26th INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2024
STUTTGART, GERMANY, 24 – 26 September, 2024

Generating a Design Structure Matrix with a Large Language Model

Edwin C.Y. Koh

Design and Artificial Intelligence Programme | Engineering Product Development Pillar
Singapore University of Technology and Design

Keywords: large language model, retrieval augmented generation, dependency modelling, design automation

1 Introduction

The Design Structure Matrix (DSM) has been applied as an analysis tool in various areas including the modelling of product, organization, and process architecture [Eppinger and Browning 2012]. DSM is known for its simplicity and conciseness in representation and exists in the form of a N × N square matrix that maps the relationships between the set of N system elements [Yassine and Braha 2003]. This implies that the set of N system elements and the relationships between them are necessary data to be collected for DSM analysis.

A series of DSM use cases documented in [Eppinger and Browning 2012] reveals that data collection is traditionally carried out through surveys and interviews. For instance, Clarkson, Simons, and Eckert [2012, pp58] present a DSM use case at an industry partner and stated that chief engineers, deputy chief engineers, and 17 senior engineers were involved in providing inputs to generate a DSM of a helicopter for design change evaluation. Suh and de Weck [2012, pp 43] describe another DSM use case at a different organization and indicated that 140 person hours were spent in creating a DSM of a digital printing system consisting of 84 components. Sosa et al. [2012, pp 105] discuss the use of an organization architecture DSM at an aerospace company to identify communication patterns between and within design teams and revealed that a four-month period was used to interview lead engineers of different teams to generate the DSM. The examples underline how time-consuming and costly it can be to put together a DSM manually.

This work presents a workflow that uses a Large Language Model (LLM) to support the generation of DSM. The proposed workflow, referred to as Auto-DSM, seeks to identify system components (i.e. DSM headings) and the relationships between them (i.e. DSM entries) by querying prescribed data (e.g. proprietary data) through an LLM. A prototype of the workflow was developed in this work to examine its feasibility and a no-code version is made available online: https://drive.google.com/file/d/18TIXWhzZGz6z9YSgVEFIAnB1qSUcpAH0/view?usp=sharing

2 Previous work

Applications of Design Structure Matrix (DSM) are well documented in research publications as reported by Eppinger and Browning [2012]. DSM approaches are also discussed in topic-specific reviews, such as the review on process improvement by Wynn and Clarkson [2018] and the review on engineering systems changeability by Sullivan et al [2023]. A detailed review on DSM is therefore not reproduced here. Instead, this section focuses the discussion on the automation of DSM generation.

The pioneering work of Dong and Whitney [2001] is recognized as the first to attempt the automation of DSM generation. The work builds on design theory to generate DSMs by extracting the relationships between design parameters and functional requirements. A method that specifically uses a language-based approach for DSM generation was introduced by Wilschut et al. [2018], which utilizes design-related text written in a specific grammatical structure by trained personnel. The work highlights the potential of using natural language in DSM generation and offers insights on how similar research can be adapted. For example, Sarica et al. [2020] present a technology semantic network (TechNet) that builds on patent data to support the retrieval of unique relational knowledge between entities. These language-based methods were not applied in DSM generation but have the potential to be extended to establish the dependencies between system elements in a DSM. An extensive review on the use of natural language processing in design is available in [Siddharth et al. 2022]. To date, the utilization of a Large Language Model (LLM) to automate DSM generation has not yet been documented.

3 Auto-DSM: Automated DSM generation using an LLM

This work explores the use of an off-the-shelf LLM to query prescribed data, such as organization-specific proprietary data, for DSM generation. The entire workflow is illustrated in Figure 1. In the first instance, relevant data in the form of a document is split into smaller parts for text embedding and stored in a vector store. A set of 2 predefined questions pertinent to DSM generation will subsequently be used to query the vector store to retrieve relevant splits that may contain answers to the questions posed. The first question queries the vector store to identify the system elements of the system analyzed – "Identify the main {product} components that make up a {product}. Output the answer as a list and a python list. Do not output anything else. If you don't know the answer, strictly state **I don't know** instead of making up an answer." The variable {product} can be any product system of interest to be inserted into the question (e.g. replace with "engine" to create a DSM of an engine). Once the list of system elements is identified, each element will be sequentially fed into the second question to query the relationships between the system elements found – "Are {Element A} and {Element B} {linkage-type} linked? State **Yes** or **No**. If you don't know the answer, strictly state **I don't know** instead of making up an answer." For this question, the entries for {Element A} and {Element B} are systematically inserted based on the list derived from the first question. The variable {linkage-type} can be defined to specifically examine a linkage type (e.g. replace with "mechanically" to examine mechanical links) or left empty to identify unspecified links. Finally, a DSM is constructed based on the answers derived from the first question (i.e. DSM headings) and the answers derived from the second question (i.e. DSM entries).

