

A Social Network Analysis-Based Approach for Modelling and Analysing Dependencies among Requirements in New Software Development Projects

Messa Alhammadi

Department of Industrial Engineering and Engineering Management, College of Engineering, University of Sharjah, Sharjah, UAE
U18105008@sharjah.ac.ae

Udechukwu Ojiako

Faculty of Engineering, University of Strathclyde, United Kingdom
udi.ojiako@strath.ac.uk

Hamdi Bashir

Department of Industrial Engineering and Engineering Management, College of Engineering, University of Sharjah, Sharjah, UAE
hbashir@sharjah.ac.ae,

Salah Haridy

Department of Industrial Engineering and Engineering Management, College of Engineering, University of Sharjah, Sharjah, UAE
sharidy@sharjah.ac.ae,

Mohammad Shamsuzzaman

Department of Industrial Engineering and Engineering Management, College of Engineering, University of Sharjah, Sharjah, UAE
mshamsuzzaman@sharjah.ac.ae

A Social Network Analysis-Based Approach for Modelling and Analysing Dependencies among Requirements in New Software Development Projects

Abstract

Previous studies proposed several methods to model and analyse the dependencies within software requirements (SRs) in development projects. For practical purposes, teams developing software require simple tools that assist in mapping and assessing the relationships between SRs during the initial stages of development. Considering the limitations of previous works, this study suggests adopting a social network analysis-based approach. This approach enables development teams to analyse and visualise the relationships between SRs, offering a holistic view of SR interactions and aiding in identifying the most critical SRs along with their paths of change propagation. As a result, this approach facilitates effective and efficient change management, reducing cost risks and lead-time overruns in projects. The practicality of this approach is demonstrated through its application in modelling and analysing 28 SRs in a real-world software development project.

Keywords: Requirements; change; dependency; software project management; modeling; social network analysis

1 Introduction

Change in software requirements (SRs), also referred to as “requirement volatility” in the literature (Thakurta et al., 2015), is an unavoidable aspect of software development projects (Otávio and Lisandra, 2023). Yin et al. (2018) pointed out that the software development process is continually evolving, and SRs are expected throughout the different stages of a project. According to Bhatti et

al. (2010), more than 50% of the SRs are changed during the lifecycle of a software development project.

Managing changes is a significant challenge for the development team and significantly influences a project's outcome, potentially resulting in business losses due to increased time demands and additional rework. Iriate and Bayona (2020) conducted a comprehensive literature review, identifying 263 factors for success, with change management emerging as the second most frequently mentioned factor. Ingram's (2011) study revealed that changes can account for 40% to 90% of the total cost of industrial projects.

Studies on managing changes have predominantly adopted different approaches. Much research on managing changes emphasises project management strategies to address the varying degrees of SRs encountered during development. Recommendations along these lines encompass the implementation of particular frameworks (such as establishing change control boards as per Jones, 2007), determining the project's execution strategy early on (for instance, choosing a suitable process model as Thakurta and Ahlemann (2011) discussed, and employing specific methodologies throughout the project's lifecycle (like the use of Joint Application Design), configuration management as Hashmi et al. (2010) and Jones (2007) suggested, establishing baseline requirements as per Wiegers (1999), and thorough change management planning (Young, 2001). Thakurta (2015) note that the applicability of these recommended management practices varies significantly across different project environments and continents, indicating that mere adherence to "best practices" may not always be effective. Nonetheless, one principal challenge in managing changes remains the complex dependencies among SRs in such projects. Carlshamre et al.'s (2001) survey reported that 80% of all SRs are interconnected. Consequently, changing one SR can trigger a cascade of changes in others, following a pattern of interconnected cause and

effect, as Dahlstedt and Persson (2005) described. Even changes initially perceived as simple can often lead to unforeseen consequences, triggering a chain reaction of modifications and causing uncertainty in project development time and cost.

Therefore, there is a need to manage dependencies among SRs, which involves identifying and keeping track of how these SRs interact and influence each other (Dahlstedt and Persson, 2005). For this purpose, several methods for modelling and analysing the dependencies among SRs have been suggested in existing research. However, these methods have various limitations. This study proposes a social network analysis (SNA)-based approach to overcome these. In addition to offering a visual aid for development teams to perceive the relationships between SRs, this approach encompasses a categorisation system that enables the recognition of the most critical SRs based on their interdependencies. This classification aids in assigning importance, overseeing, and managing the SRs that pose the most significant potential for project delays and budget overruns.

2 Related research

As per a systematic review presented by Jayatilleke and Lai (2018), the management of SRs involves three key aspects: first, recognising the need for change; second, analysis of change impact; and third, estimating the resources or costs required for the change.

This article primarily addresses impact analysis, one of the two aspects of the change impact analysis. This aspect aims to predict the impact of changes in SRs on one another. The second aspect, sensitivity analysis, is used to determine which SRs are most affected by various influencing factors and identify the design areas that are especially sensitive to SR modifications in SRs (Strens and Sugden, 1996). A crucial step in impact analysis is to model and analyse the dependencies among SRs.

2.1 Dependencies among SRs

The literature identifies various types of dependencies among SRs, broadly categorised into two main groups (Dahlstedt *et al.*, 2005; Li *et al.*, 2010). The first category divides dependencies into functional, time-related, and value-related subgroups. The functional group includes combination (SRs that should be implemented simultaneously), implication (SRs that depend on the completion of others), and exclusion (SRs that conflict and cannot be implemented together). The time-related group encompasses SRs that must align with the project schedule. The value-related group is bifurcated into revenue-based (SRs affecting income) and cost-based (SRs affecting costs).

