**22nd INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2020**
CAMBRIDGE, MASSACHUSETTS, USA, 13 – 15 October, 2020

# Concurrent Modeling Of Positive And Negative Dependencies In The Design Structure Matrix Using Complex Numbers

Ugo Chouinard[1], Yann-Seing Law-Kam Cio[1], Aurelian Vadean[1], Sofiane Achiche[1].

[1]Polytechnique Montréal, Montréal, QC, Canada

**Abstract**: The complexity of an engineering product is based in part on the number of components and their dependencies, where the latter can be desired or undesired. In some cases, where those dependencies exist concurrently, engineers can be misled regarding the design challenge at hand due to ill-adapted representation methods, especially when managing various abstraction levels. To overcome this issue, a new modeling method to concurrently handle positive and negative dependencies is proposed. This paper suggests modeling the dependencies using a complex number notation within a design structure matrix. Using the proposed representation to simultaneously model positive and negative dependencies, shown through an illustrative example, it is possible to differentiate and make use of more information when dealing with different abstraction levels. Finally, the paper discusses implications related to using the modeling method regarding the system analysis.

*Keywords: Design Structure Matrix, Complexity Management, Systems Modeling*

## 1 Introduction

One of the major challenges while designing complex systems is to handle the multiple dependencies inherent to these systems. The dependencies in a system are defined as the relationship that exists between two elements (components, subsystems, systems) whenever one of them is affected by the other. The affecting element is usually referred to as the antecedent while the affected one being the dependent (Keller et al., 2000). However, a dependency is not limited to components or systems. Indeed, for a system a dependency can be defined between (Torry-Smith et al., 2014): functions (e.g. provide power), means (e.g. batteries), and properties (e.g. discharge time).

A dependency can also be said to be positive or negative. A positive dependency is a desired effect which will help fulfill the functional requirements of the system. For instance, the dependency between the battery (a means) and a motor (another means) is said to be positive as it works towards achieving the functional requirement (motion) of the system. Negative dependencies are undesired effects or constraints and can occur in various forms. One of the common types would be the noise (heat, vibration, electromagnetic field) induced by functioning components.

It is mentioned by Pimmler and Eppinger (1994) that design challenges are related to both positive and negative dependencies, and negative dependencies should thus be accounted

for to avoid for new problems to appear. Indeed, negative dependencies will usually deteriorate a system's performance, and hence should be considered during the system's modeling and analysis (Torry-Smith et al., 2015, 2014). Nevertheless, one challenge related to negative dependencies is that they will often occur concurrently with positive ones. For instance, it is mentioned by Sosa, Eppinger, and Rowles (2003) that a transfer of vibration between low-pressure turbine vanes and blades would be detrimental, but the same vanes and blades would be positively dependent on their closeness for achieving proper turbine efficiency. Thus, considering both positive and negative dependencies should prove to be relevant for the design of complex systems.

However, concurrently handling positive and negative dependency is not an easy task. Apart from the fact that it is difficult to first identify the negative dependencies, the methods employed are not usually fitted to manage them. Indeed, negative dependencies will often be represented as negative numbers, which may cause issues for instance during DSM clustering for product module identification (Williamson and Sellgren, 2019) or may not be considered at all during complexity analysis (Chouinard et al., 2019). Consequently, a new approach is proposed to better handle negative dependencies during the modeling process which would necessarily impact the subsequent step in the system design process. With a better modeling of positive and negative dependencies simultaneously, the designers will be able to better assess the compromise between working on solving issues related with negative dependencies and working on maximizing the benefits related with positive dependencies.

This paper first briefly reviews different dependency modeling methods. Then, a new approach to represent concurrently positive and negative dependencies is proposed by using complex numbers. A discussion is finally done on how this modeling method may be employed in the future.

## 2 Related Work

Managing dependencies is a challenging task in complex systems. One of the first steps to manage them is to be able to carry out efficient and effective modeling. One of the widely used method is the Design Structure Matrix (DSM) (Steward 1981; Browning 2001; Browning 2016). The DSM is a compact form of representing dependencies between elements. It can be used for various applications such as to model system architecture, organization structures, processes and low-level relationships (Browning 2001).

