Employing machine learning techniques to assess requirement change volatility

Phyo Htet Hein¹ · Elisabeth Kames² · Cheng Chen³ · Beshoy Morkos³

Received: 19 December 2019 / Revised: 24 November 2020 / Accepted: 26 November 2020 / Published online: 5 January 2021
© The Author(s) 2021

Abstract

Lack of planning when changing requirements to reflect stakeholders' expectations can lead to propagated changes that can cause project failures. Existing tools cannot provide the formal reasoning required to manage requirement change and minimize unanticipated change propagation. This research explores machine learning techniques to predict requirement change volatility (RCV) using complex network metrics based on the premise that requirement networks can be utilized to study change propagation. Three research questions (RQs) are addressed: (1) Can RCV be measured through four classes namely, multiplier, absorber, transmitter, and robust, during every instance of change? (2) Can complex network metrics be explored and computed for each requirement during every instance of change? (3) Can machine learning techniques, specifically, multilabel learning (MLL) methods be employed to predict RCV using complex network metrics? RCV in this paper quantifies volatility for change propagation, that is, how requirements behave in response to the initial change. A multiplier is a requirement that is changed by an initial change and propagates change to other requirements. An absorber is a requirement that is changed by an initial change, but does not propagate change to other requirements. A transmitter is a requirement that is not changed by an initial change, but propagates change to other requirements. A robust requirement is a requirement that is not changed by an initial change and does not propagate change to other requirements. RCV is determined using industrial data and requirement network relationships obtained from previously developed Refined Automated Requirement Change Propagation Prediction (R-ARCPP) tool. Useful complex network metrics in highest performing machine learning models are discussed along with the limitations and future directions of this research.

Keywords Complex system design · Engineering change · Requirement change propagation · Requirement volatility · Complex network · Multilabel learning

1 Engineering change

Engineering change (EC) is the process of modifying the function or properties of a component or product (Huang et al. 2003; Hamraz et al. 2013a). Engineering change management (ECM) describes the organization and control of said changes (Hamraz et al. 2013b). ECs account for about 30% of the work effort in an engineering project (Fricke et al. 2000; Langer et al. 2012). Changes are guaranteed to happen throughout the design or manufacturing process of a product (Leng et al. 2016). Conversely, if properly managed, changes can provide opportunity for improvement to the product and increase its consumer value. If improperly managed, they can increase product cost and production time. 50% of companies considered ECs as a major source of problems (Acar et al. 1998). Studies have shown that ECs determine 70–80% of product cost (Mcintosh 1995;

✉ Beshoy Morkos
bmorkos@uga.edu

Phyo Htet Hein
phhein@fit.edu

Elisabeth Kames
ekames@floridapoly.edu

Cheng Chen
cheng.c@uga.edu

¹ Department of Mechanical Engineering, Florida Institute of Technology, Melbourne, FL, USA

² Department of Mechanical Engineering, Florida Polytechnic University, Lakeland, FL, USA

³ College of Engineering, University of Georgia, Athens, GA, USA

Fei 2011) and a single EC is estimated on average to take 120 days to implement: 40 days for design and development, 40 days for processing, and 40 days for implementation in the production line (Watts 1984; Rouibah and Kevin 2003; Shankar et al. 2012). ECM has become an important research topic as ECs can affect the overall health of the project (Hein et al. 2017a). Propagation phenomenon in change context describes the chain reaction that transpires when a single change causes other subsequent changes to occur. Given the interconnectivity of the components in an engineering project, engineering change propagation is common. Engineering change propagation is one of the key concerns when implementing an initial change (Fricke et al. 2000; Clarkson et al. 2004). The cost of engineering change propagation has been shown to increase when change occurs toward the end of the design process (Clark and Fujimoto 1991). Therefore, it is integral to manage engineering change propagation early in the design process to ensure unanticipated changes do not occur downstream.

1.1 Requirement change propagation and volatility

Requirements are the foundation for any engineering project and define stakeholders' needs (Hull et al. 2005). Requirement changes evolve over time through the duration of a project due to changes in stakeholder expectations (Morkos et al. 2010a), changes to the requirements elicitation process (Morkos and Summers 2009), designer understanding of the product (Morkos et al. 2010), changing technologies, operational environments, and business needs (Ernst et al. 2009). The initial change to one requirement may cause other requirements to change due to functional and non-functional dependencies (Hein et al. 2017b). This phenomenon is known as "requirement change propagation". A single requirement change propagation occurs when a change in a requirement causes unforeseen propagations due to its relationships with other requirements (e.g., a change to a braking system's spatial requirement causes a change to the suspension's geometry requirement), and a cumulative requirement change propagation occurs when cumulative changes result in a change to another requirement (e.g., a change to the passenger capacity requirement and a change to fuel tank size requirement cumulatively result in a need to change the suspension weight rating, but would not individually cause this change in suspension weight rating) (Morkos 2012). While a single requirement change may seem to be manageable, the process becomes difficult when requirements change both in single and cumulative manners occur continuously throughout a project. Requirement change propagation can occur uncontrollably, resulting in unforeseeable and undesirable effects (Eckert et al. 2004) while also adding uncertainty to the design process (Eckert et al. 2004; Htet Hein et al. 2015). It is important to identify potential