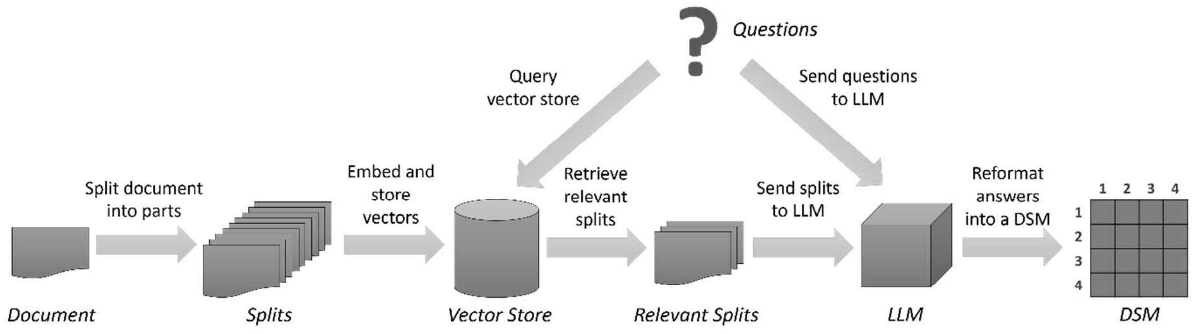
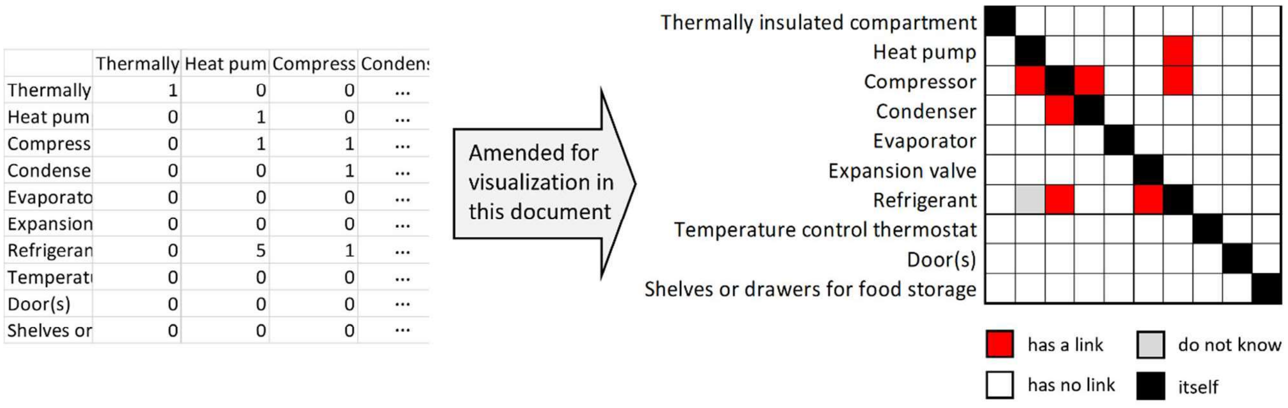


Figure 1. A workflow that uses an LLM to establish DSM headings and entries based on organization-specific proprietary data.

