FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 32, No 2 (2017), 181–193
DOI: 10.22190/FUMI1702181P

ON THE CHARACTERIZATION OF TASKS MODELED BY INTERVAL DESIGN STRUCTURE MATRIX ON DOMAIN-DRIVEN DESIGN SOFTWARE DEVELOPMENT *

Ivan Petković, Petar Rajković and Aleksandar Milenković

Abstract. Development and design of new products of various kinds often contain a very complex set of relationships among many coupled tasks. Ranking, controlling and redesigning the features of these tasks can be usefully performed by a suitable model based on the design structure matrix in an iteration procedure. The proposed interval approach of design iteration controls and predicts the convergence speed of iteration work on tasks within a project. Interval method is based on Perron-Frobenius theorem and interval linear algebra where intervals and interval matrices are employed instead of real numbers and real matrices. In this way, a more relaxed quantitative estimation of tasks is achieved and the presence of undetermined quantities is allowed to a certain extent. The presented model is demonstrated in the example of simplified domain-driven design process, an approach to software development.
Keywords: Interval method; Perron-Frobenius theorem; Interval linear algebra; Software development

1. Introduction

Industrial, software, hardware and manufacturing companies of various types are faced with the growing demands to improve, upgrade and modernize their productivity in the research, development and production of their products. Most frequently, design and development of new products of various kinds (e.g., industrial innovations, technical products, bioengineering, hardware, software, etc.) deal with very complex relationships among a large number of mutually connected tasks. For this reason, solving a given general problem in a large project often requests the solution step by step, that is, requires the *design iteration* (the terminology introduced by Smith and Eppinger (see [14], [15]) which involves different, mutually dependent tasks in the development of a new product. Models that are developed

Received September 24, 2016; accepted October 10, 2016
2010 *Mathematics Subject Classification.* Primary 15A90, 15A18; Secondary 26A18, 65G30
*This study was supported by the Serbian Ministry of Education and Science.

or designed by iteration enable an essential insight into the complex relationships within a design process.

An iterative approach to improve the characteristics of a product is entirely logical since, naturally, the development and design of products themselves are often of procedural nature and, therefore, they are carried out through an iterative process. Using an analytical analysis of a complex system modeled by the so-called *design structure matrix*, it is possible to get a proper insight into the individual contribution or influence of each task in each iteration step. The model can be regarded as successful if all the planned activities converge after a certain number of iterative steps. If the number of iterations is smaller, it is clear that the model is more successful. Such an approach enables a fruitful analysis of the global process through individual tasks. A further benefit is the recognition of those subsets of tasks that require the major part of the increased effort in the course of the iterative process.

Following Eppinger and Browning [1], design structure matrix (DSM) is "a straightforward and flexible modeling technique that can be used for designing, developing and managing complex system." For its clear advantages, such as compact format, intuitive representation, visual nature and flexibility, DSM has now become a powerful tool applicable in many disciplines: engineering management, financial systems, computer science, public policy, natural science, health care management, etc. It is of interest to note that in recent time a broad range of DSM applications address such complex issue as aircraft jet engine, Mars Pathfinder spacecraft, helicopters, automobile control system, various software, digital printing system and others, see [1].

In this paper we present an approach based on interval design structure matrix for ranking tasks of a considered design mode and for controlling the speed of improvement of tasks in design iteration. Such strategy provides a more relaxed quantitative estimation of tasks. The interval approach has not been considered in the literature and requires further investigation in different scientific disciplines, including interval mathematics. A part of research on the mentioned theme was presented in the dissertation [9]. Some basic concepts and properties of matrix algebra are given in Section 2. A model of design iteration is described in Section 3 applied to a simplified domain-driven design (DDD). Section 4 presents an interval approach for ranking tasks of a design mode using interval eigenvalues and eigenvectors. Finally, a characterization of the design iteration by interval work vectors is described in Section 5. The proposed methods are illustrated by two examples taking DDD process as a model.

2. Some basic concepts of matrix algebra

In this section we give some definitions and properties from the (interval) matrix theory necessary for our analysis of design modes.

