1 INTRODUCTION

Ever increasing product complexity continues to provide challenges in the handling and management of interactions and dependencies within products, where conflicts and constraints are needed to be resolved on an almost daily basis. For example, issues in the length of cabling required in the A380 led to a $6.1 billion delay for the project whilst Toyota has had to recall 625,000 vehicles due to faulty hybrid software (Calleam, 2011 and Bruce, 2015). Both examples represent the increasing challenge in being able to fully account for these interactions and dependencies, with failure to do so leading to costly overruns, product recalls and/or lengthy re-designs.

To support engineers and engineering projects, Design Structure Matrices (DSMs) have been widely applied as a means to identify, visualise and monitor product & organisational architectures (Sosa et al., 2003). Developed in the 1980s by Steward (1981) as a branch of graph theory, DSM seeks to understand the connected nature of engineering systems through an $N \times N$ matrix of interactions between system elements (Eppinger, 1997). These systems can represent individual components, assemblies of components, systems of components, engineers, teams of engineers, processes, and/or organisational structures to name a few, with the level of interactivity between elements being manually scored using either a range or binary set of values (Sosa et al., 2003 and Gorbea et al., 2008). From this, partitioning of the elements and matrix visualisation techniques are applied to enable insights to be drawn on the product/organisational architecture. An example is given in Figure 1, which reveals the key structures within a commercial jet engine and was used to support the design and development processes within the company. Although insightful, challenges still exist in being able to maintain an up-to-date DSM due to the labour-intensive nature of the manual capture methods.

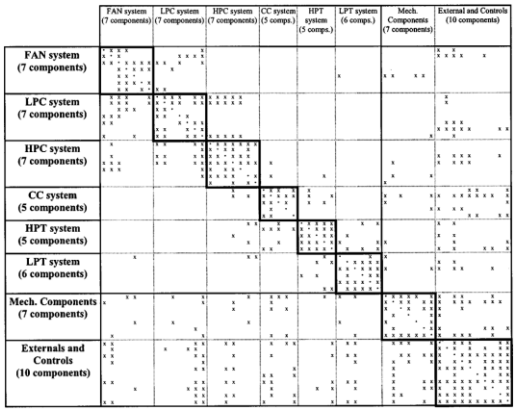


Figure 1: A DSM of a commercial jet engine (From: Sosa et al., 2003)

To overcome this, recent work in the field of DSMs has investigated the potential of using the co-occurrence of product model edits (i.e. where a product model is edited within a pre-defined time-period of another product model) to produce an automated, more objective and real-time method of generating and monitoring the evolution of DSMs (Gopsill at al., 2016 and Senescu et al., 2012). This has been possible through Product Lifecycle Management (PLM) systems that generate a digital footprint of project activity that consists of thousands of product models that have been created and edited by distributed teams. For example, the development of the Boeing 787 Dreamliner involved the production of over 300,000 Computer Aided Design (CAD) models, which were accessed between 75,000 to 100,000 times a week (Briggs, 2012). In addition, Jones et al., (2016) revealed that searches for documents across Airbus totalled 1.1 million over a six-month period. As automatically generated DSMs are more objective due to their systematic process in the modelling of the dependencies and interactions of product models, opportunities exist in performing cross-project comparisons of the evolution of product architectures. This could provide insights and learnings that could be used to support the development of best practice and project benchmarking and be of particular interest to engineering companies that operate programmes of projects to develop families of products.

It follows that the contribution of this paper lies in the exploration of two DSMs that have been automatically generated from the product model edits of two Formula Student teams. Where the focus has been on the learnings and insights for project management and performance of the design. The paper continues by discussing the product model datasets that have been captured from two Formula Student teams and that provide the basis for the generation of the DSMs. This is followed by a discussion of the method used to automatically generate the DSMs from the co-occurrence of edits to product models. The results and associated discussion are then presented where the study compares the:

- end-of-project DSMs;
- change propagation characteristics of the DSMs, and;
- evolution of the DSMs.

The discussion focuses on the learnings and insights that can be inferred through inspection of the DSMs and comparison of the two projects. This then leads into the conclusions and future work section of the paper.

