INTERNATIONAL DESIGN CONFERENCE - DESIGN 2016
Dubrovnik - Croatia, May 16 - 19, 2016.

A TOOL FOR VISUALIZATION OF CAD MODEL COMPONENT RELATIONS USING DSM

A. Zubić, N. Bojčetić, D. Žeželj and S. Flegarić

Keywords: computer aided design (CAD), design structure matrix (DSM), visualisation, complexity

1. Introduction

With today's increased level of product complexity [Königs et al. 2012] during the product development, traceability is an approach which naturally fits into an environment where future decisions need to be made fast and precise with the broad spectrum of context surrounding the product. Knowing the decision making process and how conflicts inside issues were resolved helps with future versioning and product variants. Knowledge gathered during the development process can be reused and it helps to achieve better efficiency while shortening the time for finalizing similar future projects [Bracewell et al. 2009]. Over the last few decades, automotive industry has been facing new challenges such as significant CO2 restrictions, market drifts and economic crisis which has demanded sudden changes in project management. Economically, budgets for individual projects has been reduced, while at the same time, system complexity has arisen due to the demand for a fuel efficient vehicle. Efforts have been made to achieve fuel efficiency by looking into the green energy sources and building hybrid vehicles which, in turn, has made system complexity harder to handle because of a combination of multiple power trains. Old well established and well known methods have to be modified. Now, different fields of science are coming together and therefore new model had to be developed to manage this complexity of all newly introduced artefacts in intensified cooperation, joint venture and network structure in manufacturing [Naumann et al. 2011].

One of the methods for understanding system complexity is introduced as sociotechnical system approach. It defines basic system functions as interaction between human and machine and as communication between humans. Product development process can then be modelled based on those two functions. With this approach it is possible to model the system on lower level in contrast to what was possible before, therefore it gives better overview of the system. Methods and tools like CAx and PLM technologies had to be embraced and put into the everyday process of product development. In the end, all of this should come together in a model describing how to deal with the issues related to product development under consideration of social behaviour, design methodologies and IT services [Naumann et al. 2011]. As humans interact with machines (for example computers), which often provide us with tools for solving different scenarios, it's important that those tools are efficient and produce quality results [Naumann et al. 2011]. If produced results were not correct, it's not possible to support traceability.

There are existing methods that support traceability when dealing with product requirements which are then translated into technical functions, e.g. House of Quality. House of Quality supports traceability based on the knowledge about the relations between functions and requirements [Lindemann et al. 2009]. Requirements alone can be analysed with existing data mining and text mining techniques [Chaovalitwongse et al. 2008], however, the existing tools and methods for product development are

not capable to deal with today's complex dependencies in vehicle development [Naumann et al. 2011] therefore the effort to manually model Engineering Object Relations (EORs) is the main obstacle for using traceability tools in practice. None of the existing commercial PLM tools support the modelling of EORs to the full extent [Storga et al. 2011]. It can be concluded that there is a space for improvement of traceability in the domain of component relations in product assembly (Figure 1). The tool presented in this work is based on the Design Structure Matrix (DSM).

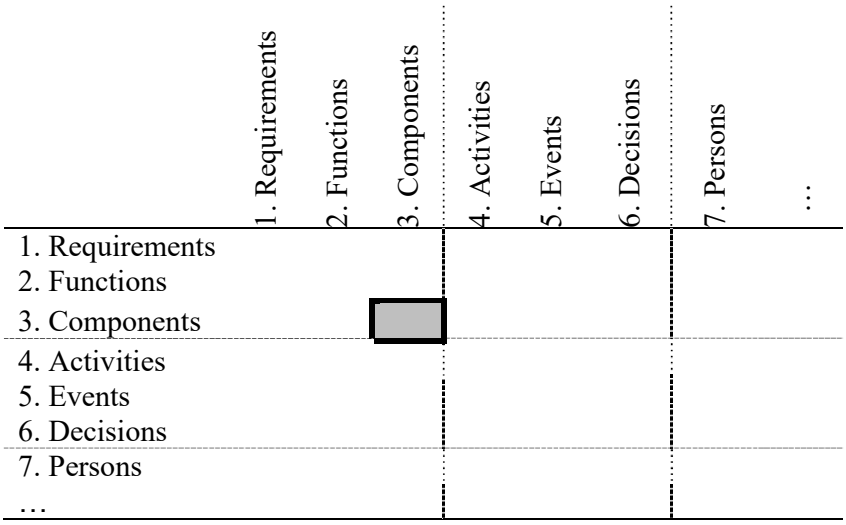

Figure 1. Traceability matrix - greyed cell represents the area of interest

DSM offers a simple way of presenting a view of an assembly component relations (Figure 2). Every DSM element can be defined with the relations to other elements of the system. That way system structure is built [Lindemann et al. 2009]. After the matrix has been created, post-processing methods allow efficient use of the matrix for analysis and further use such as search, visualization, modularization etc. For example, it is possible to detect groups of closely related components by clustering them [Eppinger and Browning 2012] and this could later lead to building one component with all the necessary functions, thus optimizing the number of system elements. DSM matrices serve as a tool to better understand system structure so it's relevant to have good data for creating the matrix.

