COMPACT COMPARISON OF COMPETING SOFTWARE DESIGNS

Noam Tamim and Iaakov Exman
Software Engineering Dept.
Jerusalem College of Engineering
POB 3566 – Jerusalem – 91035 - Israel
noamt, iaakov@jce.ac.il

ABSTRACT

The numerical Traceability Matrix of a given software system is a compact representation of the system design in terms of software components. It is a much more powerful design tool than a checklist as it has been used so far.
Two competing designs are equivalent if and only if their matrices can be put in identical form. The most modular design is that with highest computed diagonality.
The paper discusses case studies illustrating these claims.

Keywords: software components, Traceability, T-Matrix, diagonality, modularity, competing designs.

1. INTRODUCTION

The design phase in software engineering often produces – besides the previous phases' User Requirements Document, Use-Cases, Specification documents – a large documentation set including UML class diagrams, conceptual model, and more.

Facing the need to compare two or more competing designs of the same system, one asks:

- Are those designs equivalent?
- If not – which of the designs is more modular?

Since the referred large documents are written in natural language, it is very difficult to make those decisions, and probably impossible to make the decisions quickly.

This paper proposes the Traceability Matrix as a compact design tool, allowing formal and clear-cut answers to the above issues.

2. THE TRACEABILITY MATRIX

The software component Traceability Matrix is a matrix with Software Requirements as the row labels, and Software Components (or Modules) as the column labels.

In its simplest form, the Traceability Matrix has a check-mark where there is a link between a requirement and a component. In this context, the meaning of a link is that a specific requirement is implemented by a component. For example:

	C1	**C2**
R1	✓	✓
R2		✓

In this extremely simple example, requirement R1 is implemented by both components C1 and C2, while R2 is implemented by C2 alone.

Instead of the commonly adopted check-list point of view illustrated above, we shall use the Traceability Matrix – from now on referred to as the T-Matrix – as real matrices with numerical values and exploit their mathematical properties. Realistic T-Matrices are much larger than the above example and will always be sparse matrices.

In the companion paper in these proceedings [1], it is shown that the meaning of the T-Matrix is invariant to row or column reordering.

Two T-Matrices differing either by numbers of rows, columns or non-zero elements are surely non-equivalent. The way to show the equivalence of a T-Matrix to another – with the same dimensions and same numbers of non-zero elements – is to apply reordering transformations, until they are identical.

3. DIAGONALITY AND MODULARITY

3.1. Off-Diagonality Level

Given a matrix M, the Off-Diagonality Level (ODL) for element $M_{i,j}$ is defined as $|i-j|$. That way, the ODL for all elements on the main diagonal is zero, and the farther the element is from that diagonal, the higher the level is.

The ODL for row i is therefore the sum of row elements ODL. The same idea applies to columns, and finally to matrices as well.

3.2. Modularity

It is accepted wisdom (see e.g. reference [2]), that in the most modular design, for every requirement there is one – and only one – component that implements it. In terms of modularity, it means that if a single requirement changes, only one component will have to change. The T-Matrix, of such a design has all elements in the main diagonal 1-valued, and all other elements zero-valued (omitted for clarity throughout the paper).

Thus, the most modular design is represented by a square T-Matrix. This is simply because the number of requirements (row) equals the number of components (columns). An analogous result will be stated in more generality – for block-diagonal and not strictly diagonal matrices – and formally proved elsewhere.

Given two competing designs and their corresponding non-equivalent T-Matrices, we need a simple criterion for modularity comparison. Our claim is: the most diagonal T-Matrix is the one representing the most modular design.

In practice, one takes the T-Matrices for the competing designs *A* and *B* and, by means of rows and/or columns swapping, produces matrices *A′* and *B′* such that *A′* is equivalent to *A* and *B′* is equivalent to *B*, maximizing diagonality for each matrix. Then one applies the above criterion.

3.3. Master-Control and Main modules

Often there are modules which apparently break the statements in the previous sub-sections. For instance the so-called "Master-Control" or "Main" modules – see e.g. Table 1 – seem to implement all the system requirements. But normally, such modules are merely entry points, whose only functionality is invoking other functions.

Such modules indeed do not affect modularity. They create a T-Matrix column with all 1-valued elements. Hence, changes in requirements will cause changes in other modules, not in the "Master Control".