2 FORMULA STUDENT AND THE PRODUCT MODEL DATASETS

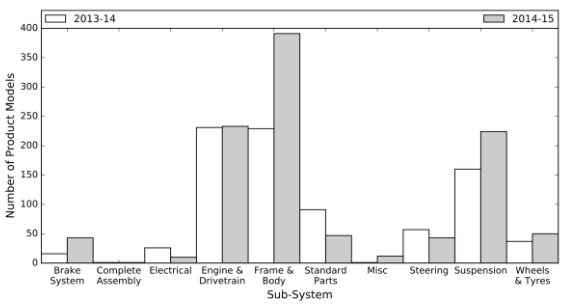
To investigate the potential of comparing automatically generated DSMs from engineering projects, this study has captured the product model editing history of two Formula Student projects. Formula Student (also known as Formula SAE) is a motor-sport educational programme whereby teams of students from competing universities create a single-seat race car that competes in various challenges set-out by the competition organisers. The competitions are held worldwide and include events in the United Kingdom, United States of America, Australia and Europe. The teams in this study consisted of approximately 30 engineering students who were in their final year of study and have undertaken a range of engineering courses including automotive, aerospace, electrical, manufacturing and mechanical.

The construction and manipulation of the product models is performed on a shared network drive hosted at the University. To manage their product models, the teams utilise a custom-built lightweight CAD management tool that manages the naming conventions, relationships and organisation of the product models on the shared network drive. The management tool also provides a hierarchy and manual classification of the product models with respect to the various sub-systems of the car as defined by the team. It is important to note that all work on the product models is performed on the shared network drive. Hence, by monitoring accesses, creation and modification of these product models, there is potential to capture and reveal the inter-dependencies across the design.

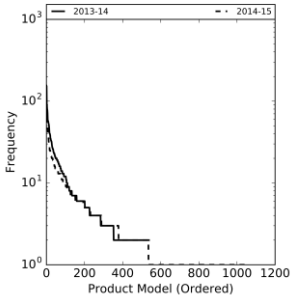
The monitoring was enabled by a Raspberry Pi that was connected to the network, which recorded the status of the shared network drive at 20-minute intervals. More specifically, the folder structure alongside the meta-data attributes of all the product models was captured. This included model size, date accessed and date modified and is akin to the metadata stored within PLM systems. Table 1 presents the details of the number of models produced and the number of edits recorded across both datasets. The results show that Team B produced almost twice the number of models compared to the Team A, however, the total number of edits recorded remained the same. To further investigate this, Figure 2 shows the distribution of product models by sub-system (2a) and frequency of edits made to the product models for both teams (2b). It is clear to see that the additional product models generated by Team B relate to the Frame & Body sub-system. In addition, the frequency plot reveals that these additional models extend the tail of the frequency plot and this suggests that these models are of standard/bought-in components that are not edited further by the team.

Table 1: FS product model statistics

Statistic / Team	A	B
Product Models	539	1053
Number of Edits Recorded	4609	4246



(a) Distribution of models by sub-system



(b) Frequency of edits by model

Figure 2: FS product model characteristics

To complement the product model datasets, both teams' performance in the competition has been recorded and will be used as the indicator for project success (Table 2). Team B outperformed Team A in the overall competition with a 4th place finish as well as being the top UK entry. The main

Table 2: FS competition results

Statistic	A	B
Overall Competition	7th	4th
UK Team	2nd	1st
Endurance Event	4th	1st
Sprint Event	13th	12th
Skid Pan	9th	13th
Acceleration Event	8th	18th

contributing factor comes from the endurance event score where Team B won overall. The endurance is typically weighted more in determining the overall results. The hypothesis is that the structure of the product architecture may be an indicator of project/product performance and that automatically generated DSMs has the potential to reveal this.

3 AUTOMATICALLY GENERATING DESIGN STRUCTURE MATRICES

The DSMs generated in this paper follow the process as defined by Gopsill et al., (2016) where the co-occurrence of product model edits have been used to form the $N \times N$ matrix of interactions. An example of a co-occurrence is where product model B is edited within a pre-defined time-period of product model A. The process comprises seven stages and covers:

1. the Initial Model Selection;
2. Generating the Co-Occurrence Matrix;
3. Evaluating the 'directedness' of the Matrix;
4. Weighting the Matrix;
5. Pruning the Matrix;
6. Partitioning the Matrix; and,
7. Optimal Time-Period and Pruning Level Selection.

As this paper is applying the same analysis to a similar dataset as analysed by Gopsill et al., (2016), the optimal time-period and pruning level has been pre-defined as 3 hours and 0.25 respectively. Thus, the paper continues by describing stages 1-6 and their application in the context of analysing the two Formula Student datasets. Further details of the overall method and how the optimal time-period and pruning levels are determined is given in Gopsill et al., (2016).