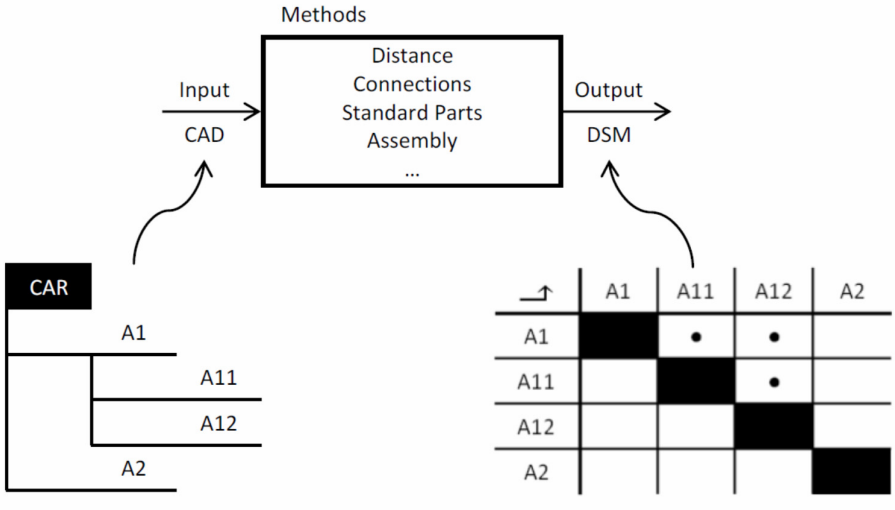

Figure 2. Developing methods for finding assembly component relations

Complete structural representation of product model with described EORs between Engineering Objects (EOs) should enable engineers to recognize which EOs of the complete system will be affected by the change of desired EO. Creating EORs manually consumes a lot of time and involves a lot of people familiar with the system design [Tarnovski 2011], thus new methods identifying the relevant parts as well as the relevant relations of complex mechatronic systems have to be developed.

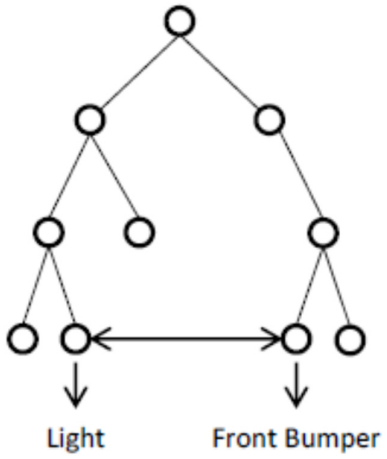

Figure 3. How are the light and front bumper related?

This article describes how to support traceability by finding new methods for automated and semi-automated extraction of EORs between given EOs (Figure 3) to eliminate manual work which is (today) needed for extracting these relations. This contribution helps to better understand product assembly structure and produces data that can consequently be used for better product management.

2. Understanding system complexity and industry practices

While developing mechatronic systems, every module of the system with a specific function should be applicable to different context of another system. This allows future development to take existing project and use it in another as a module which does its function regardless of a new context [Sanchez and Mahoney 1996]. Unfortunately, there are often new requirements that change existing modules and the new change can affect other parts of the system. While knowing which parts of the system will be affected by the change, it is easier to plan required resources to achieve well integrated solution.

A multitude of new technical dependencies emerged due to increased integration of mechatronic components in modern vehicles. Consequently, not only product complexity but also process complexity increased. The number of engineers involved in the development of complex technical systems has been rising significantly over last hundred years [Königs et al. 2012]. In 1885, Karl Benz alone built the first vehicle powered by a gasoline engine. Today, the development of a modern car involves several hundred people. Rising product complexity is directly proportional to rising complexity within the organization [Naumann et al. 2011]. Organization is responsible for partitioning complex projects in smaller ones and for re-joining them afterwards. It has the responsibility to deliver and verify the solution. When trying to understand company and development complexity, it's necessary to understand its organizational structure. Hierarchy often regulates amount of power and responsibility within development projects. Right balance of power is important for the success of collaborative development. Matrix-organization (Figure 4) is a common way of arranging departments and roles inside companies where two types of lines are facing each other. Figure 4 suggests vertical product oriented and system-oriented as horizontal lines. Vertical product oriented program teams have the responsibility to integrate different systems into specific products whereas the system oriented departments focus on the integration of specific systems into different products [Königs et al. 2012]. Projects span through multiple lines intersecting with one or few product- and system-oriented lines.