The second main dependency category segregates them into three types: structural interdependencies, cost/value interdependencies, and constraint interdependencies. Structural interdependencies are further subdivided into “refined to,” “change to,” “similar to,” and “requires.” Cost/value interdependencies include dependencies that either “increase or decrease” the cost of or “increase or decrease” the value of an SR. Last, constraint interdependencies are made up of “requires” and “conflicts with,” with “requires” being a common element in structural and constraint interdependencies.

2.2 Modelling and analysing dependencies

Various approaches and perspectives have been taken in the literature to model and analyse the dependencies among SRs. For instance, Liu and Smolka (1998) introduced Directed Graphs (DGs) with hyperedges to represent causal connections among objects. This concept paved the way for visual representations in understanding software system dependencies. Following this, Sangal *et al.* (2005) applied the Design Structure Matrix (DSM), also called the dependency structure matrix, which offered insights into interdependencies within software architecture based on a system's source code. Building on the use of DGs, Widiastuti and Siahaan (2008) proposed a model

visualising the dependencies among SRs, focusing on the sequence of modifications in requirements. This model aimed to assist stakeholders in tracking the flow and impact of changes in SRs. Similarly, Fu et al. (2012) introduced a methodology employing DSM for analysing interconnections among software architecture components, highlighting the potential for change propagation.

The focus then shifted towards the effects of changes in global software development contexts, as proposed by Ali and Lai (2016). Subsequently, Jayatilleke et al. (2017) suggested an approach using DSM to scrutinise SRs, considering the complexity of change, while Samosir and Siahaan (2018) proposed a technique for creating a requirement DG based on relationships between SRs and classes in system design.

Yin et al. (2018) introduced a method for examining the propagation of changes in SRs at the source code level, using a class-class matrix to illustrate associations among classes. This approach facilitated the development of a change propagation pathway and the analysis of change risk using a mathematical model. Following this, Priyadi et al. (2019) modelled the requirement DG for SR specification documents by extracting relations between SRs through various text processing tools and algorithms.

Addressing limitations in prior research, including works by Arora (2015), Conejero et al. (2012), Goknil et al. (2008 and 2014), Hassine et al. (2005), and Nejati et al. (2016), Arvanitou et al. (2022) proposed a metric known as the Requirements Ripple Effect Metric. Considering the conceptual overlap among Software Requirements (SRs), code segments, and inherent dependencies within the source code, this metric gauges the likelihood of occurrence, reflecting the latest advancements in the field.

In addition to the studies mentioned above, there is a stream of research focused on the relationship between class dependencies and forecasting which classes are likely to undergo changes. This research stream includes studies by Bura and Choudhary (2020), Godara and Singh (2017), Godara et al. (2018), Malhotra and Jangra (2017), Kumar and Rishi (2016), Lu et al. (2012), and Remington and Soderholm (2010). The relationship between change-prone classes and SRs is both direct and significant. When one or more SRs change, the software must be updated to align with these new requirements, often necessitating modifications to the classes that implement these changing SRs. The issue of change-prone classes is particularly relevant in the context of updating or improving existing software applications. Therefore, the research presented in these studies falls outside the scope of this current research.

3 Research gaps and study justification

Most pertinent research regarding modelling dependencies among SRs has employed either DG or DSM. The primary drawback of utilising either the binary or the non-binary form of DM is that they do not uncover the cumulative or multilevel dependencies. To address this issue, one can turn to a DG, wherein every SR is represented as a node in a network, and arrows are employed to denote dependencies. However, the classification of SRs is not feasible with DG due to its lack of measures for identifying and analysing the position of each SR within the dependency network. Categorizing SRs is a crucial stage in overseeing the development process, as it enables the identification of the most critical SRs that demand focused attention. Arvanitou et al.'s (2022) study is perhaps one of the few that has proposed a metric for quantifying the dependencies among SRs. Nonetheless, this metric's applicability relies on information unavailable during the initial phases of software development, as it is designed for use in the later stages of software maintenance.

Considering the limitations of previous studies, a Social Network Analysis (SNA)-based approach is proposed for modelling and analysing the dependencies among SRs. SNA originated in the 1930s within sociology, primarily to study the relationships between social entities known as actors (Moreno, 1960). Over the past thirty years, however, SNA has increasingly been utilised in different fields beyond its initial scope. In project management, for instance, its use has expanded to model interactions between non-human elements (e.g., Al Zaabi and Bashir, 2020; Bashir and Ojiako, 2020; Bashir et al., 2022; Bashir et al., 2023a, 2023b; Herrera et al., 2020; Mok et al., 2017; Pryke et al., 2018). A key benefit of SNA is its ability to visually depict the connection between actors (entities such as elements, factors, people, organisations, etc.) by constructing a network of nodes linked by arcs. Beyond mere visualisation, SNA also involves the analysis of a network's structure using various node-level and network-level metrics.

4 Implementation of the approach

Three primary steps are involved in the execution of the suggested SNA-based approach: (i) determining the relationships among the SRs, (ii) conducting a quantitative analysis, and (iii) creating a visual representation of the relationships among the SRs.

It was employed to model and analyse dependencies among 28 SRs within a real-world project involving developing a company's human resource (HR) management software system to showcase and affirm the practicality of this approach. This system encompasses various functions within a company's HR department, such as benefits management, payroll, hiring, training, and performance evaluation. The following list enumerates the specific SRs involved.