In this work, a no-code executable (.exe) prototype was developed to demonstrate and examine the proposed workflow. The LangChain library 'langchain.text_splitter' is used for text splitting, where each split is set as 1000 characters with an overlap of 150 characters with the next split (see https://python.langchain.com/docs/get_started/introduction.html for further information on LangChain). Next, embedding will be done using an OpenAI Large Language Model (LLM) through 'langchain.embeddings.openai' and stored in a vector store through 'langchain.vectorstores'. Splits that are relevant to the 2 predefined questions will be retrieved using the 'RetrievalQA' function from 'langchain.chains', where retrieval is based on 'similarity' and four splits will be retrieved in each retrieval (i.e. same as the default setting). The retrieved splits and the questions are processed using the 'ChatOpenAI' function from 'langchain.chat_models', which will access the OpenAI 'gpt-3.5-turbo' LLM with 'temperature' set as '0' to produce near identical results in repeated queries. The results produced (i.e. DSM headings and entries) will subsequently be arranged in a DSM format. The DSM headings identified will be kept unmodified. However, the DSM entries produced will be converted with '1' representing a **Yes** answer from the second question, '0' representing **No**, and '5' representing **I don't know**. All diagonal entries will be assigned as '1'. Lastly, a comma-separated values (.csv) file of the DSM generated will be created as shown in Figure 2.



	Thermally	Heat pum	Compress	Conden:
Thermally	1	0	0	...
Heat pum	0	1	0	...
Compress	0	1	1	...
Condense	0	0	1	...
Evaporato	0	0	0	...
Expansion	0	0	0	...
Refrigeran	0	5	1	...
Temperatı	0	0	0	...
Door(s)	0	0	0	...
Shelves or	0	0	0	...

Amended for visualization in this document
Thermally insulated compartment
Heat pump
Compressor
Condenser
Evaporator
Expansion valve
Refrigerant
Temperature control thermostat
Door(s)
Shelves or drawers for food storage
has a link
do not know
has no link
itself

Figure 2. Original csv version from prototype (left) and resized square version in this document (right).

4 A test case based on a diesel engine example

This section documents a test case that examines the feasibility of using Auto-DSM in generating a DSM of a diesel engine. A diesel engine was used in this work as its components (i.e. system elements) and their interdependencies are well documented, allowing comparisons to be made. Five settings with different combinations of input data used in Auto-

DSM were examined in this work. Setting 1 uses a handbook on the design and development of heavy-duty diesel engines [Lakshminarayanan and Agarwal 2020] as the input data for Auto-DSM to examine the scenario of using a rich input dataset (also known as [1] in this document). Setting 2 uses another handbook on diesel engines [Mollenhauer and Tschoeke 2010] as a comparison (aka [2]). Setting 3 uses 1,634 YouTube comments on diesel engines documented by [Koh 2022] to examine the use of unstructured lower quality input data (aka [3]). Setting 4 examines the merits of expanding the size of input data by combining the two handbooks as the input data used [Lakshminarayanan and Agarwal 2020; Mollenhauer and Tschoeke 2010] (aka [1, 2]), while Setting 5 further appends YouTube comments documented by [Koh 2022] to examine the effect of adding lower quality input data to the dataset (aka [1, 2, 3]). A python implementation of ChatGPT (gpt-3.5-turbo, temperature=0) was also used in this work to examine the value-add in using Auto-DSM instead of using an LLM directly. A manually generated diesel engine DSM documented by Keller et al. [2005] was used in this work as a reference (aka [4], see Figure 3). Figure 4 shows the results generated when components were set as [4].

Component List from [4]	
1 Cylinder Head Assembly	12 Fuel Injection Assembly
2 Cylinder Block Assembly	13 Adapter Plate & Flywheel Housing
3 Piston & Rings & Gudgeon Pin	14 Flywheel & Ring Gear
4 Connecting Rod	15 Starter Motor
5 Crankshaft & Main Bearings	16 Sump
6 Valve Train	17 Oil Pump
7 Cam Shaft	18 Gear Train
8 Push Rods	19 Turbocharger
9 High Pressure Fuel Pipes	20 Aircharge Cooler
10 Engine Control Module	21 Exhaust Manifold
11 Fuel Pump	22 Wiring Harness

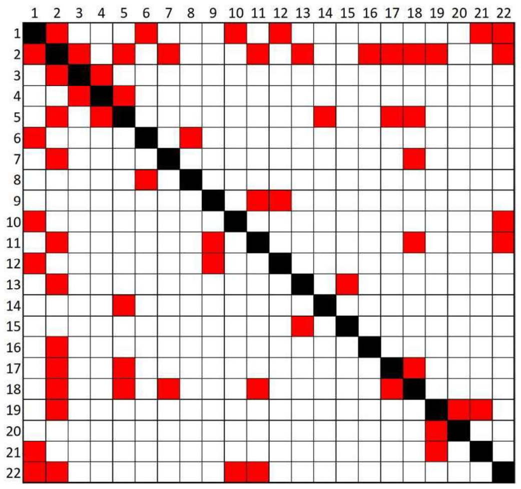


Figure 3. A DSM of a diesel engine reproduced from [4], showing mechanical links (see Keller et al. [2005]).

