THE WHY-MATRIX
Maik Maurer¹, Thomas Braun²

1Institute of Product Development, TUM, Boltzmannstraße 15, 85748 Garching, Germany
2Teseon GmbH, Parkring 4, 85748 Garching, Germany

Keywords: Multiple-Domain Matrix, indirect dependencies, software support

1 INTRODUCTION
The Multiple-Domain Matrix (MDM) allows to model dependencies within and between several domains of a system. The main feature of the MDM application is the deduction of indirect dependencies from acquired direct ones. That means that information about dependencies between elements from two domains (e.g. components and people) can be concentrated in the specific system view of one domain (e.g. dependencies between people due to their work on identical components). Figure 1 shows such an concentration of dependency information: At the left side the direct dependencies between elements of two domains are depicted. The network at the right side shows the derived indirect dependencies in one domain due to the identical connection to elements of the second domain. That means that, e.g., “1” and “2” get connected, because both link on element “I”. As the concentrated view only contains one single domain it is easier for users to understand than networks comprising several types of elements. Such a reduced system view can be seen as DSM-conform model and therefore allows the application of common analysis algorithms [1].

![Figure 1. Direct and indirect system dependencies](image)

A significant disadvantage of the concentrated representation of indirect dependencies within one domain is that users can not see the originating cause of a dependency. This problem is tackled by the “Why-Matrix”, which provides the explication for existing indirect dependencies based on the direct ones. Basically, this matrix represents an enhanced DSM that can be applied for specific information about the source of an actually considered indirect dependency. The functionality of the Why-Matrix has been implemented to the software tool LOOMEO [2].

2 CREATION OF THE WHY-MATRIX

2.1 Conventional representation of indirect dependencies
Generally, it is possible to set up a matrix of indirectly connected system elements and to note the linking causes in the matrix cells. Hereby, the causes mean the system elements that are sited on the path connecting the indirectly linked elements. [3] applied such a notation for “connectivity maps”, which indicate indirect dependencies in Domain-Mapping Matrices (DMMs). Figure 2 shows the exemplary creation of a connectivity map. If two DMMs are apparent that provide the direct links between elements from domain B to domain A and from domain A to domain C, the approach on connectivity maps derives indirect links from elements of domain B to domain C. The figure depicts these elements from domain A in the matrix cells of the resulting DMM that cause the indirect links. In practice, limits of applicability exist for this notation of indirect dependencies. Complex systems often possess a high quantity of indirect dependencies. Thus, matrices representing all indirect dependencies can become difficult to read. As well, indirect dependencies do probably not pass by one further system element only. In fact, many indirect dependencies result from dependency chains...
spanning several system elements. In addition, Figure 3 shows six general possibilities to define indirect dependencies [1]. If these are considered simultaneously, the quantity of indirect dependencies further increases.

Figure 2. Creation of a connectivity map (according to [3])

Figure 3. Possibilities of indirect dependencies (according to [1])

Generally, if comprehensive information (e.g. names of system elements) of a large quantity of indirect links has to be depicted, the notation in matrix cells (see Figure 2) becomes disadvantageous.

2.2 Selective representation of indirect dependencies
A new approach on representing indirect dependencies bases on the fact that not all indirect dependencies of a specific system are inquired simultaneously. Users always concentrate on selected aspects. Basically, two questions are supposable for the application of information about indirect dependencies:

1. Which indirect dependencies (or dependency paths) exist between two specific system components?
2. Which system components are indirectly linked by (paths containing) a specific system component?