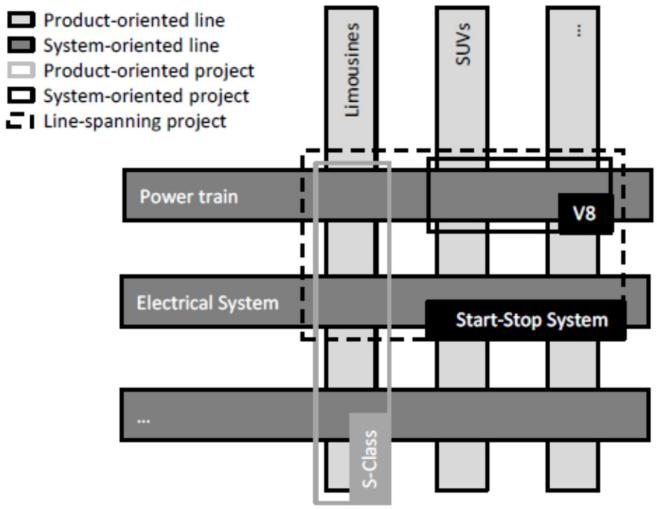

Figure 4. Projects within matrix-organization [Königs et al. 2012]

Process and development control are one other aspect of organization. Processes define the sequence, timeline, expected results and responsibilities of teams. The gateway processes [Königs et al. 2012] in automotive industry can be seen as a validation door for merged sub-processes. Those gateways are used to validate the synchronized results from different teams, to track the status of the project and to define corrective actions. Today, processes in the automotive industry are rather product-, not system-oriented. The development of a new system is initiated either top-down or bottom-up within the organizational structure. Top-down initialization comes after certain management attention and sets a mandate for development. It is often a quick and urgent response to new customer demands. Special project teams are under time pressure and they are working in close cooperation with production development to ensure fast integration [Königs et al. 2012]. Standard processes for this kind of cross department project usually don't exist and have to be defined on the fly. It is obvious that this kind of development practice causes deficiencies in development robustness. Bottom-up approach, on the other hand, is initiated by ideas within the departments. New specification comes as a result of customer feedback, supplier experience, repair and servicing shops and possibly from other sources that base new ideas on their own experience. There is usually less time pressure but projects are not mandatory to be integrated into existing systems. Projects gain attention only by achieving good results and then they might be converted into top-down system development project. Top-down and bottom-up system approaches are both dealing with same problems regarding collaborative work and cross-linked information within those processes. Problems lay in inconsistent, hard to retrieve or outdated information across departments, low transparency about changes and low transparency about impact of changes [Königs et al. 2012].

People tend to search for new tools and methods for accomplishing the task they have in front of them. This happens separately and simultaneously in different areas in the company, leading to inhomogeneous set of applied tools and methods. The current gateway processes mentioned in previous paragraph define goals, timelines and responsibilities across different teams but do not assist the coordination of information in between the gateway points.

Current practice from a tool perspective in German automotive industry suggests that Microsoft Office suite, MATLAB- and Simulink-product family tools have become de facto the standard [Königs et al. 2012]. Furthermore, there is a wide diversity of tools used by specialist departments which help them to achieve better results but stresses an ease of integration and systematic coordination of engineering data across the company. Non-coordinated data contributes to unnecessary iterations which manifest as rising product cost and possibly can have an impact on product quality. Inevitably, the number of tools and amount of generated data will persist to increase (Table 1). Companies will struggle more and more with more complex data and organization management; therefore there is a strong need to support the aspects of interaction and communication within the development process and across organization structures.

Table 1. Relevant functionality in engineering tools to enable traceability [Storga et al. 2011]

PDM	CAD	Office tools
Project management Document versioning Workflow mechanism Engineering change management Search/Query engine Report generator	Feature tree (structure of the CAD model) Associatively links between assemblies and parts File versioning 3D model characteristic management	Changes tracking mechanism Document properties management

Analyses of the automotive industry of the 1980s already showed a high complexity concerning processes and products, and numerous strategies for its systematic management have since been designed [Clark and Fujimoto 1991]. Bullinger mentioned that the trend towards increasing variant numbers and product complexity will continue in the automotive industry [Bullinger et al. 2003]. An approach which potentially addresses the mentioned problems was introduced in the late 1970s. It focuses on the management of interdependencies between software requirements and other artefacts in software engineering and is called traceability [Königs et al. 2012].

3. Conceptual development of methods for extracting product assembly relations

Methods for extracting valid relations between components from finished product assembly are developed to support project traceability and to address the product complexity [Bhaskara 2011]. Finished product assembly is virtual representation of a real product therefore it is created in a way that represents real system structure. System structure consists of multiple components that are related to each other. There are multiple types of relations that exist but the point is to extract all of them automatically. For this to be able to develop, existing design methodology and features for creating relations need to be explored. In this case, features used for creating an assembly in Siemens NX will be used to find important component relations. Even though the concept of extracting relevant relations is based on features and possibilities of Siemens NX, it can easily be generalized and applied in different contexts where components are in some way geometrically related to each other and when there is a virtual representation of them available in form of virtual system structure. Virtual system structure in this case is represented with CAD assembly model.