Furthermore, there exists modeling methods such as proposed by Pimmler and Eppinger (1994) that help identifying the dependencies (spatial, energy, material, information) and assessing whether they are detrimental, undesired, indifferent, desired or required for the functionality of the system, such as shown in Figure 1. Other methods such as proposed by the authors in Chouinard et al. (2017) help to assess negative dependencies and build DSMs by evaluating relevant dependency dimensions using fuzzy logic. Alternatively, the four-point scale by Sharman and colleagues (2002; 2004) shown in Table 1 can be used to represent if there are significant dependencies (spatial, energy, information, material) between pairs of elements of a system. The strength of the dependency can thus be used as the input between two elements of the DSM. Using this method does not require to assess

each dependency types individually, but only to identify the number of dependencies. This thus allows to have single entries in the DSM.

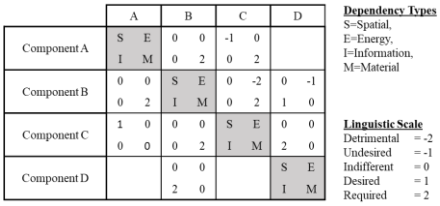



Figure 1. Design Structure Matrix Built Using the Method in (Pimmler and Eppinger, 1994)

Table 1: Four-Point Scale such as presented by Sharman and colleagues (2002; 2004)

| Strength | Title | Description |
|---|---|---|
| 3 | High | Significant flow of three or more of the dependency types |
| 2 | Medium | Significant flow of two of the above |
| 1 | Low | Significant flow of one of the above |
| 0 | Zero | No significant Relationship |

Moreover, Tilstra, Seepersad and Wood (2012, 2010) suggest the use of High-Definition Design Structure Matrix (HDDSM), to represent more specific types of dependencies within a system, rather than the generic types proposed by Pimmler and Eppinger (1994). An example of a specific type would be “Status” for the “Information” dependency. This modeling method enables one to have a more detailed view of the system that is being developed, and potentially result in better analysis and module creation.

Although different modeling methods exist, they do not allow the effective management of positive and negative dependencies. Indeed, the modeling methods do not permit to identify both a positive and negative dependency when using higher abstraction in the modeling, which is often the level that is used when analyzing the system’s complexity or creating modules. Furthermore, the previously mentioned methods do not deal with the case where positive and negative dependency of the same type exist, as it may often be the case. To deal with these issues, the following section presents a new modeling method that allows to concurrently handle positive and negative dependencies, and which will allow to consider this duality in the analysis.

## 3 Concurrent Modeling of Positive and Negative Dependencies

### 3.1 Proposed Approach: Complex Numbers

Dealing with positive and negative dependencies concurrently is a tedious task to achieve, as two components might have both positive and negative dependencies between them. To deal with this issue, complex numbers $a+bi$ are proposed to define dependencies between two elements of a system. The real part $a$ is used to represent the positive dependency, and the imaginary part $b$ represents the negative dependency. This should allow during the design process to track the evolution of both desired and undesired effects.

During the modeling process the variables $a$ and $b$ would take the value resulting from the assessment of the designer. If a dependency is identified, then $a,b$ would take the binary values 0 or 1, depending on whether there exist or not a positive/negative dependency between two elements. Alternatively, it would also be possible to have the values $a,b$ representing the strength of the dependency on a scale, such as in a range between 0 to 1. It is also possible to adapt currently existing assessment methods. For instance, Table 2 shows how the scaling could be done in comparison to the method proposed by Pimmler and Eppinger (1994). It would also be possible to adapt methods that deal with multiple dependency types such as the four-point scale by Sharman and colleagues (2002; 2004). The adapted four-point scale to a seven point one is shown in Table 3. It would then be possible to input in the DSM a combination of the positive and negative dependency strengths depending on the number of dependencies that works towards fulfilling or impairing the functional requirements. For instance, if there are 1 positive and 3 negative dependencies between two elements, then the corresponding input in a DSM would be 1+3$i$.