This paper also uses a number of matrix analysis definitions throughout the method and results & discussion sections. Their definition with respect to this context are as follows:

- Partitions - A cluster of highly-interdependent product models, which represents a sub-system within the product architecture. These also have dependencies two other partitions.
- Components - A cluster of highly-interdependent product models that are not dependent on other partitions within the DSM and hence represent distinct sub-systems within the product architecture.
- Modularity - A metric that provides an insight into the level of structure within a matrix and thus, provides an indication to the level of structure within the product architecture.

3.1 Initial Model Selection

Since these DSMs are generated from the co-occurrence of edits, the analysis can be applied to product models from a variety of sources including models from Computer Aided Design, Computational Fluid Dynamics and Finite Element Analysis, as well as reports and communications. In the case of this study, the focus has been on the CAD files as the teams applied a custom-built lightweight CAD management tool that manages the naming conventions, relationships and organisation of these product models whilst other product models used by the teams were stored in a more ad-hoc manner and across many storage devices.

The models are then filtered based on the number of edits made. For this study, this has been set to four as this includes the creation of the product model followed by three further updates. This removes models that show little to no activity and may represent constraints in the team's design space that they are unable to alter (for example, a bought in part such as the engine block and/or gearbox) as well as standard parts/fixings (such as, nuts & bolts). Following this filtering, the product models of interest totalled 348 and 282 for Teams A and B respectively.

3.2 Generating the Co-occurrence Matrix

To generate the initial DSM matrix, it is necessary to identify when a co-occurrence of product model activity has occurred. To achieve this, the process iterates through each 'date modified' date for a product model and identifies which product models have also changed within a specified time-period. The matrix is then updated to reflect the co-occurrence of edits. In the case of this analysis, the time-period has been selected as 3 hours following the settings derived by Gopsill et al., 2016.

3.3 Evaluating the 'directedness' of the Matrix

As the process of generating the co-occurrence matrix identifies models that have changed within a specific time-period following a change to another product model, the resulting matrix is inherently 'directed'. However, to apply the most suitable partitioning algorithm for the DSM a check for the level of 'directedness' must be made. If the matrices were found to be undirected (i.e. symmetrical where the number of times product model A is edited following a product model B edit is the same as the number of times product model B is edited following product model A) then it is preferable to treat the matrix as undirected as the algorithms for the partitioning of these matrices are currently more mature than that of directed matrices. This was found to be the case for the two teams and hence, undirected matrices were generated for the two teams.

3.4 Weighting the Matrix

With the matrix elements featuring the number of co-occurrences of product model edits between product models, there is an inherent potential for the strength of the dependency to be influenced by the number of edits that have been made to each product model. There is also the chance of a co-occurrence representing concurrent working practices within the project. Therefore, a matrix weighting scheme is applied to reduce the effect of these factors on the strength of dependencies that have been identified between product models.

For undirected matrices, each cell is divided by the sum of the total number of changes made to both models that the cell represents, i.e. the number of co-occurrences between model A and B divided by the sum of the total number of changes made to models A and B. This normalises the dependencies with higher values representing a greater likelihood of the existence of a dependency between the two models. This normalisation also enables comparison across the range of model activities observed.

3.5 Matrix Pruning

To further remove the potential effect of false positive dependency relations influencing the partitioning and results of the DSM, pruning of the matrix elements is performed. This pruning removes low weighted dependencies within the matrix. The pruning level for the two team's datasets applies the same setting used by Gopsill et al., 2016 of 0.25. The results of the pruning are shown in Figure 3

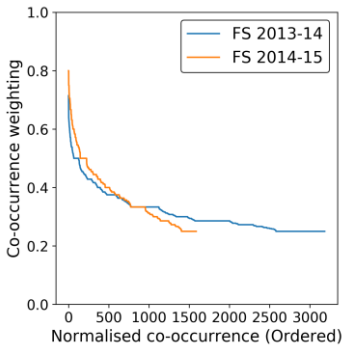


Figure 3: Pruned co-occurrence weights

3.6 Matrix Partitioning

With the matrices formed, it is possible to group highly-dependent product models through the application of matrix partitioning. To achieve this, the Louvain community partitioning is applied as it has been shown to suit continuous measurements of dependency. The Louvain community algorithms' objective is to generate a set of partitions for the matrix that returns the highest modularity value. Modularity (Q) is an assessment of the quality of the matrix partition and is defined as (Newman, 2004):

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

Where $m = \frac{1}{2}\sum_{ij} A_{ij}$ and is the number of co-occurrences within the matrix. δ is the Kronecker delta function and is 1 if a co-occurrence exists between two models and 0 otherwise. $\frac{k_i k_i}{2m}$ is the probability that a co-occurrence may exist between two models, where k_i is the number of models that have co-occurrences with model i, and k_j is the number of models that have co-occurrences with model j. And, A_{ij} is the weighted co-occurrence between two models in the matrix.