4. CASE STUDY: PARNAS' KWIC INDEX

To demonstrate the usefulness of T-Matrices, we analyze a few case studies. The first one refers to the classical paper by Parnas [3], which compares two modularizations of the KWIC index system.

The system outputs a listing in alphabetical order of all circular shifts of all input lines. Both modularizations implement the following software requirements, extracted from Parnas' description of the system:

R1: Input = ordered set of lines
R2: Line = ordered set of words
R3: Perform circular shift on each line
R4: Sort circular shifted lines in alphabetical order
R5: Output = lists all circular shifts of all lines

The next tables show the T-Matrices for the two suggested modularizations. The components in both cases are explicit in the Parnas' paper itself.

Table 1 - Parnas' 1st Modularization

R\C	C1	C2	C3	C4	C5
R1	1				1
R2	1	1	1		1
R3		1			1
R4			1		1
R5				1	1

Components for modularization 1:
C1: Input C2: Circular Shifter
C3: Alphabetizer C4: Output
C5: Master Control

Table 2 - Parnas' 2nd Modularization

R\C	C1	C2	C3	C4	C5	C6
R1	1					1
R2		1				1
R3			1			1
R4				1		1
R5					1	1

Here, a single new "Line Storage" Component is added. The component labels are:
C1: Input C2: Line Storage
C3: Circular Shifter C4: Alphabetizer
C5: Output C6: Master Control

The designs shown above are clearly not equivalent. The number of columns and non-zero elements are different in these T-Matrices. No reordering transformation can convert the 1st T-Matrix into the second one.

As stated in sub-section 3.3 we can disregard the "Master-Control" column in both matrices. This brings the 2nd T-Matrix to a square, strictly diagonal form. Then, the calculated ODL for the 1st T-Matrix is 5, while for the 2nd T-Matrix it is zero. The 2nd T-Matrix is more diagonal than the first one. By this criterion, the second design is more modular – in agreement with Parnas' original analysis.

5. CASE STUDY: MATLAB SPLINE-DRAWING

The second case study we analyze is based on a college exercise given to students of a Computer Aided Design course. We show two designs, done by two independent students attending the course, and compare them.

The computer-graphics application, designed for MATLAB, is supposed to do the following: let the user enter some "control points" via MATLAB's GUI, then calculate and draw a spline (a series of polynomial curves) that passes through all the points and has some special properties. The calculations are based on polynomial interpolation and some spline-related equations. (De-Boor and Bezier control points are geometrical features of splines; for further details see e.g. [4]).

After the initial curve is drawn, the application must allow the user to move a few control points, and as a result, the spline has to be recalculated and redrawn.

The list of software requirements is:

R1: Initialize display frame.
R2: Accept control points from user via GUI.
R3: Calculate De-Boor control points.
R4: Calculate Bezier control points.
R5: Draw initial spline.
R6: Let user move a point.
R7: Recalculate control points.
R8: Redraw spline.

5.1. Spline Drawing Design 1

The components in this design are:

C1: ResetFrame C2: GetInputPoints
C3: MovePoint C4: RecalcPoints
C5: FindPointIndex C6: GetBezierCP
C7: PlotBezierCurves C8: GetDeBoorCP
C9: GetBezierPlotPoints C10: Main

Disregarding the Main module, and swapping some rows and columns, we get the T-Matrix in Table 3. It is pretty "messy": the links among rows and columns look almost random.

Out of 8 requirements, 4 are implemented by more than one component and out of 9 components, 4 implement more than one requirement.
The ODL for this T-Matrix is 10, due to many off-diagonal non-zero elements.

Table 3 – Spline-Drawing 1st T-Matrix

R\C	C2	C3	C5	C1	C6	C8	C4	C7	C9
R2	1								
R6		1	1						
R1				1					
R8				1				1	
R4					1				
R3						1			
R7					1	1	1		
R5								1	1

5.2. Spline Drawing Design 2

Keeping the same names for similar components, these are now:

C1: ResetFrame C2: GetInputPoints
C3: GetDeBoorCP C4: GetBezierCP
C5: GetBezierPlotPoints C6: PlotBezierCurves
C7: GetPointToMove C8: RecalcPoints
C9: Main

The T-Matrix is seen in Table 4.