propagation changes early in the design process, as more than 50% of a project's requirements will change before the end of the project (Kobayashi and Maekawa 2001; Morkos et al. 2012). Requirement changes and their propagations are proven to be more costly when they occur or are discovered closer to a project's conclusion, sometimes requiring significant redesign (Clark and Fujimoto 1991; Htet Hein et al. 2015). While it is not practical to prevent initial requirement change, the ability to manage unanticipated propagations early in the design process could mitigate negative effects such as added cost and time. In this paper, requirement change propagation is assessed through requirement change volatility (RCV). Requirement volatility here is not defined in terms of document volatility characterizing the extent of requirement addition, deletion, or modification between versions (Stark et al. 1998; Kulk and Verhoef 2008). Instead, we posit that requirement volatility is defined as RCV, that is, how individual requirements behave in response to an initial requirement change based on their interconnected dependencies. This volatile behavior is measured through four volatility classes: (1) multiplier, (2) absorber, (3) transmitter, and (4) robust, characterizing requirement's ability to multiply, absorb, transmit, or be robust to the initial change.

2 State of the art

Research in change management roots from studies in engineering design, decision theory, product development, complexity, graph theory, and design for flexibility (Wright 1997; Pikosz and Malmqvist 1998). A system may be characterized by its entities and their relations' structures, configurations, interactions, and responses (Kreimeyer and Lindemann 2011). In a change context, change can be regarded as an input to the system; change propagation can be regarded as the response of the system entities (component, subsystem, or system level) due to interactions along with their relationships.

2.1 Network approaches

Some of the earliest studies serving as seminal work involving application of network analysis to engineering systems are described in Braha and Bar-Yam (2004a, b, 2007), Braha (2016). In these studies, projects such as vehicle development, pharmaceutical facility development, hospital facility development, and software development are modeled using tasks or people or teams as nodes and the information flows between them as edges or links connecting the nodes in the networks constructed by structured interviews and process/system diagrams compiled by experts. However, it was a manual process requiring existing information of the systems and there is also no indication that the resulting network

relationships are without bias or subjective influence. First, they reported small-world properties of the networks—high clustering coefficient (tendency of nodes to cluster in interconnected modules) and short average path length (distance between any two nodes) and attributed it to, first, modular architecture of the networks and second, fast information transfer through the network resulting in immediate response to the rework created by other nodes. Second, the lower cut-off value of incoming links or degrees of nodes as defined by scale-free power-law distribution was reported and attributed in relation to limited information processing capability of a node and a far-stretched notion of bounded rationality, and furthermore, since this implies that there are very few nodes with large incoming links, the researchers claimed that interactions affecting a single node are limited and reduce the potential rework increasing the likelihood of problem convergence or task resolution. It was also reported that some nodes or tasks with more outgoing links than others may play the role of coordinators improving the integration and consistency of the development processes reducing the number of potential conflicts. Another attribution of this scale-free nature of degree distribution in terms of managerial strategy is that the networks are dominated by highly connected or central nodes that are likely vulnerable and they should be attended carefully to protect against uncertain disturbances. These are relevant observations based on how networks were modeled, but it was vaguely presented as to understand how they were validated against real data. To probe more into understanding of the roles of central nodes, the studies also analyzed degree centrality—measuring the importance of a node in terms of immediate number of incoming or outgoing links, closeness centrality—measuring how easily a node can reach to other nodes or how easily it is reachable from other nodes based on shortest path lengths, and betweenness centrality—measuring the extent to which a node lies on the paths between others serving as information flow connections or controllers. It was reported that nodes central in generating or consuming large information are observed based on degree centrality; that the degree centrality is highly correlated to closeness centrality, implying that direct connections provide useful information about indirect connections; that betweenness centrality is less correlated to either of other two measures, implying that information brokers may have different importance over the network not indicated by other two centrality measures; and that failures of central nodes should be avoided to improve performance and decrease vulnerability due to errors or changes. However, these studies did not employ comprehensive network metrics representing different aspects of complex network topology and it was not clear how real events or historical data validated such observations and recommendations. Even though the studies found a certain similarity between the real-world data and the development/

problem-solving tasks network models involving simulated rework with parameters characterizing network connectivity in explaining robustness and vulnerability in relation to problem convergence, the types of errors or changes were not clearly defined or explained. Most importantly, the findings presented in these studies were not explicitly related or did not apply to the changes in the requirements domain, let alone the requirement change volatility defined and studied in this paper. However, they have evidently inspired subsequent related researches to utilize network applications to engineering design, complexity, and change management.