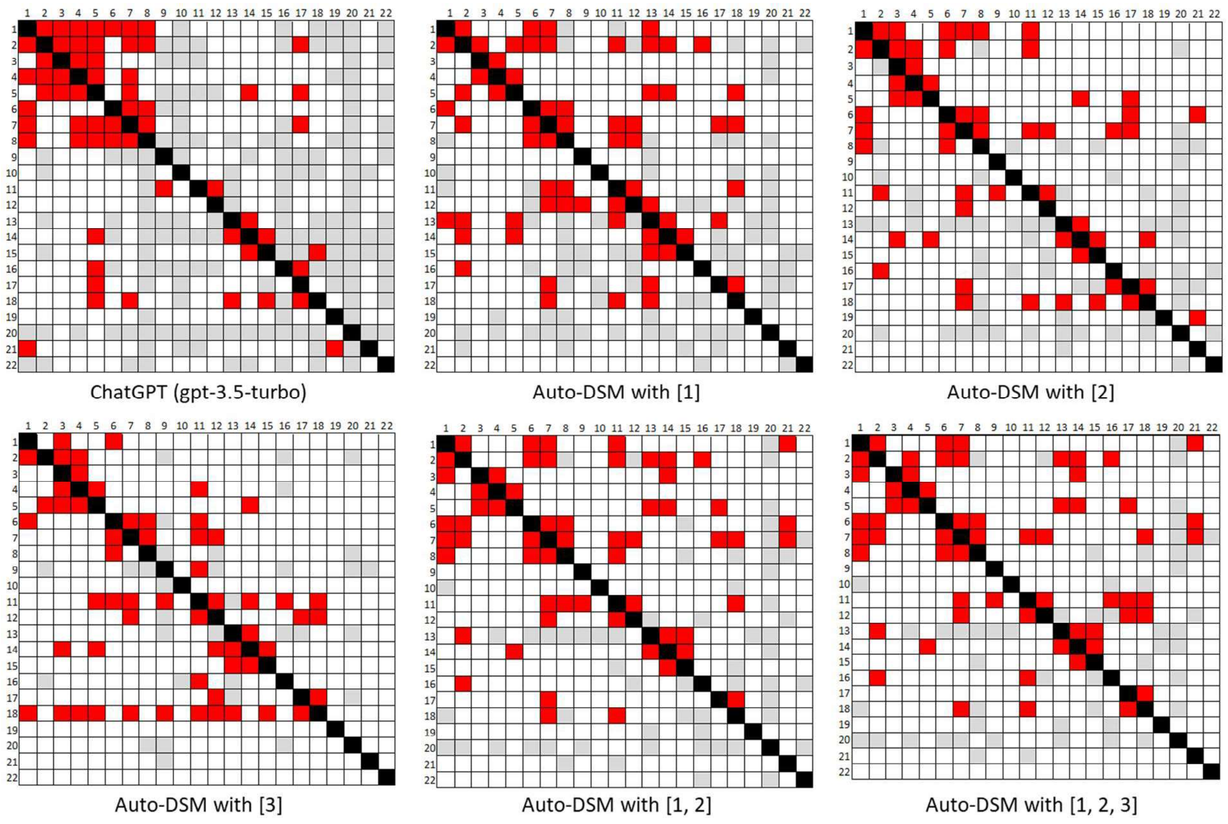


Figure 4. DSMs generated with ChatGPT and Auto-DSM based on engine components from [4].

Table 1 shows a summary of the results. Given that the DSM in [4] has 22 components (i.e. a 22 x 22 DSM), there are 462 DSM entries in [4] labelled as either ‘has a link’ or ‘has no link’. Out of these 462 entries, Auto-DSM with [3] produced the most entries with either a ‘has a link’ or a ‘has no link’ label (i.e. 430 entries, 93.1%). This implies that it has the fewest entries labelled as ‘do not know’ when compared with the other settings. Auto-DSM with [3] also produced the highest number of entries identical with the DSM in [4] (357 entries, 77.3%). All settings of Auto-DSM outperformed the direct use of ChatGPT (gpt-3.5-turbo, temperature=0) in terms of generating entries with a useful label and in terms of generating identical entries with respect to the reference DSM in [4].