Table 2: Comparison of linguistic terms using the scale from Pimmler and Eppinger (1994) and the proposed complex scale

| Linguistic term | Detrimental | Undesired | Indifferent | Desired | Required |
|---|---|---|---|---|---|
| Pimmler and Eppinger (1994) | -2 | -1 | 0 | 1 | 2 |
| Complex Scale | 0+2$i$ | 0+$i$ | 0+0$i$ | 1+0$i$ | 2+0$i$ |

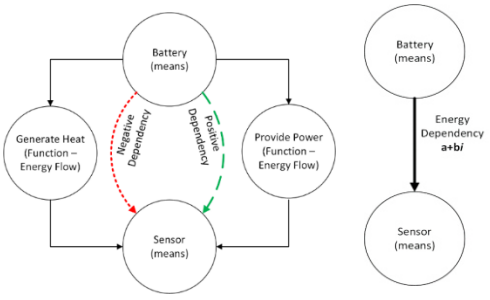
Table 3. Adapted Four Point Scale to Seven Point Scale

| Type | Strength | Title | Description |
|---|---|---|---|
| Positive Dependencies | 3+0$i$ | Positive High | Significant flow of three or more of the dependency types that contribute meeting the functional requirements |
| | 2+0$i$ | Positive Medium | Significant flow of two of the above |
| | 1+0$i$ | Positive Low | Significant flow of one of the above |
| | 0+0$i$ | None | No significant Relationship |
| Negative Dependencies | 0+1$i$ | Negative Low | Significant flow of one of the below |
| | 0+2$i$ | Negative Medium | Significant flow of two of the below |
| | 0+3$i$ | Negative High | Significant flow of three or more of the dependency types that impairs meeting the functional requirements |

### 3.2 Handling Varying Abstraction Level

The use of the two-dimensional dependency representation should ease the modeling depending on the level of detail desired. For instance, defining an energy dependency could

require defining what specific type of energy dependency it is (electrical, thermal, etc.). Then when modeling the system, part of the dependency could be positive, such as a battery providing power to a temperature sensor (electrical) and other part negative, such as the battery inducing heat to the same sensor (thermal). Indeed, the sensor needs the battery to fulfill its function, however, the heat generated from the battery can cause erroneous reading and compromise the system. Depending on the level of abstraction such as only considering the more general dependency types (spatial, energy, information, material), using a two-dimensional dependency representation could allow to characterize both a positive and negative dependencies within a single DSM. Figure 2 illustrates how from a detailed description of the energy dependency (Figure2-(a)) between a battery and a sensor, it is possible to find its complex representation (Figure2-(b)) and transfer it to the DSM (Figure2-(c)).




| | Battery | Sensor |
|---|---|---|
| Battery | X | a+b*i* |
| Sensor | | X |

(a) (b) (c)

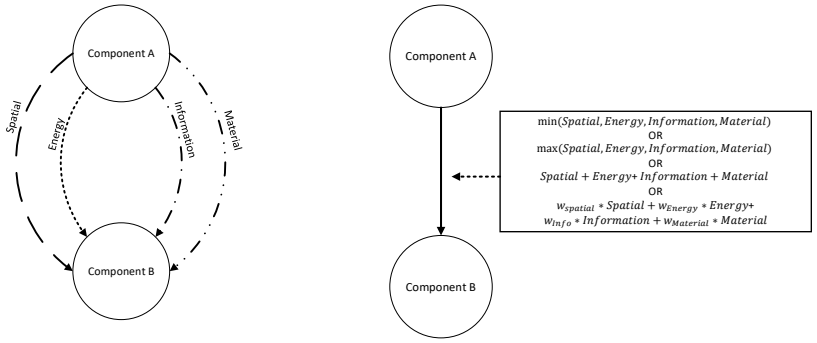
Figure 2. (a) Specific Energy Dependencies (b) General Energy Dependency Representation (c) DSM of the General Dependency

Using the complex number representation could also ease the management of multiple dependency types. Indeed, different types of dependencies could exist between a pair of components/subsystems such as the exchange of energy, and the required physical proximity for achieving functionality. Using a DSM with the method suggested by Pimmler and Eppinger (1994) or Tilstra, Seepersad and Wood (2012) (i.e. having lower abstraction in the dependency) would result in a multigraph. A multigraph, as opposed to a simple graph, is a graph where multiple edges are allowed between any pair of nodes. However, when a higher abstraction of the system is desired, it may not be always obvious how to represent it especially if there are negative dependencies. One way would be to only have a binary DSM stating the existence of dependencies at the expense of losing the information about the type of dependency. Alternatively, it would be possible to deal with multiple edges by finding a total edge value between two nodes.