Definition 1. Given a real or complex $n \times n$ matrix A, a non-zero vector $\boldsymbol{x}$ is

called an *eigenvector* of A if there is a scalar λ such that

$$A\boldsymbol{x} = \lambda\boldsymbol{x}.$$

holds. The scalar λ is called an *eigenvalue* of A corresponding to the eigenvector $\boldsymbol{x}$. The eigenvalue largest in magnitude is the *spectral radius* of the matrix, denoted by $\rho(A)$.

Definition 2. An $n \times n$ nonnegative matrix $A = (a_{ij})$ $(n \geqslant 2)$ is said to be *irreducible* (coupled) if and only if for any pair of indices $i, j \in \{1, \ldots, n\}$ there is a sequence of nonzero elements of the matrix A of the form

$$a_{ii_1}, a_{i_1 i_2}, \ldots, a_{i_m j} \quad (m \geqslant n).$$

Consequently, irreducible matrix cannot be transformed into block upper-triangular form by permutation.

Definition 3. Let A be a square $n \times n$ matrix. Then, A is *convergent* (*to zero*) if the sequence of matrices $A, A^2, A^3, \ldots$ converges to the null matrix O, and is *divergent* otherwise.

Theorem 1. *A square matrix A of order $n \times n$ is convergent if and only if $\rho(A) < 1$.*

The following theorem has an important role in the analysis of the convergence of design iterations.

Theorem 2. (Perron-Frobenius [8]) *Let A be a nonnegative and irreducible square matrix of order $n \times n$. Then*

1) *A has a positive real eigenvalue equal to its spectral radius.*
2) *To $\rho(A)$ there corresponds a positive eigenvector.*
3) *$\rho(A)$ increases when any entry of A increases.*
4) *$\rho(A)$ is a simple eigenvalue of A.*

We will use the following denotation. Real numbers are denoted by small letters $a, b, c, \ldots$; closed real intervals $[\underline{a}, \overline{a}]$ $(\underline{a} \leqslant \overline{a})$ by capital letters $A, B, \ldots$; the set of all such intervals by $I(\mathbb{R})$. The interval $[\underline{a}, \overline{a}]$ is nonnegative if $\underline{a} \geqslant 0$. Matrices whose elements are from $I(\mathbb{R})$ will be denoted with capitals in bold $\boldsymbol{A}, \boldsymbol{B}, \ldots$ or by $(A_{ij}), (B_{ij}), \ldots$; the set of these matrices is $M_{nm}(I(\mathbb{R}))$. Real and interval vectors are denoted uniquely with small bold letters $\boldsymbol{x}, \boldsymbol{y}, \ldots$ without being confused. Finally, point matrices belonging to $M_{nm}(\mathbb{R})$, will be denoted by $\underline{\boldsymbol{A}}, \underline{\boldsymbol{B}}, \ldots$. More about the properties of interval matrices and operations in $M_{nm}(I(\mathbb{R}))$ can be found in [5] and [7].

For the purpose of further analysis we introduce the notion of absolute value of an interval matrix $\boldsymbol{A}$. For a real interval $A = [\underline{a}, \overline{a}]$, we define its absolute value as $|A| := \max\{|\overline{a}|, |\underline{a}|\}$. Nonnegative real matrix $|\boldsymbol{A}| := (|A_{ij}|)$ is called the *absolute value of interval matrix* $\boldsymbol{A} = (A_{ij})$.

Definition 4. Interval matrix $\boldsymbol{A}$ is called irreducible if the real matrix $|\boldsymbol{A}|$ is irreducible.

Definition 5. Interval matrix $\boldsymbol{A} = (A_{ij})$ is called nonnegative if all elements A_{ij} are nonnegative intervals.

Remark 1. It is easy to prove that Theorems 1 and 2 hold for an interval matrix $\boldsymbol{A}$ taking $|\boldsymbol{A}|$ instead of A.

In this paper we will have to estimate the convergence speed of the geometrical (Neumann) series $\boldsymbol{I}+\boldsymbol{A}+\boldsymbol{A}^2+\cdots+$. For this purpose, we use the notion of average convergence speed introduced by the following definition.

