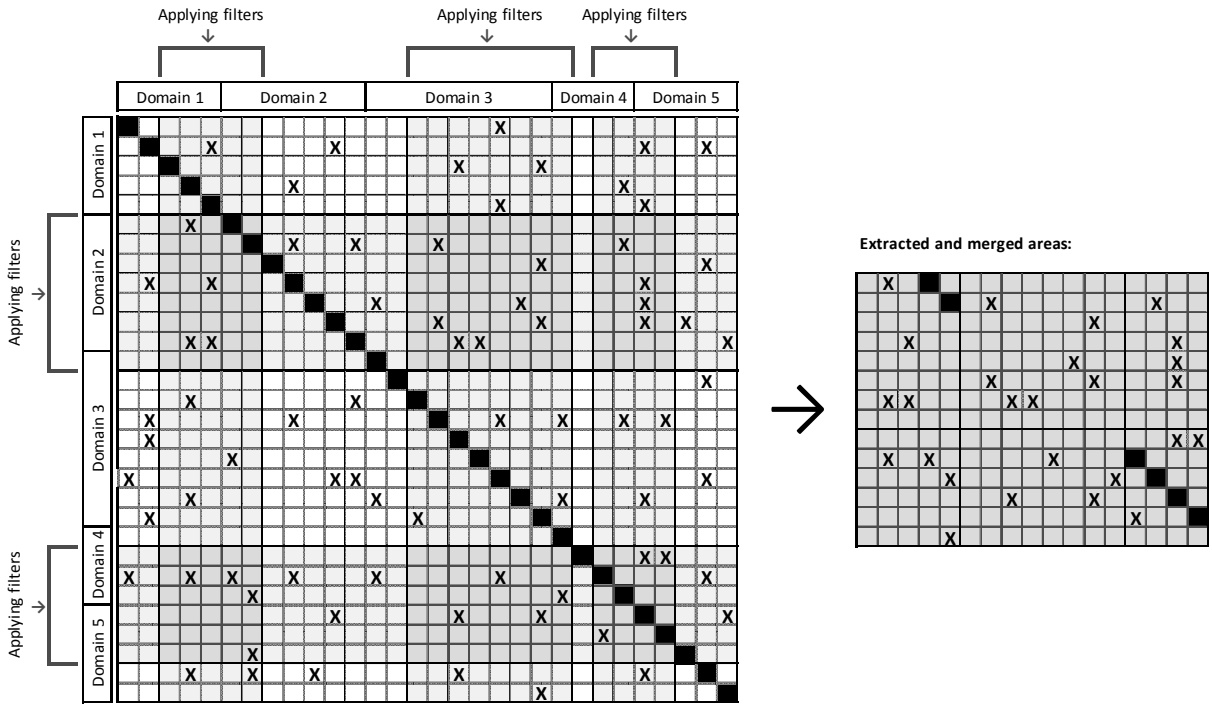

Figure 6: Reducing the "working space" on matrix to filtered - extracted areas

- Filtering on level of domains, and on level of rows and columns, enabling to hide/extract a set (combination) of rows and/or columns belonging to different domains or extracting several full domains. Applying of filters should enable the user to extract and merge the areas of matrix that are of his current interest while working on matrix data. The extracted area should keep all the indicators of domains and particular rows and columns as they are visible on the whole matrix. Here by extracting we mean only visual extraction – the rest of the matrix is just being hidden.
- Extracting only the cells that have a symbol of relation from the set of selected (filtered) rows and columns (or domains).
- Efficient way of inserting/updating areas that are built and stored outside of the "main" matrix as predefined templates.
- Domain names and their instances (EOs) as well as the relationships should be based on specially developed ontology as proposed in [PSBM13] and [SMS11].
- Layering / colouring schemes may also be beneficial in particular manipulation situations.

As equally important manipulation issue - the timeline should also be carefully considered – for which period should one MDM be valid? Should one MDM be valid for e.g. one big project or some areas will have permanent character while the others should represent certain periods in timeline? How to combine areas of matrices and/or whole matrices from different projects and time periods?

4 SUMMARY AND FURTHER RESEARCH ISSUES

This section will further elaborate how an established traceability in particular engineering environment could become a factor of integration as well as the method (instrument) to deal with complexity. Situations and/or requirements that trigger utilization and deployment of recorded traceability data varies across engineering domains (e.g. software, automotive industry, mechatronic systems), but also the significant part of them are common to all domains.