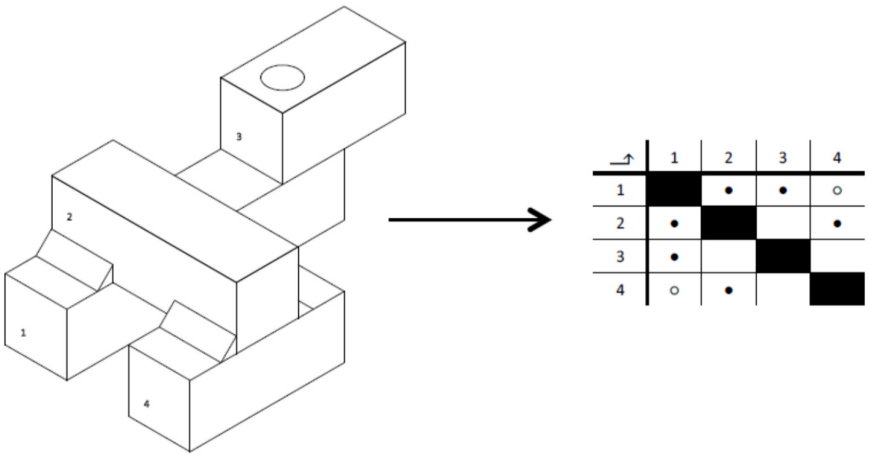

Figure 5. Assembly-to-DSM

Why to develop methods for extracting relations between system components if those relations are already defined in drawings and somewhere in project documentation? The key word is 'somewhere'

and this is the word on which the answer is based on. When there is a virtual representation of system structure available, all the relations are centralized. If the DSM matrix based on this system structure has to be created, the easiest and most logical way to extract the relations is from the source that contains all of them in one place. This is where the power of well-developed methods for extracting important relations from product model comes into place. Assembly-to-DSM (Figure 5) is the common name for all the methods described in this thesis therefore it represents the process of getting DSM matrix from product assembly.

Every CAD application has ability to manage relations between assembly components. But those relations does not say much about the nature of component's relations. Real product's components are connected by permanent (welds) or non-permanent joints (represents screws and remote forces that cause interaction between two or more components). There are eleven types of constraints in Siemens NX. Very often they are combined to achieve expected component behaviour inside the assembly. The method was developed to recognize certain relations between components and consider components in question to be permanently or non-permanently joined. To be able to analyse relations between components one very important Siemens NX functionality was explored and used. This functionality is Proximity function. Proximity function extracts component's relations and takes in the account distance between the components. No relation between components has to be defined prior to applying these function. In order to distinguish if relation should be considered permanent or non-permanent a threshold have to be defined. User chooses referenced value (R) which is then put against compared value (C). Result is TRUE or FALSE depending on the comparator method which is 'larger than' (>) in this case. TRUE comes as a result if referenced value is larger than compared value and it means that the relation for chosen distance is valid. FALSE comes as a result if compared distance is larger than referenced one and it means that the relation shouldn't be taken into account because the components are too far apart. Based on the proximity function an application was developed how to utilise this function to extract needed information about relations between assembly components. Core application was developed using Siemens NX Open API for java language. Unfortunately available APIs does not include methods for data visualization outside the Siemens NX application. That is way output of the application is written as JSON objects. Created .json files are visualized using JavaScript d3.js library. The main application workhorse is appProximity function (excerpt of the function code is shown in Table 2.)

Table 2. appProximity function code excerpt

```
for i = 1 → total # of components
·       // box
·       for j = 1 → total # of components
·       ·       create boxSize
·       ·       create inBox
·       ·       if inBox && component[i] != component[j] && !assembly
·       ·       ·       create boxVolume
·       ·       ·       create componentVolume
·       ·       ·       create volumeRatio =  componentVolume / boxVolume
·       ·       ·       currentBoxRelation
·       ·       ·       ·       += component file path
·       ·       ·       ·       += component tag
·       ·       ·       ·       += relation summary
·       ·       ·       ·       += relation type
·       ·       ·       ·       += # of related components
·       ·       ·       ·       += constraint reference type
·       ·       ·       ·       += constraint related components
·       ·       ·       ·       += constraint tag
·       ·       ·       ·       += volumeRatio
·       ·       ·       add currentBoxRelation to proximityRelations
·       ·       end if
·       next j
·       // distance
·       for k = i → total # of components
·       ·       create minDistThreshold
·       ·       create minDistMeasured
```

```
        // threshold distance is smaller than minimum measured between i and k component
        if minDistThreshold <= minDistMeasured && minDistanceThreshold != 0
        && component[i] != component[k] && !assembly
                currentDistanceRelation
                        += component file path
                        += component tag
                        += relation summary
                        += relation type
                        += # of related components
                        += proximity reference type
                        += proximity related components
                        += proximity tag
                add currentDistanceRelation to proximityRelations
        end if
        // components touch or intrude one another
        if minDistMeasured = 0 && component[i] != component[k] && !assembly
                currentDistanceRelation
                        += component file path
                        += component tag
                        += relation summary
                        += relation type
                        += # of related components
                        += proximity reference type
                        += proximity related components
                        += proximity tag
                add currentDistanceRelation to proximityRelations
        end if
    next k
next i
```