Definition 6. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two $n \times n$ interval matrices. If $\|\boldsymbol{A}^k\| < 1$ for some integer k, then

$$R(\boldsymbol{A}^k) := -\ln\big[(\|\boldsymbol{A}^k\|)^{1/k}\big] = \frac{-\ln\|\boldsymbol{A}^k\|}{k}$$

is called the *average convergence speed* for k iterations of the matrix $\boldsymbol{A}$. If $R(\boldsymbol{A}^k) < R(\boldsymbol{B}^k)$, it is said that $\boldsymbol{B}$ is *iteratively faster* than $\boldsymbol{A}$ for k iterations.

Using the result from Varga [17, p. 73] we can directly state the following assertion.

Theorem 3. *Let $\boldsymbol{A}$ be a convergent $n \times n$ interval matrix. Then the average convergence speed $R(\boldsymbol{A}^k)$ for k iterations satisfies the relation*

$$\lim_{k\to\infty} R(\boldsymbol{A}^k) = -\ln\,\rho(|\boldsymbol{A}|) =: R_\infty(\boldsymbol{A}).$$

The value $R_\infty(\boldsymbol{A})$ is called the *asymptotic convergence speed*. Obviously, the convergence is slower if the spectral radius of the matrix $|\boldsymbol{A}|$ is closer to 1.

The following example illustrates the introduced concept of convergence speed.

Example 1. For $p \in [0,1]$ let us define the real matrix A as follows

$$A = \frac{1}{p+1}\begin{bmatrix} 1 & p \\ 0 & 1 \end{bmatrix}.$$

Its spectral radius is $\rho(A) = 1/(p+1)$. Therefore, the convergence asymptotic speed is $R_\infty = -\ln(\rho(A)) = p+1$. By mathematical induction we find

$$A^k = \frac{1}{(p+1)^k}\begin{bmatrix} 1 & kp \\ 0 & 1 \end{bmatrix} \quad \text{with Euclidean norm} \quad \|A^k\|_2 = \frac{\sqrt{2+(kp)^2}}{(p+1)^k}.$$

Hence, in regard to Definition 6 and Theorem 3, $R(A^k) = \ln(p+1) - \dfrac{\ln(2+(kp)^2)}{2k} \to R_\infty = \ln(p+1)$ when $k \to \infty$ and $R(A^k) < R_\infty$ for every finite k. We observe that as the value of p is larger, the average convergence speed is faster.

In Section 5 we are interested in the convergence of the geometrical series $\boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \cdots +$ and, for this reason, we use the following assertion.

Theorem 4. (Mayer [6]) *Let* $\boldsymbol{A} \in M_{nn}(I(\mathbb{R}))$. *The Neumann (geometrical) series* $\sum_{k=0}^{\infty} \boldsymbol{A}^k$ *is convergent if* $\rho(|\boldsymbol{A}|) < 1$.

3. Model of design iteration

The solutions of the model of the design iteration can deliver very useful outcomes: for instance, they can provide some additional information concerning the considered model, involve reconstruction of the process, enable new engineering tools, redefine the problems, bound or extend the goals of design, etc. The example that will illustrate this approach is concerned with domain-driven design process, an approach to software development.

The method that uses square matrices of the design structure consists of separating individual tasks within the design project, or separated blocks of tasks, see Fig.1. Its main aim is to identify the key structure of the project by analyzing the relationships among these tasks (or blocks of tasks). More details can be found in [2] and [16]. In Fig.1 the entries "x" in each column j of a square scheme indicate which other tasks get input information from the task j. As an illustration of a relatively large model, we give a design structure matrix in Fig.1. See [2] for many examples of design structure matrices applied to various branches.