Using Design Structure Matrices (DSMs), Suh and de Weck (2007) developed an index termed Change Propagation Index (CPI) calculated based on the number of incoming and outgoing edges of the component node to classify component's change propagation behavior as multiplier, carrier, absorber, and constant (Eckert et al. 2004). Giffin et al. employed a Change DSM overlaid with Component DSM and Change Propagation Frequency (CPF) matrix for change network motif (building blocks of network) analysis and quantification of components in terms of their Change Acceptance Index (CAI—ratio of number of implemented changes to number of proposed changes), Change Reflection Index (CRI—ratio of number of rejected changes to proposed changes), and CPI introduced by Suh et al. (2007), Giffin et al. (2009). In addition to product and change layer networks (Giffin et al. 2009), Pasqual and de Weck incorporated an engineer layer to the network and introduced metrics termed Engineer CPI and Propagation Directness (PD). Engineer CPI is calculated based on the number of incoming and outgoing workflows of the engineer in Engineer DSM, and PD is the geodesic shortest path from a component to another in Component DSM if there is a change propagated between those components in Change DSM (Pasqual and Weck 2011). Wang et al. utilized network centrality metrics and their derivatives, which reflect the importance of nodes in the network in spreading information to identify their correlation with the scope of change propagation in a software class structure network (Wang et al. 2014). Colombo et al. investigated the architectural network features such as the number of components, modules, components in modules, bus elements, etc., on change propagation behavior of complex technical systems (Colombo et al. 2015). Cheng and Chu adopted network centrality to develop “degree changeability” to assess direct change, “reach changeability” to assess indirect change, and “between changeability” to assess parts that will change (Cheng and Chu 2012).

Almost all the presented studies and their approaches rely on existing information of product or processes not suitable for early design use. Creation of networks or dependency relationships are manual and subjective. Many emphasize on select systems not generalizable across heterogeneous electromechanical systems. Although some network approaches

utilize network features or metrics in their own contexts, none of them explored or utilized a comprehensive list of metrics. Most importantly, none studied change specifically in terms of requirement change volatility as defined in this paper. Although the work by Menon taxonomized relevant complex network metrics in the existing literature to predict change propagation using regression analysis to statistically determine the change state (changed or not changed) of requirements due to change propagation (Menon 2015), it did not analyze change to the resolutions of RCV in terms of multiplicity, absorbance, transmittance, and robustness for the propagation potential of requirements, which is deemed to be an important capability in this paper to predict and manage change propagation.

The merit of this study is as follows: As in the R-ARCPP tool, the use of requirements allows for the representation of information at system or subsystem or component levels at early design stage, while physical architecture is still undefined. Building requirement networks from empirically tested natural language part of speech data of requirements leads to more generalized, automated, and objective networks. The utilization of a comprehensive network metrics taxonomy representing different network characteristics allows for better understanding of the role of local and global network connectivity, both in breadth and depth, in propagating requirement changes. Finally, deploying actual historical change data of real industrial case studies as training and test data, this study will identify a set of useful complex network metrics that can characterize RCV when employed in machine learning technique algorithms that are more capable of capturing complex information than other traditional techniques simply observing linear or non-linear trends. Consequently, the results will be analyzed to see whether their validity is satisfactory to provide practical requirement change management guidelines. If successful, the findings from this study will contribute toward the development of formalized computational reasoning tools capable of predicting and managing RCV, assisting designers and engineers along the design process.

2.2 Refined automated requirement change propagation prediction (R-ARCPP) tool

The Automated Requirement Change Propagation Prediction (ARCPP) tool by Morkos utilized a Design Structure Matrix (DSM) to represent requirement relationships, and is a valuable tool for change propagation prediction in the requirement domain (Htet Hein et al. 2015). The engineering change propagation prediction method uses engineering change notifications (ECNs) mapped to requirement changes. An ECN is studied to map it to requirements it affects. This requirement is then processed in a relationship DSM to obtain related requirements that may be affected. The consequent ECN is then checked for its correspondence with the affected requirements to determine if it could have been predicted. A relationship path length DSM identifies the shortest path length by which requirements are related to each other.