There are different methods that exist to find the total value of the link between the nodes of a multigraph such as using a min or max operator on the edges, or by summing the edges weight (Newman, 2004). Furthermore, a simple weighted aggregation to calculate the total interaction between the nodes might also be used. The weighted aggregation would allow engineers to define which types of dependency makes, for instance, the integration more difficult, and thus leading to better representing the reality of the design. Information exchange would usually be easier to carry out, by passing communication bus between components for example, than solving spatial interaction in a constrained environment.

Helmer et al. (2010) also suggest an aggregation method for DSM clustering where multiple dependency types (structural, energy, signal and material) are assigned a value associated with the effect of spatial adjacency of two components on the system. Then, based on a set of decision rules, a single value among the dependency types will be chosen to represent the dependency between the two components. This method allows to highlight the importance of spatial adjacency between two components to fulfill a function but loses information about the design challenges that need to be resolved.

As another example, Figure 3 shows how the four types of dependencies (spatial, energy, information, material) between two components could be aggregated into a single dependency value when varying the abstraction level. However, no matter how the dependency types are consolidated the result might lead to a loss of information if complex numbers are not used. Indeed, the total value of the edge might result in being null if positive and negative dependencies cancel each other due to the aggregation operator (such as summing the edges). This issue will further be explored in the following section.




(a) (b)

Figure 3. (a) Multiple dependency types between two components (b) an aggregated dependency value

## 4 Illustrative Example

To illustrate how the proposed approach can be employed when dealing with varying abstraction, a subsystem of the climate control system from Pimmler and Eppinger (1994) is used. Figure 4 shows different representations of the "Front End Air Chunk" where the component A, B and C are respectively the radiator, the engine fan and the condenser. The system is modelled using the 4 general dependency types (spatial, energy, information, material) and Figure 4-(a) shows how the system would typically be modelled. Alternatively, Figure 4-(d) shows the same DSM but using the complex notation.

Similarly, Figure 4-(b) and 4-(e) compare the case where a binary DSM is used for modeling the system, without and with complex numbers. A better understanding of the underlying dependencies is obtained in Figure 4-(e) since the engineer would be aware that there would be constraints in the design, which is not the case in Figure 4-(b). Therefore, even though a higher abstraction of the system is used, the engineer will be able to grasp that more time will be required during the design process in order to deal with the constraints, and not only realize the functions of the system.

Finally Figure 4-(c) and (f) compare the result if an aggregation of the dependencies is carried out while changing abstraction, without and with the complex number notation. One of the elements which can be observed in Figure 4-(c) is how the dependency between components *A* and *C* "disappears" when aggregated, which would potentially result in the system being thought as "simpler" than it really is. On the other hand, the aggregation with complex number notation shows no loss of information, which should not result in misleading the engineer during the system analysis. Using the introduced complex number notation, consolidating the dependency types in Figure 4-(f) again retains more information in the DSM as compared to not using it. Indeed, the DSM retains the information that there exists both positive and negative dependencies, and it should be accounted for in future system analyses.