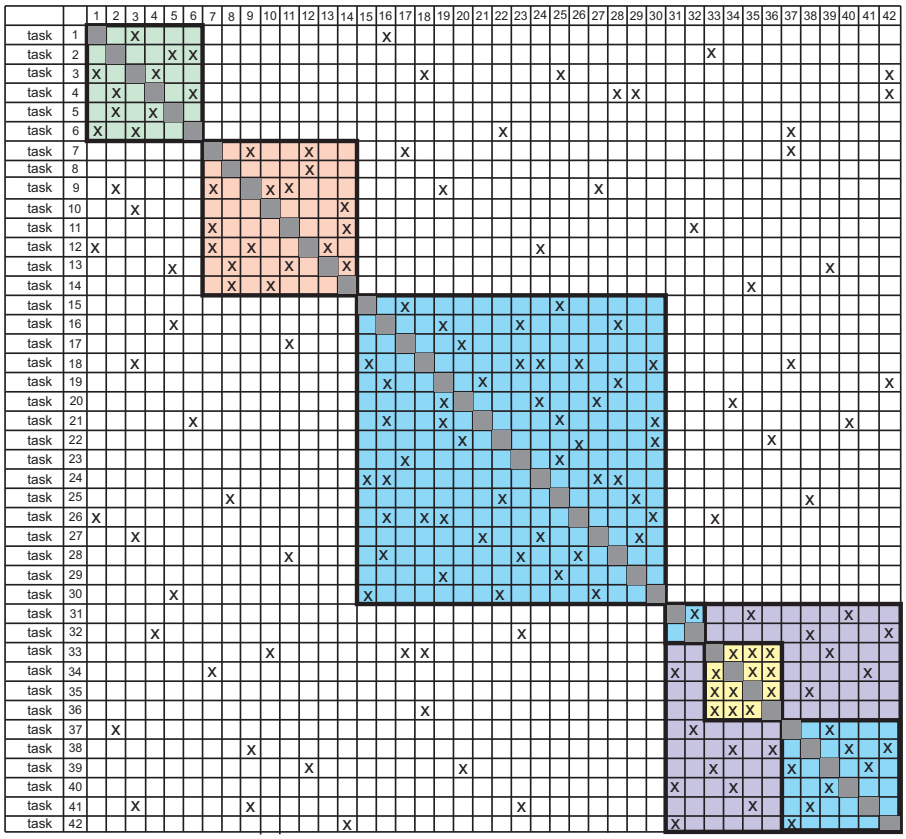

Figure 1: Design structure matrix of a large model

Specific design tasks are arranged into a square matrix A with non-negative entries, where each row and the corresponding column are associated to one of the tasks,

$$A = (a_{ij}) = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}.$$

The matrix A is called the *design structure matrix*, or shorter DSM. The coefficients of the design structure matrix A represent the intensity of dependence between tasks, which influences the transfer of work (or rework) on the tasks during the iterative process. Each entry a_{ij} of A indicates that one unit of work on design task j creates a_{ij} $(\in (0,1))$ units of rework on design task i. For example, if the strength of this dependence is 10%, then $a_{ij} = 0.1$. Diagonal entries of DSM are equal to 0 since the task does not depend on its own completion.

As already said in Introduction, a greater degree of freedom in the quantitative estimation of dependence of tasks can be accomplished by using intervals instead of real numbers in the model presented with interval design structure matrix $\boldsymbol{A} = (A_{ij})$, where we deal with real intervals A_{ij} instead of real numbers. For example, if some task participates in some design process with approximately 10%, it is more constructive to deal with the interval (0.09, 0.11) (which presents the estimate of at least 9% and of most 11%) than the fixed number 0.1 (10%). According to Theorem 3 it follows that a process of production should be designed in such way that the spectral radius $\rho(|\boldsymbol{A}|)$ is as small as possible (and ultimately less than 1), where $\boldsymbol{A}$ is interval DSM that models the process.

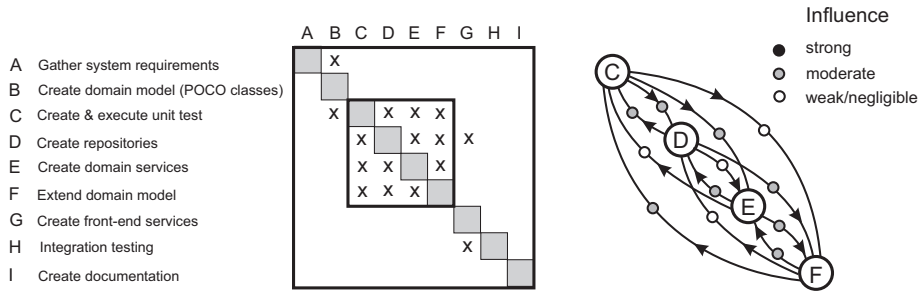

Figure 2: Example of simplified domain-driven design process

For demonstration, in this paper we consider the design structure matrix of the simplified domain driven design process, see Fig. 2. Domain-driven design (DDD), the term coined by E. Evans in the book [3], is an approach to software development where core of the software is a domain model. Domain model contains POCO classes, unaware of the rest of the infrastructure. More details on DDD can be found in [3]. In this process task T_3 needs input from tasks T_2, T_4, T_5 and T_6, task T_1 needs input only from task T_2, while task T_2 does not need any input to start. The sub-matrix 4×4 from Fig. 2 (left) presents an extracted group of tasks, the so called *design mode*; it can be also displayed by digraphs (Fig.2, right). These tasks are mutually related and complex enough so that it takes an iterative process to finish them. DSM can be used to identify the priority of the tasks, their ranking and to get insight into the most demanding tasks in the design process.