The ARCPP tool uses syntactical natural language data. Parts of Speech (POS) elements are parsed from requirement sentences using the automated Stanford POS tagger tool, shown to have a tagging accuracy 97.86% by its developers (Toutanova et al. 2003), to identify syntactical requirement relationships that are modeled to be used in the DSM model. While the original ARCPP tool uses physical domain relators (nouns), functional domain relators (verbs), and manually selected keywords to build these requirement relationships, the R-ARCPP tool used in this paper uses physical domain relators (nouns) representing only physical relationships as recommended by Hein et al. (2017b). Using this method, the completed Engineering Change Notifications (ECNs) and Engineering Changes (ECs) of the case studies are studied to identify changes which are translated to corresponding requirement changes. To model requirement change propagation, requirements relationships are built for the DSM using each requirement's nouns POS tags. Table 1 details how requirements are extracted using POS tags. For example, the noun “yarn” in the tagged set of requirement A is a POS word in the statement of requirement B, and therefore, there is a relationship from requirement A to B. However, none of the tagged nouns of requirement B are a POS word in requirement A, and therefore, there is no relationship from requirement B to A. This suggests that the relationships are not necessarily bidirectional between

Table 1 Example of POS tagged requirement statement (Hein et al. 2017a,b)

Requirement statements	Physical domain relators (noun POS elements)
<i>Req A:</i> The yarn comb shall have polished stainless-steel round pins mounted on zinc plated steel bar	[Yarn, comb, steel, pins, zinc, bar]
<i>Req B:</i> The supplier shall furnish all equipment, controls, data, drawings, etc. defined in the purchaser's documents and data to the following major components: yarn combs	[Supplier, equipment, controls, data, drawings]

requirements. Relationship selection and propagation analysis involve scoring requirement DSM based on the path length distance between requirements. More details of the tool are not discussed here for brevity, but can be consulted in Morkos (2012) and Hein et al. (2017b).

3 Proposed research

The objective of this research is to develop computational models to assess change propagation in terms of RCV using complex network metrics, as shown in Fig. 1. Requirement documents of industrial case studies will be processed in the R-ARCPP tool to produce requirement networks, from which complex network metrics of each requirement will be computed. It is important to note that the R-ARCPP tool's propagation analysis step is not executed, and the tool is only employed up to the point of producing effective requirement networks. Requirement relationships will be extracted from the networks and analyzed against actual requirement

change data to determine RCV. The four requirement volatility classes will be determined in terms of their metrics: multiplicity, absorbance, transmittance, and robustness. After complex network metrics and RCV class metrics are prepared, machine learning techniques will be applied to training data of requirement changes to generate computational models capable of determining RCV class metrics using complex network metrics. The resulting computational models will be validated against test data. Finally, model(s) that with the highest performance will be selected as generalized model(s). The following sections seek to outline the methods employed in the study in detail: Sect. 3.1 explains requirement volatility classes, Sect. 3.2 describes the explored complex network metrics, and Sect. 3.3 presents multilabel learning methods employed in this paper.

3.1 Requirement volatility classes

Brief definitions of four RCV classes are presented at the beginning of this paper. Table 2 documents their detailed

Fig. 1 Model development for predicting RCV

Table 2 Definitions of RCV classes

Change volatility classes	Definitions
Multiplier	A multiplier is a requirement that is changed by an initial change and propagates change to other related requirements
Absorber	An absorber is a requirement that is changed by an initial change, but does not propagate change to other related requirements
Transmitter	A transmitter is a requirement that is not changed by an initial change, but propagates change to other related requirements
Robust	A robust requirement is a requirement that is not changed by an initial change and does not propagate change to other related requirements

Fig. 2 Visual descriptions of requirement volatility classes

definitions with accompanying Fig. 2 providing a visualization, where requirement B is requirement of interest in all four cases.

3.1.1 Step 1: extraction of requirement relationships.

The closest works to the nature of change volatility defined in this paper are Suh et al. (2007), Giffin et al. (2009), where the degree of change propagation of a system element is measured through change propagation index (CPI) calculated based on its incoming and outgoing relationships, classifying them into multipliers, carriers, absorbers, and constants. However, in this paper, it is argued that the system element's change propagation propensity cannot be classified based on the net relationships alone. This is due to change that can propagate from that system element throughout the network along with all potential relationships to all other elements that it is directly or indirectly connected to, regardless of the number of net relationships. Additionally, change volatility definitions are based on the nature of requirements along with change propagation relationships, that is, their ability to

multiply, absorb, transmit, or be robust to change. Therefore, all potential requirement relationships stemming from a change are extracted and analyzed against the actual change propagation data of industrial case studies.