The function search for all relations in the assembly and compares the relation values to the threshold that is set by the user. Threshold filters important relations from those which are not. The number of extracted relations depends on how strict are the user parameters. The boxVolume and minDistanceTreshold are two variables that are set by the user. The boxVolume defines distance between corner box coordinate and centre point coordinate for the virtual box definition. Default component virtual box will grow. The minDistanceThreshold is compared with minimum distance between two components. If the minDistanceThreshold is larger than calculated minimum distance between components, relation is considered important. The volumeRatio is a parameter calculated as ratio between the volume of the selected component and the volume of the virtual box for this component (Figure 6). For the testing purposes threshold for this ratio is set to 0.5 (any relation calculated by the function that has ratio below this value will be discard).

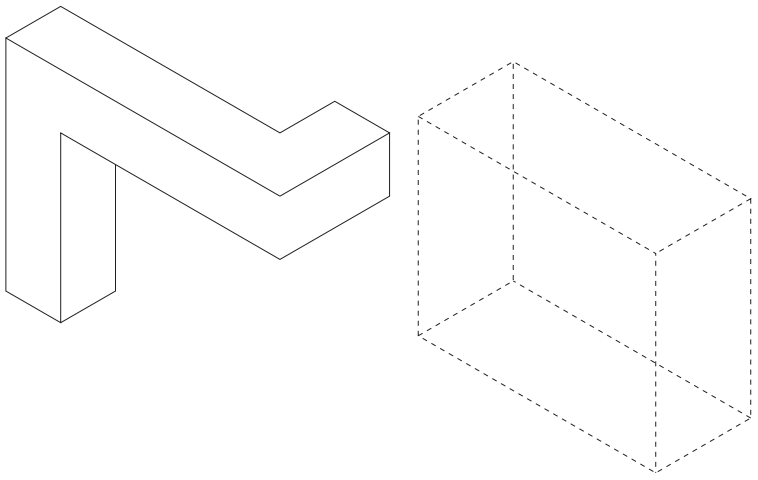

Figure 6. Low body-box ratio – all relations to selected component from other components within the box should be excluded

This is important because if we take in to consideration only box method, both components that fall into the virtual box would be excluded from extracting the relation because of the small volume ratio. Minimum distance method will still include the relation of one small component (Figure 7, black) even though it is discarded with box method. If we assume that the minimum distance value to reach the other small component (Figure 7, grey) is larger than the referenced value, then the validation of the relations from sample model in is acceptable.

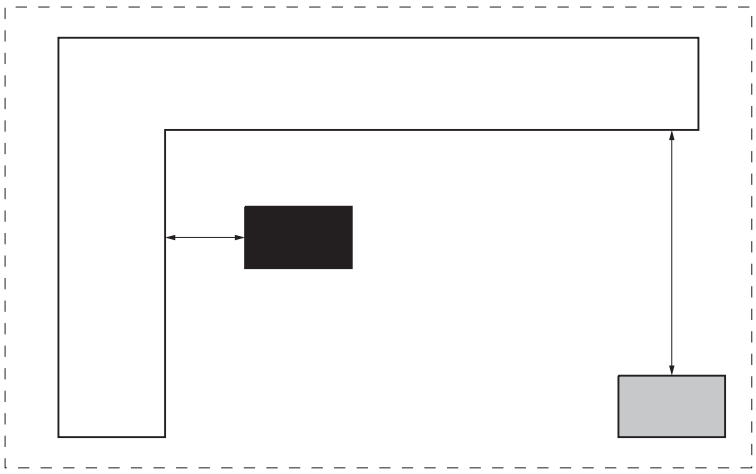

Figure 7. Box method together with distance method

Based on the user defined threshold value only relations that are of interest and that fails into category permanent or non-permanent joint is extracted and processed. At the end of application run .json file is created containing necessary data for assembly relation visualization as DSM matrix. Part of the JSON file is shown in Figure 8.


```
…
{
        "summary": "block-8 catches box of block-2-7",
        "type": "INSIDE_BOX",
        "path": "C:\\Users\\alzubic\\Desktop\\tmp NX files\\validation\\block-8.prt",
        "componentTag": [39690],
        "referenceType": ["15.0mm BOX"],
        "realName": ["8","3"],
        "referenceName": ["block-8","block-2-7"],
        "referenceTag": [39690,39660],
        "numberOfRelatedComponents": 2,
        "volumeRatio": 0.2912846327236437
}
…
```


Figure 8. Slice of result data - JSON format

4. Application validation

For application validation a test assembly was created. Validation model (Figure 9) is simplified real life representation of an assembly. Assembly components are arranged in a way that is challenging for prototype relation extraction algorithm to recognize all the important relations. For this assembly the DSM matrix is created manually so the result from the application run can be validated.