A modularity of greater than 0.3 is considered to show that a 'good' level of partitioning has been achieved and that there is underlying structure and relationships between the matrix elements that is beyond pure chance (Newman, 2006). The implementation of this algorithm has been through the NetworkX community partition python package (Hagberg at al., 2008). In the case of the two DSMs generated in this paper, the modularity scores were 0.55 and 0.64, which demonstrates that an underlying structure is present in the co-occurrence of product model edits.

4 DISCUSSION & RESULTS

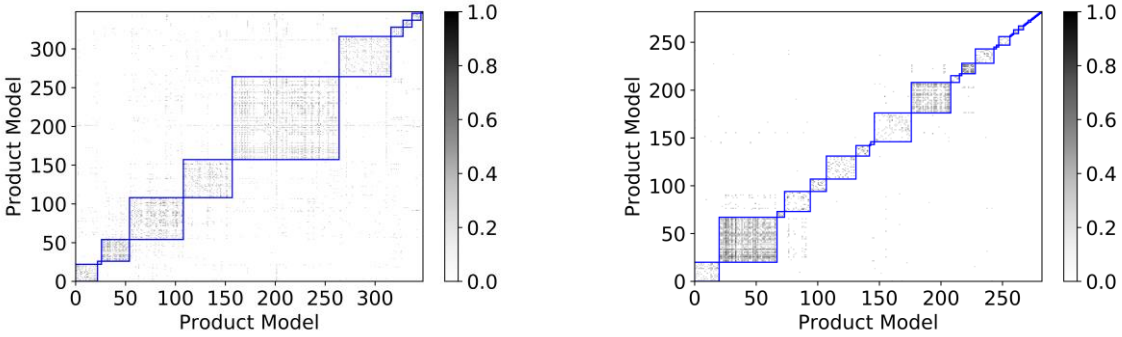
To compare the two projects using automatically generated DSMs, a comparison of the end-of-project, change propagation characteristics and evolution of the DSMs has been undertaken. The end-of-project comparison examines the partitioning results of each projects design. Whilst the change propagation characteristics reveal how changes to product models will impact the product architecture. Examination of the evolution of DSMs takes advantage of being able to automatically generate DSMs in real-time and examines the working practices between the two teams. Throughout the discussion, attention will be made to the key insights that have been made in the form (Insight X).

4.1 End of Project DSM

Figure 4 and Table 2 present the results from the analysis of the end-of-project DSM. It can be seen that Team B's design appears to be more modular with 37 partitions as opposed to 10. It is interesting to note that 29 of these partitions were components of the matrix and thus, have no additional dependencies with other partitions in the matrix. This may highlight that Team B may had a better understanding of the dependencies between components and were therefore able to generate a more modular design. Thus, the level of modularity of the design may be indicative of a potentially more successful product as Team B outperformed the Team A at the FS competition (Insight 1).

Table 2: End-of-Project DSM Statistics

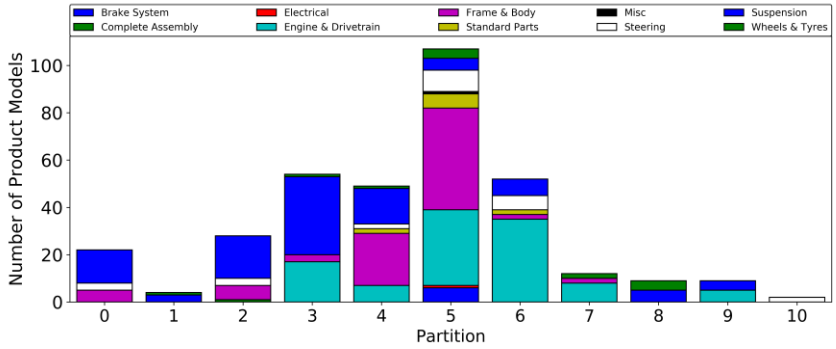
Statistic	A	B
Number of Product Models	348	282
Number of Dependencies	3185	1588
Number of partitions	10	37
Number of components	1	29
Modularity	0.55	0.64