- | | | | |
|---|---|----|---|
| 1 | User authentication is through username and password. | 15 | Managers, in their role, can modify their employees' information. |
| 2 | An administrator can generate user accounts | 16 | Managers can also conduct searches for employees under their supervision. |
| 3 | Users can log out of the HRMS system. | 17 | HR and administrative roles can search for information on all employees. |

- | | | | |
|----|--|----|--|
| 4 | An error message will appear if a user is not found in the database or has not been authorised by the administrator. | 18 | The search function operates based on specific keywords, revealing employees' characteristics, traits, skills, and other attributes. |
| 5 | Users have the option to reset their passwords using the phone number stored in their records. | 19 | The report button refines the results of the search function. |
| 6 | Interface and data access are contingent on the user's assigned role. | 20 | The administrator has the authority to update the role of a specific user. |
| 7 | The administrator is responsible for creating roles (HR, Manager, Employee). | 21 | HR users can add a new employee to the database. |
| 8 | The administrator assigns permissions to each role. | 22 | The administrator can modify user details. |
| 9 | Roles are assigned to users by the administrator. | 23 | The administrator can list all users. |
| 10 | The user's role status can be verified in the database. | 24 | Additionally, the administrator can conduct searches for users. |
| 11 | Users with specific roles can view the contents of the database. | 25 | The administrator has the authority to adjust a user's role. |
| 12 | Users with the Employee role can view their personal information. | 26 | Listing all roles is within the administrator's purview. |
| 13 | Users with the Employee role can access and modify their personal information | 27 | The administrator can display all users who hold a specific role. |
| 14 | Users with a Manager role can view information about their employees. | 28 | The administrator is responsible for managing permissions for each role |

4.1 Identify relationships among the SRs

In this step, the development team responsible for creating the software establishes the relationships between each pair of SRs using a Structural Self-Interaction Matrix (SSIM). This matrix is an adaptation from interpretive structural modelling (Sage, 1979) and employs four symbols to denote the relationships between SRs i and j :

V denotes that a change in SR_i would trigger a change in SR_j (a forward link).

A denotes that a change in SR_j would trigger a change in SR_i (a reverse link).

X denotes a mutual dependency between SR_i and SR_j .

O denotes that SR_i and SR_j are unrelated.

It is important to note that these relationships align with the "structural interdependencies" category, particularly the "change to" type. This categorisation describes scenarios where alterations in one requirement directly necessitate changes in another, thereby highlighting a straightforward structural interrelation between the two requirements (Noviyanto et al., 2023).

Subsequently, the SSIM is converted into an $m \times m$ adjacency matrix, where m is the count of SRs. This conversion is achieved by replacing V , A , X , and O with ones and zeros using the following rules:

- "If the (i, j) cell in the SSIM is V , then the (i, j) cell in the initial reachability matrix becomes 1, and the (j, i) entry becomes 0.
- If the (i, j) cell in the SSIM is A , then the (i, j) cell in the initial reachability matrix becomes 0, and the (j, i) entry becomes 1.
- If the (i, j) cell in the SSIM is X , then the (i, j) cell in the initial reachability matrix becomes 1, and the (j, i) entry becomes 1.
- If the (i, j) cell in the SSIM is O , then the (i, j) cell in the initial reachability matrix becomes 0, and the (j, i) entry becomes 0." (Jawad & Bashir, 2015)

4.2 Quantitative analysis

In this step, the relationships among SRs are analysed using four metrics, namely density, in-degree centrality, out-degree centrality, and betweenness centrality.

Network density gauges the relative number of ties among nodes in a network. Network density is the ratio of existing direct links among nodes to the maximum possible links if every node were connected to every other node (Wasserman and Faust, 1994). This metric in our study indicates project complexity: the greater the network density, the more complex the project is assumed to be. Our example study's network density was 0.098, signifying a low-complexity

project. In our demonstrative study, the computed network density was 0.098, indicating that this project was low complexity. Network density alone does not reveal which SRs are the most dependent or influential. For this, we suggest using degree-centrality metrics.

Tables 1 and 2 show the developed SSIM and initial reachability matrix for the demonstrative development project.

Table 1 Structural self-interactional matrix (SSIM)

SR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	O	V	V	V	V	V	O	O	V	V	V	V	O	O	O	O	O	O	O	V	O	V	V	V	V	V	O	O
2	A	O	V	O	V	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	V	V	V	V	V	O	O
3	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
4	A	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
5	A	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
6	O	O	O	O	O	O	V	V	V	V	V	V	V	V	V	V	V	V	O	O	V	O	O	O	O	O	O	O
7	O	O	O	O	O	A	O	V	V	V	V	O	O	O	O	O	O	O	O	V	O	O	O	O	O	O	O	O
8	O	O	O	O	O	A	A	O	O	X	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
9	A	A	O	O	O	A	A	O	O	V	V	O	O	O	O	O	O	O	O	V	O	O	O	O	O	O	O	O
10	A	A	O	O	O	A	X	X	X	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
11	A	O	O	O	O	A	A	X	A	O	O	V	V	V	V	V	V	O	O	O	O	O	O	O	O	O	O	O
12	O	O	O	O	O	O	O	O	O	O	O	O	V	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
13	O	O	O	O	O	O	O	O	O	O	O	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
14	O	O	O	O	O	O	O	O	O	O	O	O	O	O	V	O	O	O	O	O	O	O	O	O	O	O	O	O
15	O	O	O	O	O	O	O	O	O	O	O	O	O	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O
16	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
17	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
18	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
19	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
20	O	O	O	O	O	A	A	O	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
21	O	V	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
22	A	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
23	A	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
24	A	A	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
25	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
26	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
27	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
28	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

Degree-centrality metrics help identify the significance or role of actors in a network based on their direct connections (Borgatti, 2005). This centrality is the count of direct links a node has. In directed networks, it splits into in-degree and out-degree centrality: in-degree is measured by counting the incoming links to a node, while out-degree is measured by counting the outgoing

links from a node (Wasserman and Faust, 1994).