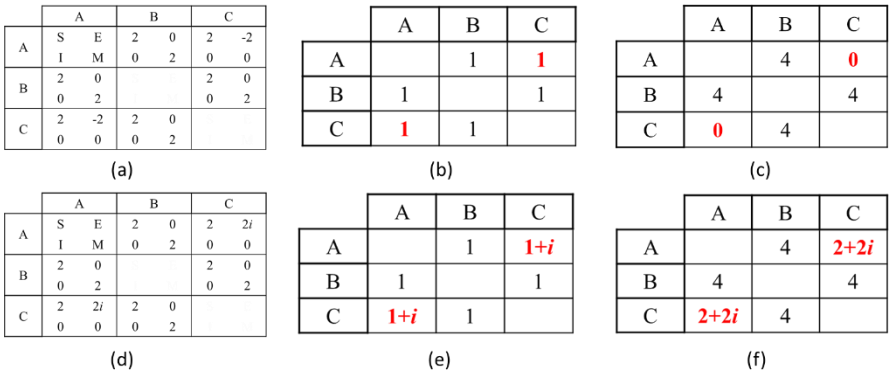



Figure 4. (a) Subsystem DSM, (b) simplified binary DSM ,(c) aggregated DSM (d) subsytem DSM complex number notation, (e) simplified binary DSM with complex numbers (f) aggregated DSM with complex numbers

## 5 Discussion

Although the use of complex numbers to model dependencies has its usefulness when dealing with various abstraction level, it is not its sole purpose. Such representation could also facilitate the integration of negative dependencies during complexity metrics calculation or, even clustering. There are multiple research works that have been carried out to develop new metrics to better represent either integration effort (Sinha et al., 2018; Sinha and de Weck, 2016), rework (Hölttä and Otto, 2005), or modularity (Jung and Simpson, 2017; Tamaskar et al., 2014) just to name a few. However, none of the developed metrics integrate negative dependencies in their calculation. It is clear, however, that these negative dependencies would have an impact on the final design. Consequently, it would be necessary to integrate negative dependencies during the computation of the metrics, which could eventually be achieved through the used of the proposed complex number notation.

Indeed, it is hypothesized that if negative numbers are used to model negative dependencies, then the computation of complexity metrics may result in something similar than what was shown in Section 4; some positive and negative dependencies may cancel each other, and the resulting metric may portrait a system a being less complex. This would

probably not be the case if complex arithmetic is used. A similar hypothesis may be drawn when carrying out other system analysis.

However, doing so might add some complexity in the analysis process as the calculation methods would need to be adapted. For instance, during complexity analysis, many metrics use the graph energy as a basis for calculation. If instead complex numbers are used to represent dependencies, then the energy of the graph computation would need to be adapted (Bhat et al., 2018). As observed in other works such as Helmer et Al. (2010),including negative dependencies in the modelling process implies the need to modify clustering algorithms. Hence, the different clustering algorithms would also need to be adapted to account for complex numbers. Although more research work is needed to fully integrate complex numbers in the design process, the benefit of using them would outweigh the added complexity.

## 6 Conclusion

In this paper, a new way of representing dependencies during system modeling has been introduced. The usefulness of using complex numbers to concurrently represent positive and negative dependencies when dealing with different level of detail has been discussed. Moreover, an illustrative example has further exemplified the limitations of the current system modeling as well as the effectiveness of the proposed approach. However, it was shown that complex number notation also introduces multiple new research questions. Indeed, using the proposed dependency modeling method, new way of analyzing the complexity of systems should thus be developed, or existing method should at least be adapted. Furthermore, including both negative and positive dependencies should change other DSM operators, such as clustering algorithms. This will be considered in future works. Other aspects to consider is the integration of such dependency representation in other type of DSMs, such as during the modeling of process or organizational structures.

## References

Bhat, M.A., Pirzada, S., Rada, J., 2018. On spectra and real energy of complex weighted digraphs. Linear and Multilinear Algebra 1–13. https://doi.org/10.1080/03081087.2018.1529138

Browning, T.R., 2016. Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. IEEE Transactions on Engineering Management 63, 27–52. https://doi.org/10.1109/TEM.2015.2491283

Browning, T.R., 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. IEEE Transactions on Engineering Management 48, 292–306. https://doi.org/10.1109/17.946528

Chouinard, U., Achiche, S., Baron, L., 2019. Integrating Negative Dependencies Assessment During Mechatronics Conceptual Design using Fuzzy Logic and Quantitative Graph Theory. Mechatronics.

Chouinard, U., Achiche, S., Leblond-Ménard, C., Baron, L., 2017. Assessment of dependencies in mechatronics conceptual design of a quadcopter drone using linguistic fuzzy variables. Presented at the 21st International Conference on Engineering Design, Vancouver, Canada, pp. 031–040.