Design iteration is modeled by the use of the design matrix which obeys the following properties:

1) All tasks are carried out in every step, meaning fully parallel iterating.

2) Such a model is constructed so that design structure matrix (DSM) is non-negative and irreducible.

3) Rework is done as a function of work in the previous iterative step.

4) The coefficients A_{ij} of DSM do not vary with time.

A detailed discussion on these assumptions is given in [14].

4. Ranking the tasks

The problem of ranking the tasks can be solved in a satisfactory way using Keener's method [4]. Let $P_1, \ldots, P_n$ be participants or constitutive elements in some event (engineering design, competition, game, etc.) whose outcome depends on the influence of P-s, assuming that interactions between participants exist. For example, let $P_1, \ldots, P_n$ be players in a game. Define a_{ij} as nonnegative numbers which depend on the outcome of the game between players i and j, and let m_i is the number of games played by the player i. The elements a_{ij} form matrix $A = (a_{ij}/m_i)$, which is called a *preference matrix*. The task is to find a good method to produce a ranking list of players to satisfactory objective extent, taking into account both the outcome of interaction, and the relative strength of the opponents.

Following Keener's method [4], one can estimate the strength r_j of the player j and form the ranking vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ by a direct manner. The elements of $\boldsymbol{v}$ determine the position of a player on the ranking list. As proposed in [4], the ranking place of a participant should be proportional to its score, that is,

$$A\boldsymbol{v} = \lambda \boldsymbol{v}.$$

According to the last relation, Definition 1 and Theorem 2, an interesting and very useful conclusion follows: *the ranking vector $\boldsymbol{v}$ is a positive eigenvector of the nonnegative matrix A.*

Let us return to the problem of ranking the tasks. Having in mind that the design structure matrix $\boldsymbol{A}$ is nonnegative and irreducible, by virtue of the Perron-Frobenius theorem (Theorem 2 with $|\boldsymbol{A}|$ instead of A), the matrix $|\boldsymbol{A}|$ has one positive eigenvalue largest in magnitude (and equal to the spectral radius $\rho(|\boldsymbol{A}|)$ and the corresponding positive eigenvector. Just this eigenvector, denoted by $\boldsymbol{v}$, gives the rank of the tasks.

From the previous discussion we conclude the following:

- The largest eigenvalue, say λ_m, has the dominant influence to the convergence speed of design iteration.
- The i-th entry of the corresponding eigenvector $\boldsymbol{v}_m$, related to the eigenvalue λ_m $(= \rho(|\boldsymbol{A}|))$, characterizes the relative contribution of the design task i to the design mode.
- The greater the entry of the positive eigenvector $\boldsymbol{v}_m$, the more important the influence of that entry (task) to the mode.

In other words, *by ranking entries of the corresponding eigenvector, we rank the contributions of the design tasks.* The following example illustrates these conclusions.

Example 2. Let interval design structure matrix be given by the interval matrix

$$\boldsymbol{A} = (A_{ij}) = \begin{bmatrix} 0 & [0.23, 0.25] & [0.28, 0.3] & [0.08, 0.1] \\ [0.18, 0.2] & 0 & 0 & [0.23, 0.25] \\ 0 & [0.18, 0.2] & 0 & [0.28, 0.3] \\ [0.29, 0.31] & [0.08, 0.1] & [0.18, 0.2] & 0 \end{bmatrix}.$$

This interval matrix is, in fact, a quantitative version of the coupled blocks (tasks $T_3 - T_6$) in the domain-driven design shown in Fig. 2. For example, $A_{21} = [0.18, 0.2]$ means that if the class diagram is totally redesigned, then the developed single module will take 18 to 20% of rework.