Table 2 Adjacency matrix

SR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0
2	0	0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	1	1	1	1
7	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0
10	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Based on their in-degree and out-degree centrality values, SRs can be represented in a four-quadrant diagram, a concept derived from the cross-impact matrix multiplication analysis by Duperrin and Godet (1973):

- The first quadrant includes autonomous SRs characterised by low dependence and driving power, meaning each SR has limited interactions with other SRs.
- The second quadrant contains SRs that are dependent, exhibiting low driving power but high dependence power. This implies that each SR in this quadrant relies heavily on numerous other SRs, yet few, if any, SRs depend on them.
- The third quadrant comprises linkage SRs with high driving and dependence power. In this category, each SR relies on many others, and many SRs depend on these.

- The fourth quadrant encompasses independent SRs with limited reliance on other SRs but significant influence over many other SRs. In other words, each SR in this quadrant relies on only a few or none of the other SRs, while numerous other SRs depend on it.

Categorising SRs into these defined groups assists in determining their level of importance. The initial category (“autonomous”) comprises the least critical SRs, while the final category (“independent”) comprises the most critical SRs. Additionally, centrality metrics, particularly betweenness, help to identify SRs of varying levels of criticality. In Social Network Analysis (SNA), betweenness centrality determines a node's importance in a network based on the shortest paths. It is quantified by the degree to which a node lies along the geodesic paths between all other node pairs. Thus, the more frequently a node appears on these paths, the higher its centrality. In the context of this study, an SR with a high value of betweenness centrality in any of the aforementioned categories can be considered critical, as it lies between two non-adjacent nodes in the network, meaning it can appear in multiple propagation paths. Mathematically, betweenness centrality is defined as:

$$BC_i = \sum_{k < l} \frac{\alpha_{kl}}{\beta_{kl}}$$

Where α_{kl} = the count of shortest routes that link k with l passing through SR_i and β_{kl} = the overall count of shortest routes connecting k with l .

In the illustrative development project, Table 3 presents the calculated values for degree centrality, out-degree centrality, and betweenness centrality. Based on the first two metrics, the out-in-degree centrality diagram presented in Figure 1 was created.

4.3 Visualisation of relationships

The last step of the proposed approach entails visualisation through constructing a network with nodes and directed arcs representing the relationships among SRs. This visualisation can help

identify potential paths through which changes may propagate following an initial change in SR_i and detect relationship patterns among SRs, which might not be obvious to a development team (Liu et al., 2014).

Table 3 The computed values of metrics used for analysing relationships among SRs

SR _i	In-degree Centrality	Out-degree Centrality	Betweenness Centrality	SR _i	In-degree Centrality	Out-degree Centrality	Betweenness Centrality
1	0	13	0.000	15	3	0	0.000
2	2	9	0.034	16	3	2	0.012
3	2	0	0.000	17	2	2	0.011
4	2	0	0.000	18	2	1	0.000
5	2	0	0.000	19	4	0	0.000
6	1	16	0.010	20	4	0	0.000
7	2	9	0.035	21	0	1	0.000
8	4	3	0.021	22	2	0	0.000
9	4	5	0.032	23	2	0	0.000
10	6	2	0.034	24	2	0	0.000
11	5	7	0.076	25	5	0	0.000
12	2	1	0.000	26	2	0	0.000
13	3	0	0.000	27	3	0	0.000
14	2	3	0.004	28	3	0	0.000

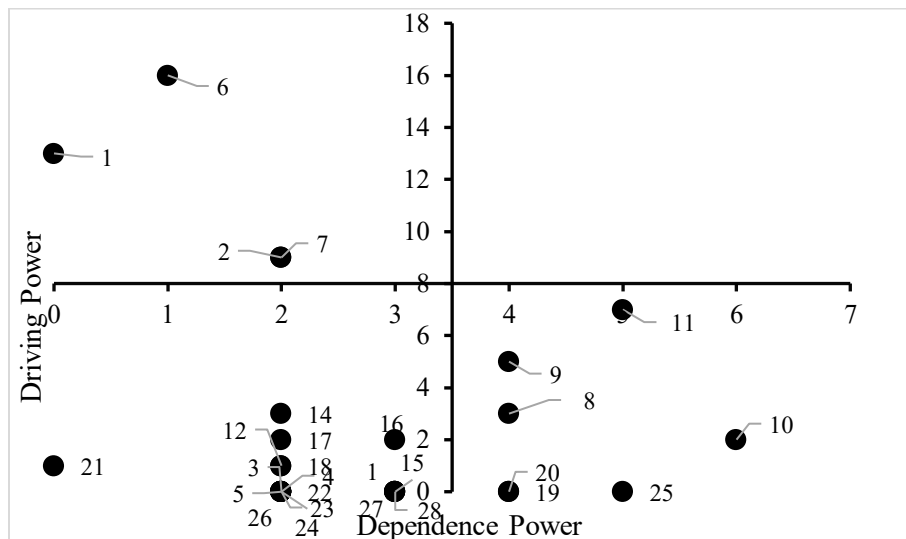


Figure 1 Out-in-degree centrality diagram

Using an adjacency matrix as input, easily generating a network using various SNA software tools is possible. In the demonstrative study, the adjacency matrix from Table 2 was employed as input in the NetMiner software package to create a network comprising 28 nodes and 74 arcs, as shown in Figure 2. The size of the nodes in this network corresponds to their respective “out-degree” values. Furthermore, different colours are assigned to nodes representing SRs based on their classification determined in Step 2.