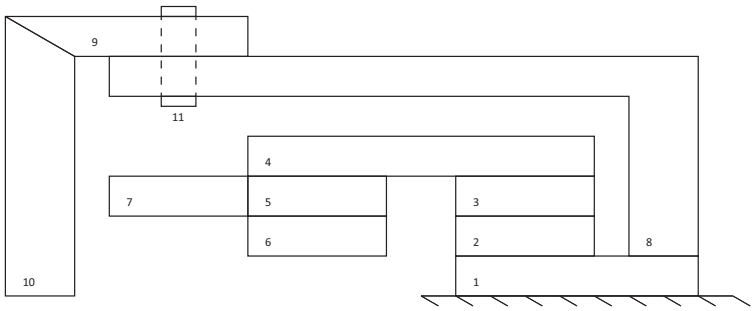

Figure 9. Validation model

The created DSM is shown in Figure 10. It is important to note that the amount of time is significant for describing even the simple model like this. If complex technical systems are taken into account, it becomes almost impossible and completely inefficient to create DSMs manually. Again, this confirms the need for automatic relation extraction approach. Such approach is beneficial in saving man power, working hours and in many other activities related to the process of manually creating DSMs.
Final results are visualized using d3.js graph library with some help from JavaScript, JQuery, HTML and CSS. As can be seen in Figure 11, results are put in DSM style table where darker grey colour represents existing relation between the components. Necessarily, there is not only one type of relation between two components, but for the validation purposes is determined that if there is at least one relation detected, the field is darker grey coloured. All the other detected relations between the same two components are still in JSON files even though the number of them is not shown in the visual output.

↓→	1	2	3	4	5	6	7	8	9	10	11
1		|●□						|●□			
2	|●□		|●□					|□			
3		|●□		|●□				|□			
4			|●□		|●□		|□	|□			
5				|●□		|●□	|●□				
6					|●□						
7				|□	|●□					|□	
8	|●□	|□	|□	|□					|●	|□	|●□
9								|●		|●◊□	|●□
10							|□	|□	|●◊□		
11								|●□	|●□		

•	◊	|	□	_
TOUCH/INTERSECT	WELD	MIN. DISTANCE	BOX	CONSTAINT

Figure 10. Manually created aggregated DSM based on model from Figure 11

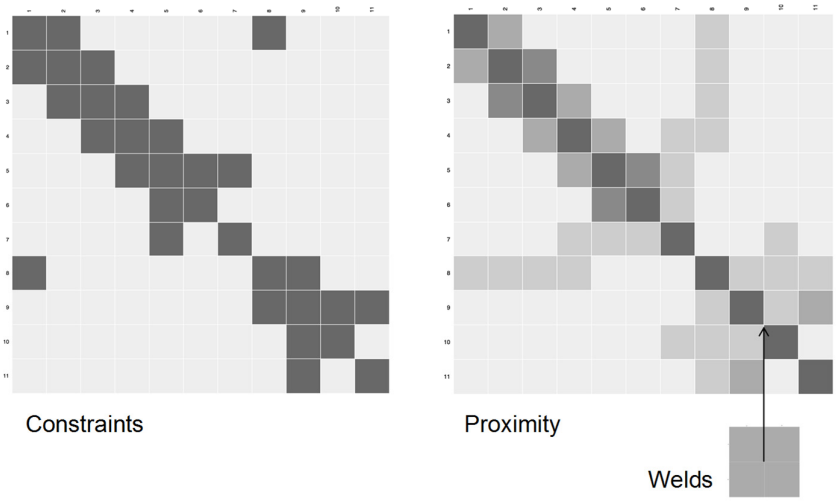

Figure 11. Visualized results - separated based on the extraction method

The application was applied on real model and the results gained are supporting the validity of the proposed model. Tested model was download from the internet site GrabCAD (https://grabcad.com/) and it is shown on the Figure 12. The resulting DSM table is shown in the Figure 13.