For both use cases information from the Why-Matrix can be represented by use of a simple list that complements the matrix of indirectly linked elements. The practical use of selected information from the Why-Matrix requires software support, as dynamic analyses of the considered network are mandatory. If users select specific system elements (case 1) or a dependency (case 2) in the view of indirectly linked elements, individual list of dependency causes have to be promptly generated.
3 SOFTWARE IMPLEMENTATION OF THE WHY-MATRIX

Figure 4 shows two screenshots of the application of the Why-Matrix as implemented in the software LOOMEO. The example comprises a network of indirectly linked product designers. The cause for dependencies between designers arises from their occupation with the same product component. At the left side of Figure 4 a dependency between two designers has been selected (by mouse click). In the second window the cause for the people linkage is depicted (both designers are occupied with the component “differential”), as it can be deduced from the Why-Matrix. At the right side of Figure 4 first the dependency cause has been selected (from a list of all dependency causes). The software implementation then highlights all designers that are linked due to this cause.

![Figure 4. Implementation of the use cases within the software LOOMEO](image)

4 CONCLUSIONS

The Why-Matrix represents a useful information base for aggregated system views. So far, applied applications possess disadvantages in case of larger quantities of indirect dependencies, as information representation becomes complex. Here, the selective representation of individual dependency causes in list form depicts a possible improvement, as typically not all information from the Why-Matrix is required simultaneously. This selective representation of dependency causes has been realized in the software LOOMEO for both basic application scenarios. The implementation has already been applied successfully in several industrial projects.

REFERENCES


Contact: Maik Maurer
Technische Universität München
Institute of Product Development
Boltzmannstraße 15
85748 Garching
Germany
Phone +49 89 3074815-11
Fax +49 89 3074815-29
maik.maurer@pe.mw.tum.de
The Why-Matrix

Maik Maurer, Thomas Braun

Institute of Product Development (TUM), Teseon GmbH

Introduction

• Direct and indirect dependencies
• Representation of aggregated system views
• The necessity of the Why-Matrix
• Existing approaches and their limits
• Two basic use cases for the Why-Matrix
• Software implementation for practical use
MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

Direct and indirect dependencies

Deduction of indirect dependencies by matrix multiplication

Systematic deduction of indirect dependencies

- Six basic logics for deduction of indirect dependencies
- Result:
  - DSM-conform model
  - Information about links to other domains aggregated in the dependencies of the DSM

Representing native dependencies in several domains

- All native system dependencies represented
- Structural constellations can hardly be interpreted
- No expressiveness of elements’ positioning

Representation of aggregated system views

Designers collaboration due to responsibility for cross-linked components

Designers collaboration due to exchanged documents

Significant constellations – but dependency causes are unknown
The Why-Matrix approach applied to the deduction of DMMs

General Layout of the Why-Matrix

Designers are (indirectly) linked, because:

- a) work on component 2
- b) work on related components 3 and 4

- a) work on the same document 5
- b) provides document 3

Limits for the conventional notation of the Why-Matrix

- The existence of many indirect dependencies makes the Why-Matrix difficult to read.
- Indirect dependencies can pass by more than one additional system element.
- Six basic logics for the deduction of indirect dependencies.
- Specifications of dependencies (e.g., dependency meaning) cannot be displayed in matrix cells.

Selective Application of the Why-Matrix

- Users do not require comprehensive information from the Why-Matrix.
- Two application scenarios exist for the consideration of dependency causes:
  - Which indirect dependencies (or dependency paths) exist between two specific system components?
  - Which system components are indirectly linked by (paths containing) a specific system component?

Representation of dependency causes in list form is possible.
Use Case 1: Dependencies between specific nodes

- Both use cases have been implemented to LOOMEO
- Click on one dependency in graph representation provides the linking cause in an additional window
- Dependency causes are identified on demand (dynamic system changes can be handled)

Use Case 2: Nodes connected by the same dependency cause

- Provision of all possible dependency causes in a separated list
- Click on one dependency cause highlights the connected elements linked due to this cause
Conclusions

• Aggregated system views are required to gain system understanding
• Disadvantage of aggregated views is the absence of dependency reasons
• The (DSM-conform) Why-Matrix can provide the dependency reasons but can become rather complex to read
• The entire Why-Matrix is not needed for analyses
• Two different application scenarios allow representation in list form
• Software implementation available