Helmer, R., Yassine, A., Meier, C., 2010. Systematic module and interface definition using component design structure matrix. Journal of Engineering Design 21, 647–675. https://doi.org/10.1080/09544820802563226

Hölttä, K.M.M., Otto, K.N., 2005. Incorporating design effort complexity measures in product architectural design and assessment. Design Studies 26, 463–485. https://doi.org/10.1016/j.destud.2004.10.001

Jung, S., Simpson, T.W., 2017. New modularity indices for modularity assessment and clustering of product architecture. Journal of Engineering Design 28, 1–22. https://doi.org/10.1080/09544828.2016.1252835

Keller, A., Blumenthal, U., Kar, G., 2000. Classification and computation of dependencies for distributed management. IEEE Comput. Soc, pp. 78–83. https://doi.org/10.1109/ISCC.2000.860604

Newman, M.E.J., 2004. Analysis of weighted networks. Physical Review E 70. https://doi.org/10.1103/physreve.70.056131

Pimmler, T.U., Eppinger, S.D., 1994. Integration analysis of product decompositions. Presented at the ASME Design Theory and Methodology Conference, Minneapolis, MN.

Sharman, D.M., Yassine, A.A., 2004. Characterizing complex product architectures. Systems Engineering 7, 35–60. https://doi.org/10.1002/sys.10056

Sharman, D.M., Yassine, A.A., Carlile, P., 2002. Architectural Optimisation Using Real Options Theory and Dependency Structure Matrices, in: Volume 2: 28th Design Automation Conference. Presented at the ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME, Montreal, Quebec, Canada, pp. 799–811. https://doi.org/10.1115/DETC2002/DAC-34119

Sinha, K., de Weck, O.L., 2016. Empirical Validation of Structural Complexity Metric and Complexity Management for Engineering Systems. Systems Engineering 19, 193–206. https://doi.org/10.1002/sys.21356

Sinha, K., Suh, E.S., de Weck, O., 2018. Integrative Complexity: An Alternative Measure for System Modularity. Journal of Mechanical Design 140, 051101. https://doi.org/10.1115/1.4039119

Sosa, M.E., Eppinger, S.D., Rowles, C.M., 2003. Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions. Journal of Mechanical Design 125, 240. https://doi.org/10.1115/1.1564074

Steward, D.V., 1981. The design structure system: A method for managing the design of complex systems. IEEE transactions on Engineering Management 71–74.

Tamaskar, S., Neema, K., DeLaurentis, D., 2014. Framework for measuring complexity of aerospace systems. Research in Engineering Design 25, 125–137. https://doi.org/10.1007/s00163-014-0169-5

Tilstra, A.H., Seepersad, C.C., Wood, K.L., 2012. A high-definition design structure matrix (HDDSM) for the quantitative assessment of product architecture. Journal of Engineering Design 23, 767–789. https://doi.org/10.1080/09544828.2012.706748

Tilstra, A.H., Seepersad, C.C., Wood, K.L., 2010. The Repeatability of High Definition Design Structure Matrix (HDDSM) Models for Representing Product Architecture. ASME, pp. 529–542. https://doi.org/10.1115/DETC2010-28717

Torry-Smith, J.M., Mortensen, N.H., Achiche, S., 2014. A proposal for a classification of product-related dependencies in development of mechatronic products. Research in Engineering Design 25, 53–74. https://doi.org/10.1007/s00163-013-0161-5

Torry-Smith, J.M., Mortensen, N.H., Ploug, O., Achiche, S., 2015. Industrial Application of a Mechatronic Framework. Presented at the Proceedings of the 20th International Conference on Engineering Design (ICED 15), Milan, pp. 247–258.

Williamson, D., Sellgren, U., 2019. Introducing Implementation Dependent Behavior Into Integrated Product Architecture Clustering, in: Proceedings of the 21st International DSM Conference. Presented at the The 21st International DSM Conference, The Design Society. https://doi.org/10.35199/dsm2019.9