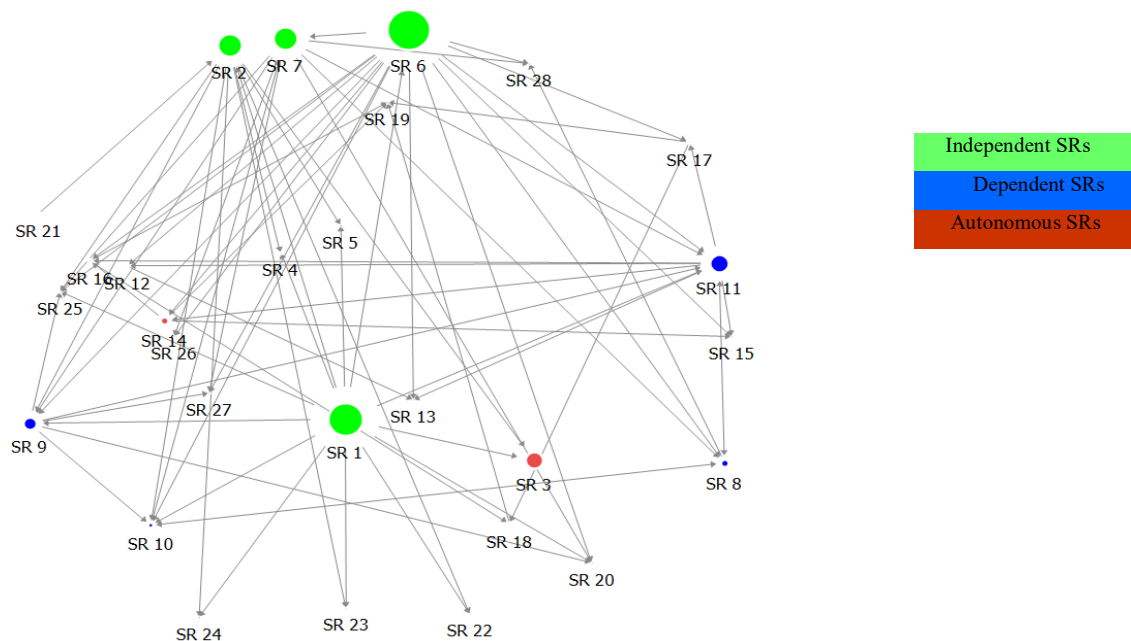


Figure 2 Network of *SRs*

5 Discussion

This study suggests utilising an SNA-based approach to model and analyse the dependencies among SRs in the early phase of a software development project.

One advantage of this approach is that it enables development teams to classify SRs into four distinct categories (autonomous, dependent, linkage, and independent) based on their interdependencies. Along with evaluating their betweenness centrality values, this classification assists the development team in

comprehending the interdependencies among SRs. This understanding helps development teams pinpoint the most critical SRs, enabling them to manage changes effectively and efficiently.

The most critical SRs are those classified as independent or linkage and those with a high betweenness centrality value. This is because alterations to these SRs will likely trigger changes in numerous other SRs, either directly or indirectly. For the demonstrative development project, Figure 1 shows that there are four independent SRs (1, 2, 6, and 7) but no linkage SRs. Hence, these four independent SRs, along with SR 11, which has the highest betweenness centrality value, are of paramount importance. Changes in these SRs should be minimised unless they are essential to meet customer needs. Furthermore, it is crucial to consider these SRs more carefully to mitigate the likelihood of rework caused by errors occurring in the development process. Failure to do so could result in substantial impacts on both project cost and duration.

Furthermore, classifying SRs into the groups mentioned above and their computed betweenness centrality values provides the development team with a framework for comparing alternative changes. For example, if a choice must be made between two alternative SRs in the same group, the one with a lower betweenness centrality value should be selected. Similarly, in the case of changes needed in two alternative SRs from different groups (e.g., one independent and the other dependent), and if their impact on meeting customer requirements is considered equal, it is recommended to prioritise the dependent SR_{*i*}.

Apart from analysing relationships, this approach empowers development teams to visualise SRs and their direct and indirect relationships through a network of arcs and nodes. Visualisation proves valuable because visual feedback facilitates faster and more thorough comprehension in tasks involving multidimensional information processing than numerical feedback (Hoffman *et al.*, 1981).

The network serves as a valuable tool for identifying potential paths of change propagation that may result from an initial modification in SR_i . However, when the generated SR network is complex, existing algorithms for identifying paths between particular nodes can be employed to determine these change propagation paths, for example, Migliore et al. (1990) and Tarjan (1972). Using these algorithms requires defining a starting and ending node for each path. The starting node corresponds to SR_i , where the change is initiated, while the ending node is the SR_i with an out-degree value of zero. For example, the potential change propagation paths triggered by a change in SR 11 are as follows:

11 -> 12 -> 13, 11 -> 13, 11 -> 14 -> 15, 11 -> 15, 11 -> 14 -> 16 -> 18 -> 19, 11 -> 14 -> 19, 11 -> 17 -> 18 -> 19, 11 -> 8 -> 10 -> 7 -> 9 -> 20, 11 -> 8 -> 10 -> 7 -> 20, 11 -> 8 -> 10 -> 7 -> 9 -> 25, 11 -> 8 -> 10 -> 7 -> 25, 11 -> 8 -> 10 -> 7 -> 26, 11 -> 8 -> 10 -> 7 -> 9 -> 27, 11 -> 8 -> 10 -> 7 -> 27, 11 -> 8 -> 10 -> 7 -> 28, and 11 -> 8 -> 28.