The problem of finding interval eigenvalues and eigenvectors of interval matrices in general case is not an easy task. Fortunately, we need only dominant eigenvalue (spectral radius) and the corresponding eigenvector of nonnegative irreducible point matrices so that the problem is relaxed to a certain extent. Using the programming package INTLAB created by Rump [12, 13], combined by Rohn's checking process by random matrices $\texttt{random}\,(R) \in \boldsymbol{A}$ (10^7 random matrices were applied [11]), and theoretical results from Rohn's paper [10], we have found that the dominant eigenvalue belongs to the interval

$$\Lambda_1 = [0.5050, 0.5551].$$

Its corresponding positive interval eigenvector $\boldsymbol{v}_1$ is given by the interval vector

$$\boldsymbol{v}_1 = \begin{bmatrix} [0.2617, 0.2844] \\ [0.2114, 0.2330] \\ [0.2191, 0.2409] \\ [0.2642, 0.2853] \end{bmatrix}.$$

From the values of $\boldsymbol{v}_1$ we conclude that the fourth and first task are dominant, that is, they require more work than the second and third task.

5. Design iterations

In order to follow and control the considered model during the developing and improving process (for example, that given in Fig. 2), we introduce the notion of the work (real or interval) vector $\boldsymbol{u}_k$ as an n-dimensional vector, whose components may be real numbers or real interval. Here n is the number of coupled design tasks which have to be performed. Each component of the work vector contains the information on the quantity of work which has to be completed on every task after the k-th iteration. Obviously, the initial work vector $\boldsymbol{u}_0$ is given by $\boldsymbol{u}_0 = (1, 1, \ldots, 1)$. Simply, at the beginning of the iteration process the remaining work on all tasks in the design process is 100%.

The required work is carried out during every iterative step on all design tasks. However, due to the mutual dependence of tasks (defined by the design structure matrix $\boldsymbol{A} = (A_{ij})$), the work on some task, say i, will request the rework on all other tasks with the amount depending on the information from the considered task i. Each element A_{ij} of $\boldsymbol{A}$ denotes that one unit of work on design task j creates A_{ij} units of rework on design task i (given by an interval, or by real number in a

special case working in real arithmetic). This means that the matrix $\boldsymbol{A}$ defines the strength of the dependence between tasks. For example, $A_{ij} = [0.6, 0.7]$ denotes a strong dependence, while $A_{ij} = [0.06, 0.07]$ denotes a weak dependence.

After the first iteration the remaining work to be performed is calculated as $\boldsymbol{u}_1 = \boldsymbol{A}\boldsymbol{u}_0$, after the second iteration is $\boldsymbol{u}_2 = \boldsymbol{A} \cdot \boldsymbol{A}\boldsymbol{u}_0 = \boldsymbol{A}^2\boldsymbol{u}_0$, etc. Each iteration step leads to the change in the work vector. After the k-th iteration the remaining work is given by

$$\boldsymbol{u}_k = \boldsymbol{A}\boldsymbol{u}_{k-1} = \boldsymbol{A}^k\boldsymbol{u}_0. \tag{5.1}$$

It can be noticed that the described model is analogous to dynamical systems.

The *total work vector* $\boldsymbol{u}$, which contains the information about the total work done on all tasks, is the sum of all work vectors obtained in the course of m iterative steps:

$$\boldsymbol{u} = \sum_{k=0}^{m} \boldsymbol{u}_k = \sum_{k=0}^{m} \boldsymbol{A}^k\boldsymbol{u}_0 = \left(\sum_{k=0}^{m} \boldsymbol{A}^k\right)\boldsymbol{u}_0. \tag{5.2}$$

The obtained quantity $\boldsymbol{u}$ is the total amount of work required for finishing each task. For example, if the i-th component of $\boldsymbol{u}$ is [1.5, 1.55], this means that 50% to 55% of rework will be necessary for task i after the initial iteration.