According to [SZ05] traceability relations may be deployed in the development life cycle of a software system to support different development and maintenance activities, including:

- *change impact analysis and management*;
- system verification, validation, testing and standards compliance analysis;
- *the reuse* of software artefacts; and
- software *artefacts understanding.*

Based on an analysis of the project management processes and findings gathered in [MSB11b], we have extracted major traceability issues and requirements relevant for project management in one medium sized automotive company:

- Which *documents are associated* with one particular context or viewpoint?
- What is the completeness and accuracy of document content involved in a *particular project milestone*?
- Are all documents and information correctly and completely transferred from one main business process to another *("handover" scenarios between different teams*)?
- *What were the major business changes in the project portfolio*, when and why did they happen, and how did they influence currently active projects?

Koehler [KNV14] and Chucolowski [CWW14] also emphasize the *change impact analysis*, *change management* and *understanding of complex (mechatronic) systems* when they have to be reused (their research is also focused to automotive industry).

According to [SZ05] the simplest form of analysing the impact of a change in a given artefact (e.g. a requirement statement) is the identification of all the other artefacts that will be affected by the change (e.g. design artefacts and software code). Primitive change impact analysis requires the provision of basic querying facilities to retrieve traceability relations of specific types that may also have specific values for the properties defined for these types. Most of the existing traceability tools and environments provide such querying facilities (in the area of software engineering).

Spanoudakis and Zisman [SZ05] also state that more complex forms of change impact analysis may also be desired in different settings. Examples of these forms are: (a) the classification of affected artefacts into different groups subject to the exact effect that the change will have on them, (b) the identification of side-effects that the change may have, and (c) the estimation of the cost of propagating the change. The delivery of such capabilities requires support for the composition of different traceability relations into *tracepaths*. These trace-paths can demonstrate how impact is propagated across artefacts that are not directly related.

We believe that the MDM – based traceability model and framework could provide a good basis for further development of algorithms that will realize above mentioned requirements and especially visualization functionalities – where tracepaths will be shown as diagrams "extracted" from relevant matrix areas.

Sherba et al. [SAF03] have proposed an approach that allows the generation of new traceability relations based on relationship chaining. This approach uses special integrators, which can discover and create traceability relations between software artefacts and other previously defined relations. The new identified relations can be generated based on indirect and transitivity dependencies, complex dependencies containing more than one source or destination elements being related.

Proposed MDM - based traceability framework should further contribute to two important factors that influence design communication: *an awareness of what information the other party needs* and an *overview of the sequence of tasks in the design process* [MKH08].

Besides bridging the gaps in information flow (described in [ECS01]), the proposed traceability methodology should offer the possibilities to *integrate knowledge* toward the creation of shared understanding in collaborative product development teams [KBV10]. Based on the proposed approach to defining domains and EOs as elements/subsets of ontology, the knowledge integration could be accomplished in two ways:

- using the existing relations in ontology to navigate (perform semantic searches) between related elements of several domains;
- establishing new relationships (either compositional or associative) between elements of different domains that were not recorded manually.

To conclude, all issues listed in this section are actually open research issues that require further intensive efforts from engineering design community.

REFERENCES

[AB13] *Aurisicchio, M. & Bracewell, R.H. (2013). Capturing an integrated design information space with a diagram based approach, Journal of Engineering Design, Vol. 24, Issue. 6, pp. 397-428.*

[SZ05] *Spanoudakis, G., Zisman, A., "Software traceability: a roadmap", Handbook of Software Engineering and Knowledge Engineering, Vol.3, 2005, pp. 395-428.*

[CWW14] *Chucholowski N., Wolfenstetter T., Wickel M.C., Krcmar H., Lindemann U., Towards Cycle-Oriented Traceability In Engineering Change Management, Proceedings of the 13th International Design Conference DESIGN 2014, Dubrovnik, Croatia 2014, pp, pp. 1491 – 1500*

[KNV14] *Koehler N., Naumann T., Vajna S., Supporting the Modeling of Traceability Information, Proceedings of the 13th International Design Conference DESIGN 2014, Dubrovnik, Croatia, 2014. pp. 1811 – 1820.*

[HK07] *Hurwitz, J., & Kaufman, M. (2007). Leveraging information for innovation and competitive advantage. Hurwitz & Associates White Paper, www.hurwitz.com.*

[ECS01] *Eckert, C., Clarkson, P.J., & Stacey, M. (2001). Information flow in engineering companies: problems and their causes. Proceedings of ICED01, Glasgow.*