Table 1. Summary of results on DSMs generated based on engine components in [4].

Methods	Entries with a useful label		Identical DSM entries as [4]	
	(Quantity)	(As % of *)	(Quantity)	(As % of ^)
DSM from [4]	*462	100.0%	^462	100.0%
ChatGPT on [4]	305	66.0%	244	52.8%
Auto-DSM with [1] on [4]	379	82.0%	315	68.2%
Auto-DSM with [2] on [4]	398	86.1%	329	71.2%
Auto-DSM with [3] on [4]	430	93.1%	357	77.3%
Auto-DSM with [1, 2] on [4]	401	86.8%	335	72.5%
Auto-DSM with [1, 2, 3] on [4]	409	88.5%	342	74.0%

5 Closing remarks

This document presents the Auto-DSM workflow, which uses a Large Language Model (LLM) to support the automation of DSM generation. The goal is to improve productivity as manual DSM generation can be time-consuming and costly. A prototype of the workflow was developed in this work and a test case involving a diesel engine was carried out to examine its feasibility. The results generated show promise with Auto-DSM reproducing as high as 77.3% of DSM entries that are identical to those manually generated by experts. A no-code version of the prototype is available online to serve as a construct for future research and to support industry adoption. An extended discussion of this work is available in Koh [2023].

References

Clarkson PJ, Simons C and Eckert C (2012) Agusta Westland Helicopter Change Propagation. In Eppinger SD and Browning TR (Eds.), Design structure matrix methods and applications, 58-62. MIT press. Cambridge, MA.

Dong Q and Whitney D (2001) Designing a requirement driven product development process. In: Proceedings of the ASME international conference on design theory and methodology, Pittsburgh, PA, 1–11.

Eppinger SD and Browning TR (2012) Design structure matrix methods and applications. MIT press. Cambridge, MA.

Koh ECY (2022) Design change prediction based on social media sentiment analysis. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 36, e24, 1-16.

Koh, ECY (2023) Using a Large Language Model to generate a Design Structure Matrix. arXiv:2312.04134. Preprint. 1-16.

Keller R, Eger T, Eckert CM and Clarkson PJ (2005) Visualising change propagation. In: Proceedings of the International Conference on Engineering Design, Melbourne, 1-12.

Lakshminarayanan PA and Agarwal AK (2020) Design and development of heavy duty diesel engines: A handbook. Springer Singapore.

Mollenhauer K and Tschoeke H (2010) Handbook of diesel engines. Springer-Verlag Berlin Heidelberg.

Sarica S, Luo J and Wood K (2020) Technet: technology semantic network based on patent data. Expert Systems with Applications 142, 112995.

Siddharth L, Blessing L and Luo J (2022) Natural Language Processing in-and-for design research. Design Science 8, e21, 1-76.

Suh ES and de Weck O (2012) Xerox digital printing technology infusion. In Eppinger SD and Browning TR (Eds.), Design structure matrix methods and applications, 43-48. MIT press. Cambridge, MA.

Sullivan BP, Nava EA, Rossi M and Terzi S (2023) A systematic literature review of changeability in engineering systems along the life cycle. Journal of Engineering Design 34(12), 1046-1098.

Sosa M, Eppinger S and Rowles C (2012) Pratt & Whitney jet engine development. In Eppinger SD and Browning TR (Eds.), Design structure matrix methods and applications, 105-108. MIT press. Cambridge, MA.

Wilschut T, Etman LFP, Rooda JE and Vogel JA (2018) Generation of a function-component-parameter multi-domain matrix from structured textual function specifications. Research in Engineering Design 29(4), 531-546.

Wynn DC and Clarkson PJ (2018) Process models in design and development. Research in Engineering Design 29, 161-202.

Yassine A and Braha D (2003) Complex concurrent engineering and the design structure matrix method. Concurrent Engineering 11(3), 165-176.

Contact: Edwin C.Y. Koh, DAI / EPD, Singapore University of Technology and Design. Email: edwin_koh@sutd.edu.sg