These paths were determined by utilising the NetMiner software package, which employs a search technique rooted in the algorithm that Migliore et al. (1990 developed).

6 Conclusions

Dealing with planned and unforeseen changes in SRs is typical in software development. A significant hurdle in handling these changes arises from the need to evaluate their impacts during the requirements analysis at the beginning of the development stage, especially in projects involving developing software from scratch. To address these challenges, it is essential to provide development teams with easy-to-use tools that facilitate the modelling and analysis of

dependencies among SRs during the initial stages of the development process. For this purpose, this study proposed utilising an SNA-based approach.

The approach uses four metrics to analyse the relationships among SRs: density, betweenness centrality, driving power, and dependence power. The latter two metrics categorise SRs into distinct groups based on their interconnections. This categorisation enables a development team to prioritise SRs for improvement initiatives and identify the most pivotal SRs that require the utmost attention to prevent rework.

Additionally, the approach assists in comprehending the relationships among SRs by creating a network. Such a network is valuable for assessing the consequences of changes, as it helps identify possible paths of change propagation resulting from an initial change in an SR_i . These paths can be identified from the network or by employing one of the path search methods.

Similar to any research undertaking, this study has specific limitations that should be considered in future research endeavours. For instance, this study utilised a binary network for modelling the dependencies among SRs in the early phase of a new software development project. For updating or improving existing software applications, a future study could consider weighted relationships among SRs instead of considering the relationships among SRs in a binary form. Moreover, this study relied on only four metrics of SNA—density, out-degree centrality, in-degree centrality, and betweenness centrality—for analysing the dependencies among SRs. Thus, future studies may explore the use of additional metrics. Another challenge for subsequent research is integrating the “change to” dependency type, as considered in this study, with other types of dependencies.

Last, the approach's practicability was illustrated through a single real-world software development project. Nevertheless, the results obtained from this practical example can act as a

stimulus for researchers and professionals to investigate how the approach can be applied to a broader range of real-world projects, including those of different types and varying levels of complexity.

References

- Al Zaabi, H. and Bashir, H. (2020) 'Modelling and analysing project interdependencies in project portfolios using an integrated social network analysis-fuzzy TOPSIS MICMAC approach', *International Journal of System Assurance Engineering and Management*, Vol. 11, pp. 1083–1106, <https://doi.org/10.1007/s13198-020-00962-3>.
- Ali, N. and Lai, R. (2016) 'A method of requirements change management for global software development', *Information and Software Technology*, Vol. 70, pp. 49-67, <https://doi.org/10.1016/j.infsof.2015.09.005>.
- Arora, C., Sabetzadeh, M., Goknil, A., Briand, L.C. and Zimmer, F. (2015) 'Change impact analysis for natural language requirements: An NLP approach', *Proceedings of the 23rd International Requirements Engineering Conference (RE)*, Ottawa, August 24-August 28, <https://ieeexplore.ieee.org/xpl/conhome/7310793/proceeding>.
- Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P. and Tsiridis, N. (2022) 'A metric for quantifying the ripple effects among requirements', *Software Quality Journal*, Vol. 30, pp. 853–883, <https://doi.org/10.1007/s11219-021-09581-y>.
- Bashir, H. and Ojiako, U. (2020) 'An integrated ISM-MICMAC approach for modelling and analysing dependencies among engineering parameters in the early design phase', *Journal of Engineering Design*, Vol. 31, No. (8-9), pp. 461-483, <https://doi.org/10.1080/09544828.2020.1817347>.
- Bashir, H., Araci, Z.C., Obaideen, K. and Alsyouf, I. (2023b) 'An approach for analysing and visualising the relationships among key performance indicators for creating sustainable campuses in higher education institutions', *Environmental and Sustainability Indicators*, Vol. 19, 100267, <https://doi.org/10.1016/j.indic.2023.100267>.
- Bashir, H., Hamdan, S., Ojiako, U., Haridy, S., Shamsuzzaman, M. and Al Zarooni, H.A. (2023a) 'A weighted fuzzy social network analysis-based approach for modelling and analysing relationships among risk factors affecting project delays', *Engineering Management Journal*, <https://doi.org/10.1080/10429247.2022.2162305>.
- Bashir, H., Ojiako, U., Marshall, A., Chipulu, M. and Yousif, A.A. (2022) 'The analysis of information flow interdependencies within projects', *Production Planning & Control*, Vol. 33, No. 1, pp. 20-36, <https://doi.org/10.1080/09537287.2020.1821115>.
- Bendoly, E. (2014) 'System dynamics understanding in projects: information sharing, psychological safety, and performance effects', *Production and Operations Management*, Vol. 23, No. 8, pp. 1352-1369, <https://doi.org/10.1111/poms.12024>.
- Bhatti, M., Ehsan, N., Ishaque, A., Hayat, F., Ahmed, S. and Sarwar, S. (2010) 'An investigation of changing requirements with respect to development phases of a software project', In: *2010*

International Conference on Computer Information Systems and Industrial Management Applications (CISIM), IEEE, May, pp. 323-327.