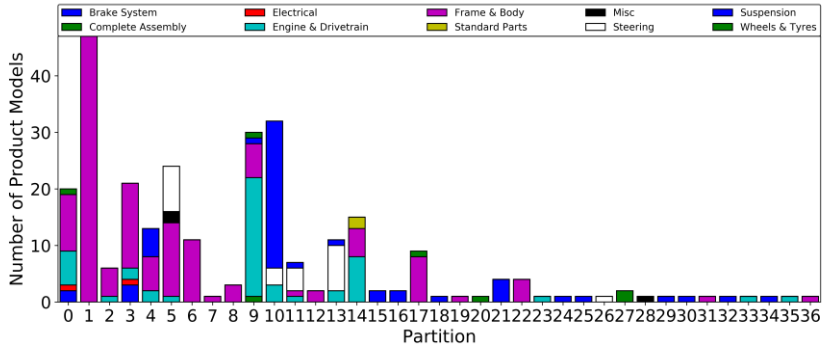
(a) Team A *(b) Team B*

Figure 4: Partitioned DSMs for the two teams

Figure 5 delves further into the composition of the partitions that have been generated from the analysis and shows the composition of the clusters in relation to the pre-defined product model classification made by the teams. Team B's more modular design can be clearly seen through the greater number of smaller partitions that are focused on single pre-defined sub-systems. In addition, the larger partitions for Team B are noticeably more in line with the initial sub-system classification outlined by the team, which potentially further highlights that the team were able to better manage the inter-dependencies between sub-systems as well as potentially better at identifying the sub-system boundaries during their initial manual classification (Insight 2).



(a) Team A



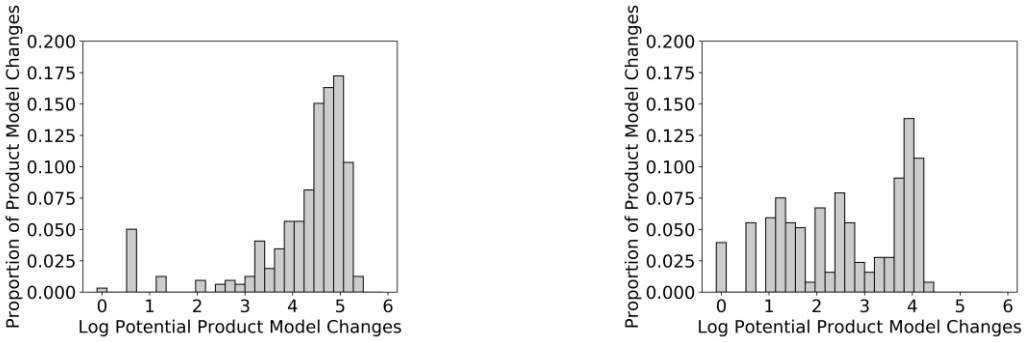
(b) Team B

Figure 5: Distribution of sub-system components across the partitions

4.2 Change Propagation

A typical application of DSMs is to use them as a predictor/indicator for the capability of a system to accommodate changes and the resulting propagation of the change through adjacent sub-systems and components (Clarkson et al., 2004 and Pasqual & de Weck 2012). Figure 6 shows a normalised histogram of the logarithm of the number of changes that are likely to occur for each product models' three-branch propagation tree. A threshold of 0.3 for the dependency weighting was set to indicate that a change will occur to a related product model. The logarithm has been taken as the change propagation trees are a continuous positive value and thus, the distribution exhibited is most likely to be of a log normal form.

The histograms presented in Figure 6 clearly demonstrate that most of the product models within the DSM of Team A have more extensive change propagation trees than that of Team B. Thus, any change within Team A's product would likely involve a large amount of re-work by the engineers in order to accommodate it. In contrast, Teams B's DSM reveals a greater distribution of propagation trees across the product models. This suggests that the majority of changes to the product models would not require extensive re-work across the entire product (Insight 3).



(a) Team A *(b) Team B*

Figure 6: Potential change propagation across the product models

Table 3 provides further details on the distribution of the degree of change that could be expected. The difference in the mean value for the degree of change across product models between the two teams further confirms that Team B's design is impacted less by a change to a product model than Team A. Although Team B's DSM shows a significant decrease in the mean, the standard deviation for the distribution slightly increases, which means that the change propagations trees vary more across the product models and sub-systems. Therefore, greater awareness and understanding of the propagations for each product model is required whilst Team A would expect considerable re-work to be required no matter, which product model was altered.