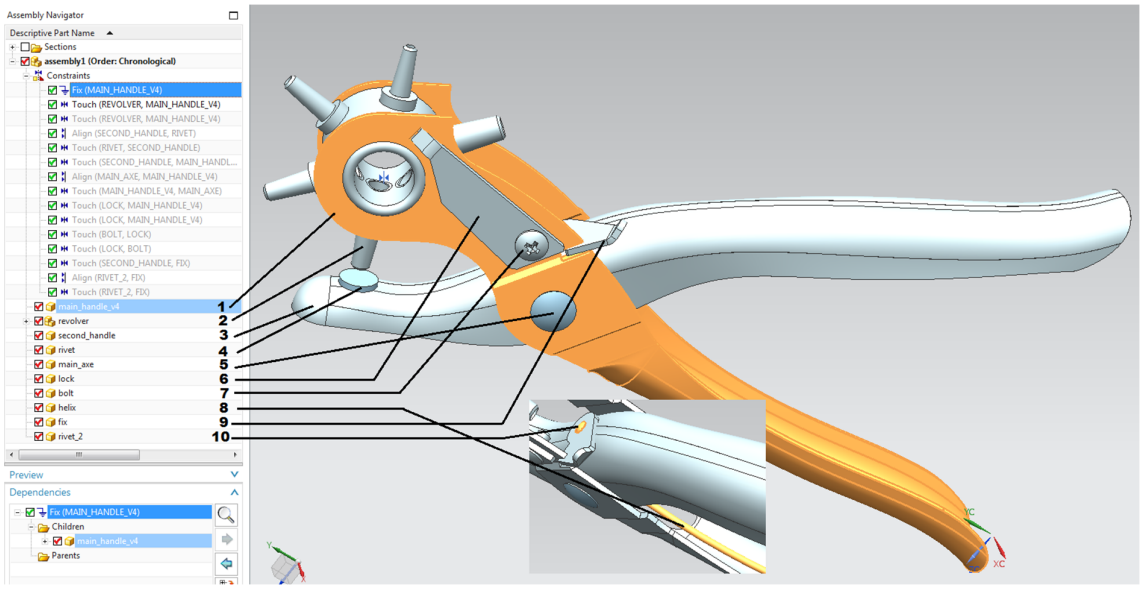

Figure 12. Hole puncher (model created by Dmitry Natkha, taken from GrabCAD site)

	1	2	3	4	5	6	7	8	9	10
1	■				▒					
2	▒	■								
3	▒		■	▒					▒	
4			▒	■						
5	▒				■					
6	▒					■	▒			
7						▒	■			
8								■		
9									■	▒
10									▒	■

Figure 13. Application output

5. Conclusion

Figure 14 presents the aggregated view of the separate results shown in Figure 11 on the right together with manually created DSM on the left. Those are two DSMs which need to be compared to validate the results produced by the algorithm. Rows and columns contain numbers from 1 to 11 that represents the number of the components labelled in Figure 9. When Manual and Assembly-to-DSM matrices are overlapped, it can be stated that developed algorithms are well designed. Algorithms extract proper relations from given CAD assembly model and results are successfully validated in comparison with manually created DSM. There is one important lesson to be learned here and th4s is that computer never miss their goal if well-tuned. Proof to support this claim comes from Figure 12.

It should be noted that developed methods are applicable on an existing assembly while creating the product or after the product is finished. It means that it compliments traceability with new data after the product is completed or during iterative design process. Methods are especially helpful in case of creating new version of existing product because of an existing overview of system architecture. Presented work is done in cooperation with Daimler company and is meant to be implemented and used in their design processes. Daimler has certain design rules for creating CAD assemblies. With those rules in place, standardization of process is achieved. While writing this article, it is noticed that there is some room for improvement. If Daimler adopted following design rule for creating assemblies, other methods on DSM matrix (not just clustering) would be applicable: while creating constraints between components, first select the one on which the next component depends on. Also, start from the root component - component from which the real-life assembly starts. It primarily means that sequencing methods could be helpful in creating step-by-step component mounting diagrams because the order of assembling could be extracted.

Currently, component assembly sequence is identified by the user. Decomposing that kind of an assembly with the suggested rule applied in the process, it would give non-symmetric DSM matrix which could be used to remove 'PowerPoint Engineering'