Table 4 – Spline-Drawing 2nd T-Matrix

R\C	C1	C2	C3	C4	C5	C6	C7	C8
R1	1							
R2		1						
R3			1					
R4				1				
R5					1	1		
R8					1	1		
R6							1	
R7								1

By inspection, the above T-Matrix is seen to be pretty modular. Most requirements are implemented by a single component. There are two requirements – R5 and R8 – implemented by two components, C5 and C6, forming a block in the matrix, highlighted in gray, making it block-diagonal (cf. ref. [1]). The ODL of this design is 2.

6. CASE STUDY: MULTIPLE CHOICE EXAM

Here we analyze a simplified system for interactive, GUI-based, multiple choice exam. The system loads the questions and answers data from a file. The user starts the exam when he/she is ready. The exam durations is less or equal than timeout, as set by a countdown timer. These and the remaining software requirements are:

R1: Load from file – all questions and answers
R2: Single answer – is correct for each question;
R3: There is a countdown timer
R4: User decides when exam starts/finishes;
R5: User can browse questions;
R6: Single question – is shown on screen;
R7: Time/Score are displayed when exam finished.

6.1. Multiple Choice Exam Design 1

The respective components list is:
C1: Input from files
C2: Timer: set, stop, alarm
C3: Question "class": text to be displayed, accepted answer, correct answer, compare and save answer;
C4: sum-up results
C5: Browse: choose question
C6: Input answer
C7: I/O timer and grade

The design approach is to try to completely separate internal components (C1 to C4) from the GUI (components C5 to C7). The T-Matrix after reordering is seen in the next table.

Table 5 – Multiple Choice Exam 1st T-Matrix

R\C	C1	C3	C5	C6	C2	C7	C4
R1	1						
R2		1		1			
R5			1				
R6		1		1			
R3					1		
R4					1	1	
R7						1	1

This T-Matrix is almost block-diagonal with a natural interpretation. At R1 – initial input from files; Rows 2 to 4 – refer to questions; Rows 5 to 7 – refer to start/finish activities. Its calculated ODL is 6.

6.2. Multiple Choice Exam Design 2

Here the design was directed towards overall simplicity. Its components are:
C1: LoadQuestions
C2: ShowQuestion by index
C3: StartTimer
C4: StopTimer
C5: ShowTimeScore
C6: SelectAnswer

The T-Matrix is shown in Table 6. Although this T-Matrix is rectangular, 1-valued elements do accumulate around the diagonal (i=j). It even forms a diagonal block - Timer related, highlighted in gray. The calculated ODL is 8.

Table 6 – Multiple Choice Exam 2nd T-Matrix

R\C	C1	C6	C3	C4	C2	C5
R1	1					
R2		1				
R3			1	1		
R4			1	1		
R5					1	
R6		1			1	
R7						1

Thus, the first design is more modular, perhaps because the components cover all the requirements at a more uniform design level.

7. DISCUSSION

There have been design comparison proposals in the literature besides our T-Matrix (e.g. [5]). Among the most similar, Sullivan et al. [6] suggest the Design Structure Matrix (DSM) as a means of comparing designs. While it can be used to judge design quality, it seems to take into account too many parameters, most of them irrelevant to modularity. To appreciate the dimension differences, two 5x5 T-Matrices are needed for the Parnas' example, while 16x16 DSMs are needed to get the same insight.

REFERENCES

[1] I. Exman, "Software Component Completeness by Block-Diagonalized Traceability Matrices", in Proc. 23rd IEEE Convention in Israel, Herzlia, September 2004.

[2] I. Navarro, N. Leveson and K. Lundquist, "Reducing the Effects of Requirements Changes through System Design", MIT, Software Engineering Research Laboratory, Technical Report, Cambridge, MA, 2001.

[3] D. L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", Comm. ACM, **15**, 1053-1058 (1972).

[4] G. Farin, *Curves and Surfaces for CAGD*, 5th edition, Morgan-Kauffman, San Francisco, 2002.

[5] R. M. Podorozhny and L. J. Osterweil, "The Criticality of Modeling Formalisms in Software Design Method Comparison", Proc. 19th Int. Conf. on Software Engineering , Boston, MA, 1997.

[6] K.J. Sullivan et al., "The Structure and Value of Modularity in Software Design", Proc. 8th European Software Engineering Conference, pp. 99-108, Vienna, Austria, September 2001.