To state the conditions under which the sum of interval matrices $\boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \cdots + \boldsymbol{A}^m$ in (2) is bounded and converges to a fixed interval matrix as the number of iterations m grows, we use Theorem 4. According to this theorem, it follows that this series converges if and only if $\rho(|\boldsymbol{A}|) < 1$. In that case we have

$$\sum_{k=0}^{m} \boldsymbol{A}^k \subset \sum_{k=0}^{\infty} \boldsymbol{A}^k = (\boldsymbol{I} - \boldsymbol{A})^{-1}.$$

According to (2) and the last inclusion, the total work vector can be bounded in the following way

$$\boldsymbol{u} = \left(\sum_{k=0}^{m} \boldsymbol{A}^k\right)\boldsymbol{u}_0 \subset (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{u}_0. \tag{5.3}$$

Example 3. The characterization of tasks which take part in the design mode through the eigenvalues and eigenvectors of interval matrix $\boldsymbol{A}$ will be illustrated using data obtained in Example 2. Design structure matrix $\boldsymbol{A}$ is nonnegative and irreducible by assumption. According to the Perron-Frobenius theorem (Theorem 2 with $|\boldsymbol{A}|$ instead of A), there is exactly one positive eigenvalue largest in magnitude (and, therefore, equal to the spectral radius $\rho(|\boldsymbol{A}|)$) and the corresponding eigenvector with positive elements. Assuming that this eigenvalue is less than 1, then it dominantly influences the convergence speed of the sequence $\{\boldsymbol{A}^k\}$, and, consequently, the sequence of work vectors $\{\boldsymbol{u}_k\}$ (according to (1)). From Theorem 3 it follows that as its value is higher (but not exceeding 1), the convergence is slower so that such a design mode requires special attention during the reconstruction of model.

We shall show that the higher the entries of the corresponding positive eigenvector, the greater the influence of that entry (and the corresponding task) to that mode. We found in Example 2, were a domain-driven design is considered, that the dominant eigenvalue

belongs to the interval $\Lambda_1 = [0.5050, 0.5551]$. This interval has a dominant influence to the convergence speed of the sum of interval matrices $\boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \boldsymbol{A}^3 + \cdots$, defining in this way the slowest design mode. This means that most of the work in the iteration process is described by that primary design mode. Since $|\Lambda_1| = 0.5551... < 1$, the geometrical series $\boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \boldsymbol{A}^3 + \cdots$ is convergent (due to Theorem 4).

This is also confirmed by the values of the interval work vectors, evaluated for the first four iterations,

$$\boldsymbol{u}_1 = \begin{bmatrix} [0.59, 0.65] \\ [0.41, 0.45] \\ [0.46, 0.50] \\ [0.55, 0.61] \end{bmatrix}, \qquad \boldsymbol{u}_2 = \begin{bmatrix} [0.267, 0.326] \\ [0.233, 0.283] \\ [0.228, 0.273] \\ [0.286, 0.347] \end{bmatrix},$$

$$\boldsymbol{u}_3 = \begin{bmatrix} [0.1402, 0.1872] \\ [0.1140, 0.1514] \\ [0.1221, 0.1605] \\ [0.1370, 0.1832] \end{bmatrix}, \quad \boldsymbol{u}_4 = \begin{bmatrix} [0.07139, 0.10428] \\ [0.05677, 0.08322] \\ [0.05890, 0.08521] \\ [0.07178, 0.10525] \end{bmatrix}.$$

Conclusion: *the work done in the second and third task is less than the work in the first and fourth task.*

The formula (3) is applied for the estimation of the vector of total work. The inversion of the interval matrix $\boldsymbol{I} - \boldsymbol{A}$ is performed by the use of programming package INTLAB [12]:

$$(\boldsymbol{I} - \boldsymbol{A})^{-1} = \begin{bmatrix} [1.1399, 1.1933] & [0.3479, 0.4194] & [0.3667, 0.4286] & [0.2720, 0.3528] \\ [0.2915, 0.3501] & [1.1154, 1.1618] & [0.1374, 0.1811] & [0.3168, 0.3798] \\ [0.1578, 0.2038] & [0.2656, 0.3258] & [1.1121, 1.1506] & [0.3836, 0.4471] \\ [0.3807, 0.4457] & [0.2361, 0.3114] & [0.3160, 0.3811] & [1.1711, 1.2368] \end{bmatrix}.$$

According to this we have estimated the interval total work vector

$$\boldsymbol{u} \subset (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{u}_0 = \begin{bmatrix} [2.1265, 2.3941] \\ [1.8611, 2.0728] \\ [1.9191, 2.1273] \\ [2.1039, 2.3751] \end{bmatrix}.$$

The elements of $\boldsymbol{u}$ reveal that *the first and fourth task demand more work*, as was concluded from the insight to the values of each work vector $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ and the interval eigenvalue $\boldsymbol{v}_1$ (see Example 2).