- Bhosale, V. and Kant, R. (2016) 'An integrated ISM fuzzy MICMAC approach for modelling the supply chain knowledge flow enablers', *International Journal of Production Research*, Vol. 54, No. 24, pp.7374-7399, <https://doi.org/10.1080/00207543.2016.1189102>.
- Bohner, S.A. (2002) 'Software change impacts-an evolving perspective', In: *International Conference on Software Maintenance, 2002. Proceedings*. IEEE, October, pp. 263-272, <https://doi.org/10.1109/ICSM.2002.1167777>.
- Bura, D. and Choudhary, A. (2020) 'A novel change impact model for enhancing project management', *International Journal of Project Organisation and Management*, Vol. 12, No. 2, pp. 119-132, <https://dx.doi.org/10.1504/IJPOM.2020.106373>.
- Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B. and Natt och Dag, J. (2001) 'An industrial survey of requirements interdependencies in software product release planning', In: *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, IEEE*, pp. 84–91, <https://doi.org/10.1109/ISRE.2001.948547>.
- Conejero, J.M., Figueiredo, E., Garcia, A., Hernandez, J. and Jurado, E. (2012) 'On the relationship of concern metrics and requirements maintainability', *Information and Software Technology*, Vol. 54, No. 2, pp. 212–238, <https://doi.org/10.1016/j.infsof.2011.09.003>.
- Dahlstedt, Å.G. and Persson, A. (2005) 'Requirements interdependencies: state of the art and future challenges', In: *Engineering and Managing Software Requirements*, Springer, pp. 95-116. https://doi.org/10.1007/3-540-28244-0_5.
- DeSanctis, G. (1984) 'Computer graphics as decision aids: Directions for research', *Decision Sciences*, Vol. 15, No. 4, pp. 463-487, <https://doi.org/10.1111/j.1540-5915.1984.tb01236.x>.
- Duperrin, J. and Godet, M., 1973. Méthode de Hiérarchisation des éléments d'un système: Essai de prospectivité du système de l'énergie nucléaire dans son contexte sociétal [Hierarchy method elements of a system: Test system prospectively of nuclear energy in its societal context]. Rapport CEA-R-4541. Commissariat à l'Energie Atomique [Commissioner for Atomic Energy]. Available at: <https://hal-lara.archives-ouvertes.fr/hal-02185432/document> (Accessed: February 2, 2025).
- Fu, Y., Li, M. and Fuzan, C. (2012) 'Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix', *International Journal of Project Management*, Vol. 30, No. 3 pp. 363–373, <https://doi.org/10.1016/j.ijproman.2011.08.004>.
- Godara, D. and Singh, R.K. (2017) 'Exploring the relationships between design measures and change proneness in object-oriented systems', *International Journal of Software Engineering, Technology and Applications*, Vol. 2, No. 1, pp. 64–80, <https://doi.org/10.1504/IJSETA.2017.086931>.
- Godara, D., Choudhary, A. and Singh, R.K. (2018) 'Predicting change prone classes in open source software', *International Journal of Information Retrieval Research (IJIRR)*, Vol. 8, No. 4, pp. 1–23, <https://doi.org/10.4018/IJIRR.2018100101>.

- Goknil, A., Kurtev, I. and van den Berg, K. (2008) 'Change impact analysis based on formalisation of trace relations for requirements', *4th ECMFA Traceability Workshop*, <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3047b3eff6ae5eaa4825608ef80ebec63b93182#page=59>.
- Goknil, A., Kurtev, I., van den Berg, K. and Spijkerman, W. (2014) 'Change impact analysis for requirements: A metamodeling approach', *Information and Software Technology*, Vol. 56, No. 8, pp. 950–972, <https://doi.org/10.1016/j.infsof.2014.03.002>.
- Hashmi, S.I., Lane, S., Karastoyanova, D. and Richardson, I. (2010) 'A CMMI based configuration management framework to manage the quality of service based applications', in *EuroSPI: Industrial Proceedings of the 17th European Conference on Software Process Improvement*, Grenoble, France, 1–3 September.
- Hassine, J., Rilling, J., Hewitt, J. and Dssouli, R. (2005) 'Change impact analysis for requirement evolution using use case maps', *8th International Workshop on Principles of Software Evolution (IWPSE '05)*, Lisbon, Portugal.
- Herrera, R.F., Matus, J., Santelices, C. and Atencio, E. (2020) 'Interaction between project management processes: a social network analysis', *International Journal of Project Organisation and Management (IJPOM)*, Vol. 12, No. 2, pp. 138-148, <https://dx.doi.org/10.1504/IJPOM.2020.106374>.
- Hoffman, P.J., Earle, T.C. and Slovic, P. (1981) 'Multidimensional functional learning (MFL) and some new conceptions of feedback', *Organizational Behaviour and Human Performance*, Vol. 27, No. 1, pp. 75-102, [https://psycnet.apa.org/doi/10.1016/0030-5073\(81\)90040-4](https://psycnet.apa.org/doi/10.1016/0030-5073(81)90040-4).
- Hughes, D., Dwivedi, Y., Rana, N. and Simintiras, A. (2016) 'Information systems project failure – analysis of causal links using interpretive structural modelling', *Production Planning & Control*, Vol. 27, No. 16, pp. 1313-1333, <https://doi.org/10.1080/09537287.2016.1217571>.
- Hwang, C. and Lin, M. (1987) *Group Decision Making under Multiple Criteria Methods and Applications*, Berlin Heidelberg, Springer-Verlag.
- Iriarte, C. and Bayona, S. (2020) 'It projects success factors: A literature review', *International Journal of Information Systems and Project Management*, Vol. 8, No. 2, pp. 49–78, <https://aisel.aisnet.org/ijispm/vol8/iss2/4>.
- Jawad, A.N.A. and Bashir, H. (2015) 'Hierarchical structuring of organisational performance using interpretive structural modelling', in *2015 International Conference on Industrial Engineering and Operations Management (IEOM)*, IEEE, March, pp. 1-7, doi: 10.1109/IEOM.2015.7093829.
- Jayatilleke, S. and Lai, R. (2018) 'A systematic review of requirements change management', *Information and Software Technology*, Vol. 93, pp. 163-185, <https://doi.org/10.1016/j.infsof.2017.09.004>.
- Jayatilleke, S., Lai, R. and Reed, K. (2017) 'A method of requirements change analysis', *Requirements Engineering*, Vol. 23, pp. 493-508, <https://doi.org/10.1007/s00766-017-0277-7>.
- Jones, C.T. (2007) *Estimating Software Costs*, 2nd ed., McGraw-Hill, New York.