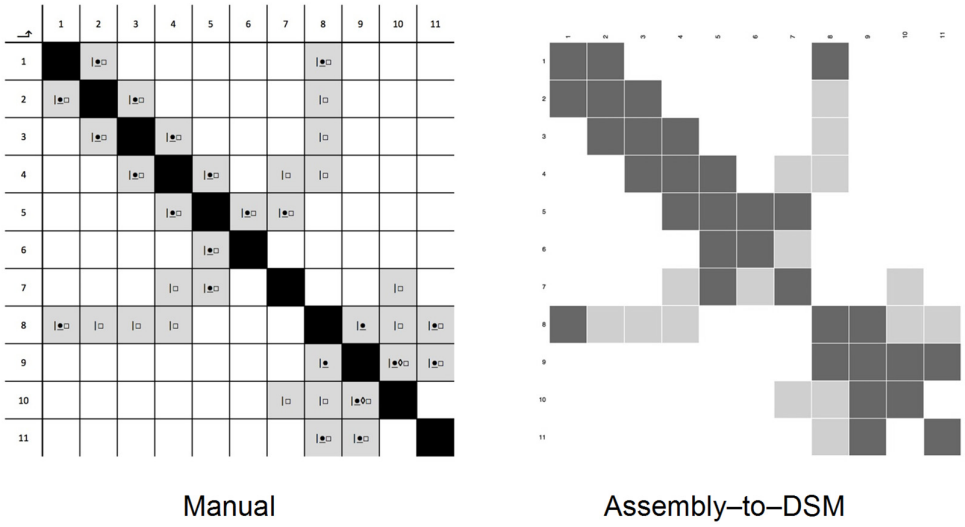

Figure 14. Visualized results – compared manually and programmatically created DSM matrix

Closer look at coordinates 6-7 and 7-6 This is symmetric DSM and therefore both coordinates indicate the same relation between components 6 and 7 (proximity), but you can notice that this relation was not recognized in manually created DSM! This is not done on purpose and since result is valid, credit goes to developed algorithms. Statistical probability of human error in recognizing relevant component relations rises with the product complexity whereas machine needs less time and is much more precise. Methods for extracting relations that exist between components in the Siemens NX product assembly model were successfully identified. Extracted relations are then stored in JSON format which is

convenient for usage in any application that aims to improve product traceability. Methods are derived from four main relation types – constraints between components, component proximity, permanent and non-permanent joints. DSM matrices are chosen as a tool to visualize relations because of their ability to simply visualize a complex system architecture. Additionally, post-processing methods to manipulate DSM data are described for further system analysis. PoC demonstrated the way it is possible to automate the extraction of relations from the CAD model. Results are successfully verified by comparing the DSM matrix produced by the PoC algorithm with a manually created matrix based on a validation CAD model created for the purposes of this thesis.

References

Bhaskara, S., "Analysis and Visualization of Complex Computer Aided Design Models as a Design Structure Matrix", 13th international dependency and structure modelling conference, DSM'11, September 14–15, Cambridge, Massachusetts, USA, 2011.

Bracewell, R., Marina, G., Moss, M., Knott, D., Wallace, K., Clarkson, J., "DRED 2.0: A method and tool for capture and communication of design knowladge deliberated in the creation of tehnical products", Proceedings of ICED 09, the 17th International Conference on Engineering Design, 24.-27.08., Palo Alto, CA, USA, 2009.

Bullinger, H., Kiss-Preussinger, E., Spath, D., "Automobilentwicklung in Deutschland – wie sicher in die Zukunft? Chancen, Potenziale und Handlungsempfehlungen für 30 Prozent mehr Effizienz", Fraunhofer-IRB, Stuttgart, 2003.

Chaovalitwongse, W., Pham, H., Hwang, S., Liang, Z., Pham, C., "Recent Advances in Data Mining for Categorizing Text Records", Recent Advances in Reliability and Quality in Design, Springer, 2008, pp. 423-440.

Clark, K., Fujimoto, T., "Product Development Performance: Strategy, Organization and Management in the World Auto Industry", Harvard Business School Press, Boston, 1991.

Eppinger, S. D., Browning, T. R., "Design Structure Matrix Methods and Applications", MIT Press, 2012.

Königs, S. F., Beier, G., Figge, A., Stark, R., "Traceability in Systems Engineering - Review of industrial practices, state-of-the-art technologies and new research solutions", Advanced Engineering Informatics, Vol.24, No.4, 2012, pp. 924-940.

Lindemann, U., Maurer, M., Braun, T., "Structural Complexity Management: An Approach for the Field of Product Design", Springer, 2009.

Naumann, T., Königs, S., Kallenborn, O., Tuttass, I., "Social Systems Engineering - An Approach for Efficient Systems Development", Proceedings of ICED 11, the 18th International Conference on Engineering Design, 15.-19.08., Lyngby/Copenhagen, Denmark, 2011.

Sanchez, R., Mahoney, J. T., "Modularity, flexibility, and knowledge management in product and organization design", Strategic Management Journal, Vol.17, 1996.

Storga, M., Bojcetic, N., Pavkovic, N., Stankovic, T., "Traceability of Engineering information Development in PLM Framework", Proceedings of the 8th International Conference on PLM, Eindhoven, 2011.

Tarnovski, B., "DSM razvojnog projekta električnog sportskog automobila", Master-Thesis, FSB, Zagreb, 2011.

Nenad Bojčetić, Associate Professor
University of Zagreb FSB, Chair for design and product development
Ivana Lucica 5, 10000 Zagreb, Croatia
Email: nenad.bojcetic@fsb.hr