Table 3: Change Propagation Statistics

Team	A	B
Mean	4.32	2.65
Standard Deviation	1.12	1.27
Minimum	1.00	1.00
Maximum	5.48	4.55

Figure 7 shows the distribution of the top 30 product models that are most likely to change as a consequence of a change in another product model, it can be seen that the product models for Team A's design span a range of sub-systems whilst the most likely models to change from Team B's design are exclusively from the Frame & Body and Engine & Drivetrain. As the Frame & Body models form the external structure of the design, this may indicate that Team B's design was such that any internal changes within the design could be accommodated by the Frame & Body alone whilst Team A's design required more extensive cross sub-system alterations to accommodate changes. Although this study presents only two cases, it does suggest that in the context of a FS project, a more successful design is achieved through a product that is more amenable to change and where the impact of changes relate to the Frame & Body. It is arguably also the case that the affordance of having the Frame & Body accommodating the change is that it is can be modified more easily at the later stages of the project (Insight 4).

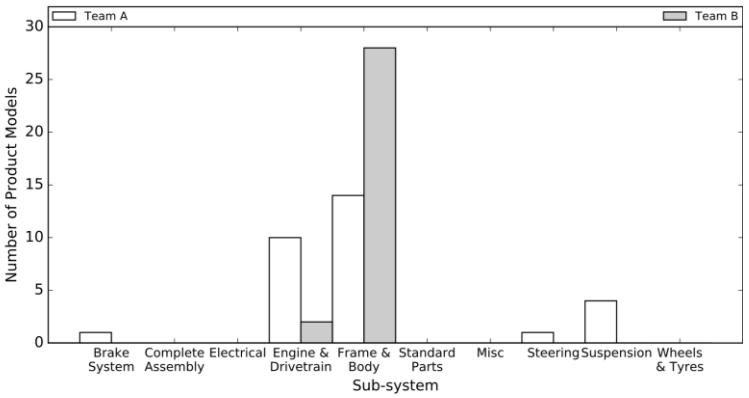


Figure 7: Model most likely to change as a consequence of a change in another model

4.3 DSM Evolution

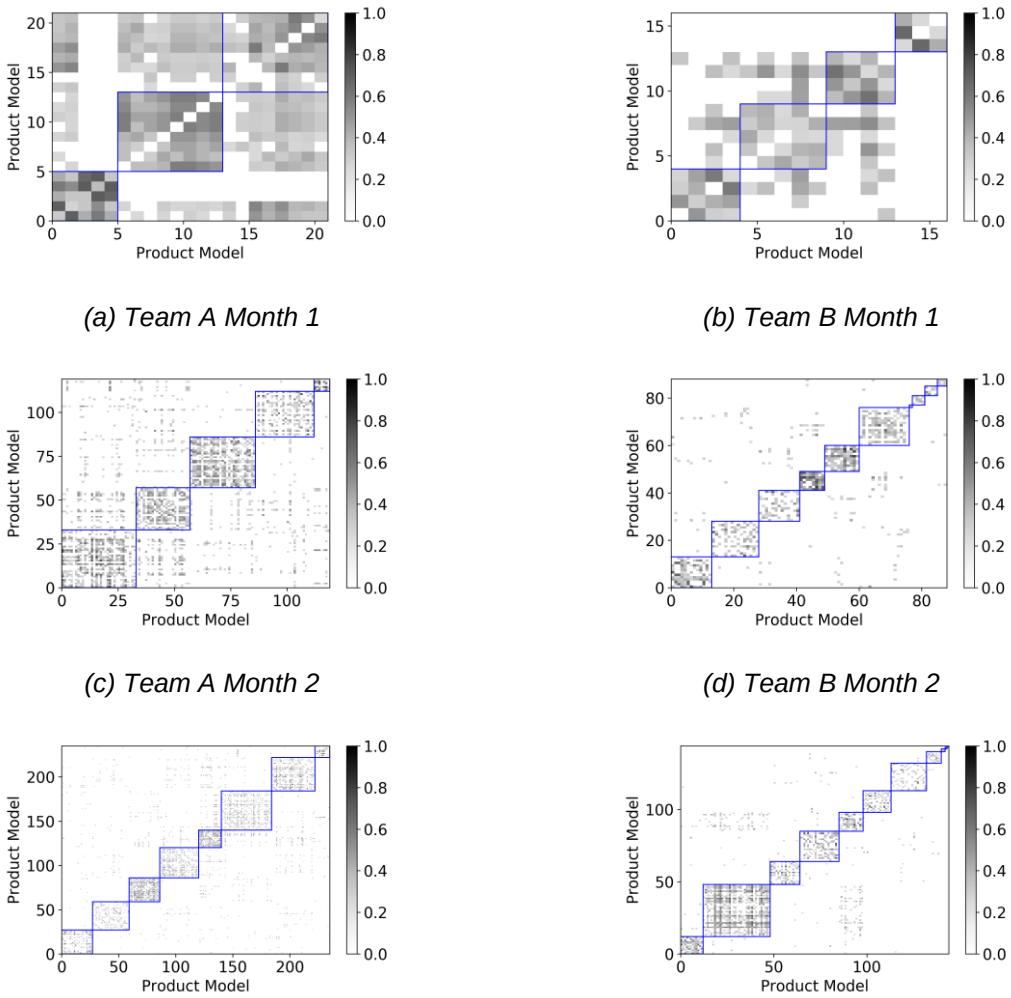
A particular affordance of automatically generated DSMs is that they can be produced in real-time. To illustrate this and the potential insights that it could be inferred from them, DSMs for the first three months of both projects have been generated (Figure 7 & Table 4). On the outset (7a & 7b), there appears to be little difference in the DSMs that have been generated between the two teams, however, closer inspection of the associated metrics generated from the analysis reveal that Team B's design already contains two separate components with four partitions of high-interrelated product models whilst Team A's has one component with three partitions. The striking difference between the two teams' DSMs on the first month lies in the modularity (i.e. level of structure within the current product architecture) where Team B's DSM shows a greater level of modularity to that of Team A.