- Kumar, S. and Rishi, R. (2016) 'Metrics to develop high quality software', *Indian Journal of Science and Technology*, Vol. 9, No. 48, pp. 1–6, <https://ijcsse.org/published/volume5/issue8/p3-V5I8.pdf>.
- Li, C., van den Akker, M., Brinkkemper, S. and Diepen, G. (2010) 'An integrated approach for requirement selection and scheduling in software release planning', *Requirements Engineering*, vol. 15, no. 4, pp. 375–396, May. <https://doi: 10.1007/S00766-010-0104-X>
- Liu, X. and Smolka, S.A. (1998) 'Simple linear-time algorithms for minimal fixed points,' In: K.G. Larsen, S., Skyum, and G. Winskel, G. (eds.) *Automata, Languages and Programming. ICALP 1998. Lecture Notes in Computer Science*, Vol. 1443, Springer, Berlin, Heidelberg, pp. 53-66, <https://doi.org/10.1007/BFb0055040>.
- Lu, H., Zhou, Y., Xu, B., Leung, H. and Chen, L. (2012) 'The ability of object-oriented metrics to predict change-proneness: a meta-analysis', *Empirical Software Engineering*, Vol. 17, No. 3, pp. 200–242, <https://doi.org/10.1007/s10664-011-9170-z>.
- Malhotra, R. and Jangra, R. (2017) 'Prediction and assessment of change prone classes using Statistica and machine learning techniques', *Journal of Information Processing Systems*, Vol. 13, No. 4, pp. 788-804, <https://doi: 10.3745/JIPS.04.0013>.
- Nejati, S., Sabetzadeh, M., Arora, C., Briand, L.C., and Mandoux, F. (2016) 'Automated change impact analysis between SysML models of requirements and design', *24th International Symposium on Foundations of Software Engineering (FSE'16)*, Seattle, USA, <https://doi.org/10.1145/2950290.2950293>.
- Noviyanto, F., Razali, R. and Ahmad Nazri, M.Z. (2023) 'Understanding requirements dependency in requirements prioritisation: a systematic literature review', *International Journal of Advances in Intelligent Informatics*, Vol. 9, No. 2, pp. 249-272, <https://doi.org/10.26555/ijain.v9i2.1082>.
- Priyadi, Y., Djunaidy, A., and Siahaan, D. (2019) 'Requirements Dependency Graph Modeling on Software Requirements Specification Using Text Analysis', In: *1st International Conference on Cybernetics and Intelligent System (ICORIS)*, IEEE, pp. 221-226, <https://doi.org/10.1109/ICORIS.2019.8874920>
- Remington, K. and Soderholm, A. (2010) 'Time: one factor influencing the project management of change', *International Journal of Project Organisation and Management*, Vol. 2, No. 3, pp. 221–235, <https://doi.org/10.1504/IJPOM.2010.035583>.
- Sage, A.P. (1979) *Methodology for Large-Scale Systems*, New York: McGraw-Hill.
- Samosir, H. and Siahaan, D. (2018) 'Generating requirement dependency graph based on class dependency', *IPTEK The Journal for Technology and Science*, Vol. 29, No. 2, p. 56-64, <https://pdfs.semanticscholar.org/5a04/5eac35c3ea9de550cc500483d04a059354d.pdf>.
- Sangal, N., Jordan, E., Sinha, V., and Jackson, D. (2005) 'Using Dependency Models to Manage Complex Software Architecture', in *Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pp. 167-176, https://web.cs.wpi.edu/Research/trg/public/papers/Sangal_LDM_oopsla05-dsm.pdf.

- Strens, M. and Sugden, R. (1996) 'Change analysis: a step towards meeting the challenge of changing requirements', In: *Proceedings, IEEE Symposium and Workshop on, IEEE*, pp. 278-283, <https://doi.org/10.1109/ECBS.1996.494539>.
- Thakurta, R. (2015) 'Projects as temporary organisations: insights from requirement volatility', *International Journal of Project Organisation and Management*, Vol. 7, No. 2, pp. 184-202, <https://doi.org/10.1504/IJPOM.2015.069617>.
- Thakurta, R. and Ahlemann, F. (2011) 'Understanding requirement volatility in software projects', *Engineering Management Journal (EMJ)*, Vol. 23, No. 3, pp. 3–7, <https://doi.org/10.1080/10429247.2011.11431900>.
- Wasserman, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press, New York.
- Widiastuti, M. and Siahaan, D. (2008) 'Mapping the impact of requirement changes using (LTRC)', In. *4th International Conference Information & Communication Technology and System*, pp. 315-319.
- Wieggers, K.E. (1999) *Software Requirements*, Redmont, Washington: Microsoft Press.
- Yin, L., Liu, Y., Shen, Z., and Li, Y. (2018) 'A technique to predict software change propagation', in *The 8th International Conference on Computer Engineering and Networks (CENet2018)*, Nanjing 210007, China, Springer, pp. 538-543, https://doi.org/10.1007/978-3-030-14680-1_59.
- Young, R.R. (2001) *Effective Requirements Practices*, Boston: Addison-Wesley.