[KT05] *Keller, T., Tergan, S.O., (Eds), Visualizing Knowledge and Information: An Introduction, Lecture Notes in Computer Science: Knowledge and Information Visualization, Springer 2005.*

[KBV10] *Kleinsmann, M., Buijs, J., & Valkenburg, R. (2010). Understanding the complexity of knowledge integration in collaborative new product development teams: a case study. Journal of Engineering and Technology Management 27, 20–32.*

[LMB09] *Lindemann U., Maurer M., Braun T., (2009). Structural Complexity Management, Springer 2009.*

[MKH08] *Maier, A.M., Kreimeyer, M., Hepperle, C., Eckert, C.M., Lindemann, U., & Clarkson, P.J. (2008), Exploration of correlations between factors influencing communication in complex product development. Concurrent Engineering: Research and Applications 16, 37–59*

[MSB11a] *Marjanovic, D., Storga, M., Bojčetic, N., Pavkovic, N., Stankovic, T. (2011a). EUREKA E4911 TRENIN Project Final Report, www.trenin.org.*

[MSB11b] *Marjanovic, D., Storga, M., Bojčetic, N., Pavkovic, N., Stankovic, T. (2011b). EUREKA E4911 TRENIN Report on Stakeholder Perspectives on Traceability, www.trenin.org.*

[PBF11] *Pavkovic, N., Bojcetic, N., Franic, L. & Marjanovic, D. (2011). Case studies to explore indexing issues in product design traceability framework. Proc. of the ICED 11 Int. Conf. on Eng. Design 6, 131–140. Copenhagen.*

[MP14] *Martinec T., Pavković N., Visualization of Information Traceability In Product Development, Proceedings of the 13th International Design Conference DESIGN 2014, Dubrovnik, Croatia 2014, pp. 1831 – 1842.*

[SEW08] *Salustri, F.A., Eng, N.L., Weerasinghe, J.S.(2008). Visualizing Information in the Early Stages of Engineering Design. Computer-Aided Design & Applications, 5(1-4).*

[SEW10] *Schwarz, H., Ebert, J., & Winter, A. (2010). Graph-based traceability: a comprehensive approach. Software Systems Modelling, Vol. 9, No 4, 473-492.*

[SAF03] *Sherba S.A., Anderson K.M., Faisal M., "A Framework for mapping Traceability Relationships", proceedings of the 2nd International Workshop on Traceability for Emerging forms of Software Engineering (TEFSE 2003), Montreal, September, 2003.*

[MXC08] *Mohan, K., Xu, P., Cao, L., & Ramesh, B. (2008). Improving change management in software development: Integrating traceability and software configuration management. Decision Support Systems, 45, 922–936*

[RJ01] *Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. IEEE Transastions on Software Engineering. 27(1), 58–93.*

[WSS13] *Wickel, M. C., Schenkl, S. A., Schmidt, D. M., Hense, J., Mandl, H., Maurer, M., Knowledge structure maps based on Multiple Domain Matrices, InImpact: The Journal of Innovation Impact, Vol. 5 No1, 2013*

[PSBM13] *Pavković, N., Štorga, M., Bojčetić, N. & Marjanović, D. (2013). Facilitating Design Communication Through Engineering Information Traceability, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Volume 27, Special Issue 02, pp. 105-119.*

[PTS12] *Pavković, N., Tečec Ribarić, Z. & Sviličić, T. (2012). Traceability Case Study on Rail Vehicle Control Unit Development Project, Proceedings of DESIGN 2012, the 12th International Design Conference, Dubrovnik, pp. 1567-1578.*

[SMS11] *Štorga, M., Marjanović, D. & Savšek, T. (2011). Reference model for traceability records implementation in engineering design environment. Proceedings of the ICED 11 International Conference on Engineering Design, Copenhagen, Vol. 6, pp. 173-182.*

[M07] *Maurer, M. S. (2007). Structural Awareness in Complex Product Design. Doctoral Dissertation. Product Development, Technical University of Munich, Munich, Germany.*

Contact:
Neven Pavković
University of Zagreb, Faculty of Mechanical Engineering & Naval Architecture
Ivana Lucica 5
10000 Zagreb
CROATIA
Phone: +385 1 6168 545
Fax: +385 1 6168 284
e-mail: neven.pavkovic@fsb.hr
URL: http://www.cadlab.fsb.h