Table 4: DSM Evolution Statistics

	Team A			**Team B**		
Month	M	N_p	N_c	M	N_P	N_c
1	0.18	3	1	0.31	4	2
2	0.47	6	1	0.70	10	3
3	0.60	8	1	0.57	12	4

Continuing into month two, both teams see an increase in the number of partitions being formed as well as an increase in modularity for the matrix, which suggests the product models are beginning to distinguish and associate themselves with one another to form the final design. The difference again lies in the level of structure exhibited between the two teams as well as the number of partitions and components that exist. In the case of the Team B, both the number of partitions and components has increased, which demonstrates that the team is keeping to a more modularised design to that of Team A.

In month three, Team A's DSM equals Team B's in terms of modularity which suggests that both teams are converging on their final product designs and that a modularity of approximately 0.6 is to be expected for a fully defined design (Insight 5). However, in contrast to Team A, Team B's modularity has decreased from month two to month three, which may indicate that they have begun work on integrating the sub-systems of product models to form the final design. Even though an element of integration is occurring in Team B, the team continues to increase the number of partitions and components within their DSM whilst Team A's DSM continues to contain a single component. This may indicate that if the design does not diverge initially into a set of components and partitions then it may become more difficult to introduce them later in the process. Conversely, having a number of components and partitions at the beginning of the project may promote further modularisation of the product as it evolves (Insight 6). This may also indicate the differences in underlying design strategies that the two teams have taken.



(a) Team A Month 1

(b) Team B Month 1

(c) Team A Month 2

(d) Team B Month 2

(e) Team A Month 3

(f) Team B Month 3

Figure 7: Evolution of the DSM for the two Formula Student teams

5 CONCLUSIONS AND FUTURE WORK

The drive to remain competitive and produce highly innovative products has led to engineering projects with an ever-increasing number of dependencies and inter-dependencies across both the product, supply-chain, and organisations involved. Such is the extent of these dependencies, it has become almost impossible for engineers to manually maintain, manage and monitor all of them and for these reasons techniques such as Design Structure Matrices (DSMs) have been developed. With the recent paradigm shift towards automatically generating DSMs in real-time, there now exists opportunities to monitor and investigate how they evolve throughout an engineering project and develop learnings and insights through cross-project comparison.

To investigate this opportunity, this paper has compared the final DSMs, change propagation characteristics and evolution of DSMs for two Formula Student projects. The investigation revealed six main insights that could be used as indicators for quality of the design, project performance and ultimately success in the FS competition. These were:

1. The more successful FS project generated a more modular product architecture.
2. The partitions of the DSM of the more successful FS project aligned more closely with the manual sub-system categorisation of product models.
3. The DSM of the more successful team is less perturbed by changes to product models.
4. The DSM of the more successful team showed that a change in a product model is more likely to be accommodated by a change in the product models of the Frame & Body.
5. Both teams DSMs converged on a common modularity indicating that product maturity may be

related to the modularity of a DSM.