6. Conclusion

In this paper we are concerned with a method for ranking tasks modeled on design structure interval matrix and a precise interpretation of the model matrix structure of the design iteration by the use of the eigenvalues and eigenvectors of the corresponding matrix. Compared to the analysis presented in [15], we give a more general approach using interval linear algebra where we deal with real intervals and interval matrices instead of real numbers and real matrices. In this way a

more relaxed quantitative estimation of tasks is achieved and the presence of the "undetermined" quantities is allowed to a certain extent (defined by the length of interval). The use of intervals is natural for practical reasons since many quantities are available only approximately.

Acknowledgment. The authors are grateful to Professor J. Rohn (Institute of Computer Science, Czech Academy of Sciences) for his very fruitful discussion on interval eigenvalues, and to Professor S. Eppinger from Massachusetts Institute of Technology (Cambridge, Massachusetts) and Professor G. Mayer from the University of Rostock for their earlier helpful discussions concerning the topic.

REFERENCES

1. S. D. Eppinger and T. R. Browning, *Design Structure Matrix Methods and Applications*. MIT Press, Cambridge, MA, 2012.
2. S. D. Eppinger, D. E. Witney, R. P. Smith and D. A. Gebala, *A model-based method for organizing tasks in product development*. Res. Engineering Design **6** (1994), 1–13.
3. E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004.
4. J. P. Keener, *The Perron-Frobenius theorem and the ranking of football teams*. SIAM Review **35** (1993), 80–93.
5. G. Mayer, *On the convergence of power of interval matrices*. Linear Algebra with Applications **58** (1984), 201–216.
6. G. Mayer, *On the convergence of the Neumann series in interval analysis*. Linear Algebra with Applications **65** (1985), 63–70.
7. G. Mayer, *Interval Analysis and Automatic Result Verification*. Walter de Gruyter (to appear).
8. C. Meyer, *Matrix Analysis and Applied Linear Algebra*. SIAM, 2000.
9. I. Petković, *Analysis of Processor and Computational Iterations by Applying Modern Computer Arithmetics*. Ph. D. Thesis, Faculty of Electronic Engineering, University of Niš, Niš, 2011.
10. J. Rohn, *Perron vectors of an irreducible nonnegative interval matrix*. Linear and Multilinear Algebra **54** (2006), 399–404.
11. J. Rohn, *Eigenvalues and eigenvectors of interval matrices* (private correspondence).
12. S. M. Rump, *INTLAB - INTerval LABoratory*. In: Proceedings of a Conference on Developments in Reliable Computing (T. Csendes, ed.), Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104.
13. S. M. Rump, *Verification methods: Rigorous results using floating-point arithmetic*. Acta Numerica **19** (2010), 287–449.
14. R. P. Smith, *Development and verification of engineering design iteration models*. Ph. D. Thesis, MIT Sloan School of Management, Cambridge, MA, 1992.
15. R. P. Smith and S. D. Eppinger, *Identifying controlling features of engineering design iteration*. Management Science **43** (1997), 276–293.

16. D. V. Steward, *The design structure system: a model for managing the design of complex systems*. IEEE Trans. Engineering Management **EM-28** (1981), 71–74.
17. R. S. Varga, *Matrix Iterative Analysis*. Springer-Verlag, Berlin-Heidelberg, 2009.

Ivan Petković
Faculty of Electronic Engineering
University of Niš
A. Medvedeva 14
18000 Niš, Serbia
ivan.petkovic@elfak.ni.ac.rs

Petar Rajković
Faculty of Electronic Engineering
University of Niš
A. Medvedeva 14
18000 Niš, Serbia
petar.rajkovic@elfak.ni.ac.rs

Aleksandar Milenković
Faculty of Electronic Engineering
University of Niš
A. Medvedeva 14
18000 Niš, Serbia
aleksandar.milenkovic@elfak.ni.ac.rs