6. The more partitions and components created in the early stages of a project, the greater the opportunity for further partitioning and components to be formed as a project progresses.

Although insights and learnings have been generated from the comparison of the DSMs of the two teams, three areas of future work have been identified to fully realise the potential of automatic DSM analysis to support engineering projects. First, is the need to verify and validate the insights and learnings that have been generated as well as how context dependent the learnings and insights are. This is currently being achieved through further data capture of FS teams product model edit histories, which will provide further DSMs that can be compared. In addition, further secondary data on the design intent of the team will captured to allow for a more detailed comparison of the DSMs with respect to design strategies. Second, given these learnings and insights, opportunities now exist in providing this information to project managers at the start and throughout the project. How this information may affect future project performance remains unclear and is fundamental to understanding the utility of such analyses. And third, these historic DSMs could be used to facilitate the supervised learning of agent-based models that could simulate the product model editing processes of an engineering project. From this, predictions and optimisations of the design process could be potentially generated for similar products and fed back to future design teams.

ACKNOWLEDGEMENTS

The work reported in this paper has been undertaken as part of the Language of Collaborative Manufacturing Project at the University of Bath & University of Bristol, which is funded by the Engineering and Physical Sciences Research Council (EPSRC), grant reference EP/K014196/2.
Underlying data are openly available from the University of Bath Research Data Archive at http://researchdata.bath.ac.uk/id/eprint/351 or https://doi.org/10.15125/BATH-00351.

REFERENCES

David Briggs. Establish digital product development (dpd) low end viewer (lev) and archival standard for 787 project. In Collaboration & Interoperability Congress (CIC), 2012.

Chris Bruce. Toyota recalls 625k Prius models for faulty hybrid software, 2015. URL http://www.autoblog.com/2015/07/15/toyota-recalls-625k-prius-faulty-hybrid-software/.

Calleam. Airbus - a380, 2011. URL http://calleam.com/WTPF/?p=4700. Steven D Eppinger. A planning method for integration of large-scale engineering systems. In International Conference on Engineering Design, pages 199–204, 1997.

Clarkson, P.J., Simons, C. and Eckert, C., 2004. Predicting change propagation in complex design. Journal of Mechanical Design, 126(5), pp.788-797.

Eppinger, S.D. and Browning, T.R., 2012. Design structure matrix methods and applications. MIT press.

J.A. Gopsill, C.M. Snider, C. McMahon, and B.J. Hicks. Automatically generated design structure matrices: a comparison of two formula student teams. AIEDAM. 2016.

Carlos Gorbea, Tobias Spielmannleitner, Udo Lindemann, Ernst Fricke, et al. Analysis of hybrid vehicle architectures using multiple domain matrices. In DSM 2008: Proceedings of the 10th International DSM Conference, Stockholm, Sweden, 11.-12.11. 2008, 2008.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (SciPy2008), pages 11–15, Pasadena, CA USA, August 2008.

D. E.Jones, Y. Xie, C. McMahon, M. Dotter, N. Chanchevrier, and B. Hicks. Improving Enterprise Wide Search in Large Engineering Multinationals: A Linguistic Comparison of the Structures of Internet-Search and Enterprise-Search Queries, pages 216–226. Springer International Publishing, Cham, 2016. URL http://dx.doi.org/10.1007/978-3-319-33111-9 20.

M. E. J. Newman. Modularity and community structure in networks. Proceedings of the National Academy of Sciences, 103(23):8577–8582, 2006. URL http://www.pnas.org/content/103/23/8577.abstract.

Mark EJ Newman. Analysis of weighted networks. Physical Review E, 70(5): 056131, 2004.

Pasqual, M.C. and de Weck, O.L., 2012. Multilayer network model for analysis and management of change propagation. Research in Engineering Design, 23(4), pp.305-328.

R. R. Senescu, A. W. Head, M. Steinert, and M. A. Fischer. Generating a network of information dependencies automatically. In 14th International Dependency and Structure Modelling Conference, DSM12, 2012.

Manuel E Sosa, Steven D Eppinger, and Craig M Rowles. Identifying modular and integrative systems and their impact on design team interactions. Journal of mechanical design, 125(2):240–252, 2003.

D.V. Steward. The design structure system: A method for managing the design of complex systems. Engineering Management, IEEE Transactions on, EM- 28(3):71–74, Aug 1981. doi: 10.1109/TEM.1981.6448589.