The first step in performing this analysis is to define all potential requirement relationships that are modeled in requirement networks, obtained from the R-ARCPP tool. A visualization of one example network of a case study is shown in Fig. 3. This network is the requirement network of a project involving the creation of yarn on a spool or comb using an automated creel system (Morkos 2012). The other two case study projects in this paper involve an assembly line design for the development of an exhaust gas recirculation bypass flap (Morkos 2012) and the development of a threading station for steel pipes (Morkos 2012). The brief backgrounds of all case studies are described in Sect. 3.6. There are 159 nodes and 9865 links in the first case study, 214 nodes and 9245 links in the second, and 202 nodes and 9558 links in the third case study. The diameter of a network is 4 in all case studies. The clustering coefficients are 0.68, 0.48, and 0.65, with the average path lengths 1.64, 1.88, and 1.98, respectively. Since all three networks have relatively high clustering coefficients and short average path length, they exhibit small-world properties as tasks/people information flow networks of vehicle development, pharmaceutical facility development, hospital facility development, and software development studied in Braha and Bar-Yam (2004a, b, 2007), Braha (2016). The maximum shortest path length between any two nodes is found to be four across all case studies. Therefore, up to path-length-four, up to walk-length-four, and the maximum number of paths from individual nodes to all other nodes are considered, accounting for the relationships' directions. It is expected that different requirements documents may have different maximum shortest path lengths, depending on the number of requirements and how they are written. As repeated changes are not observed from the case studies, cyclic relationships are not considered. All the potential relationships up to path-length-four with no cycling paths stemming from the initial changed requirements (source nodes) are extracted.

Fig. 3 Example of requirement network

3.1.2 Step 2: identification of change instances

The second step is to identify change instances from actual change data of industrial case studies. This requires a complete analysis of an engineering change notification (ECN) form (an example is shown in Fig. 4). If the engineering change in fact results in a change in requirements, requirements affected by that change are identified. In this manner, any initial requirement changes and all the subsequent requirement changes are identified resulting in a change relationship path of requirements. The actual change propagation relationship is therefore defined as a requirement relationship along which the initial requirement change travels or propagates changing all requirements that lie on it. It may contain up to (n) number of requirement nodes with $(n - 1)$ edges connecting them. It is observed from the ECN forms of the case studies that none of the actual change propagation relationships contains repeated changes. These relationships are taken as described in the ECNs, and no assumptions about further

possible propagations including cycling paths are made in this paper.

3.1.3 Step 3: volatility class metric scoring

After all potential requirement relationships are extracted and the actual change instances are identified, the final step is to check them against each other to score each requirement for volatility classes. The scoring schemes for the four RCV classes, shown in Table 3, depend on requirement's ability to multiply, transmit, absorb, or be robust to change, penalized by its path length (PL) distances to other nodes.

Table 4 describes a scoring example for a change instance using extracted, path-length-four, potential relationships. The table represents how change can stem from a single requirement and the various ways it can propagate to another requirement through intermediate requirements. The initial change is represented on the left-hand side, with the final changed requirement on the right-hand side. The actual change propagation instance is shown with the

Fig. 4 Engineering change notification (ECN) form

Date:	January 16, 2008	ECN#:	Company Line 3 Creel-01 Rev. 1
Customer:	Company	Customer PO #:	P42730-00
Project:	Line 3 Conversion Creel Only	Approved []	Rejected []
Client Signature:			
Comments:			
Change Notice Originated by:		John Smith	
Condition or Reason which Resulted in the Change:		Change in customer requirements	
Client Initiating Change:			
Brief Description of Change or Deviation from Scope: Replacement of manual tool for opening and closing of core locks with automated air locks. Includes independent control of 5 lower spindles.			
Estimated Impact on Engineering			
Schedule Delay		Explanation and breakdown:	
none			
Additional Engineering Expense		Explanation and breakdown:	
Engineering		\$	
Programming		\$	
Clerical		\$	
Additional Equipment/Installation Expense		Explanation and breakdown:	
Fabrication		\$	
Materials		\$	
Total Cost of this Change:		\$	

Table 3 Volatility class scoring scheme

Volatility class metric	Scoring scheme for requirement i on a relationship
Multiplicity (M)	Sum of multiplicative inverses of path lengths from it to nodes it multiplies the change to $M_i = \sum \frac{1}{PL_{i \rightarrow j} \mid \{(i,j) \in \{1,2,3,\dots,n\}, i \neq j\}}$ Both R_i and R_j changed
Absorbance (A)	Sum of multiplicative inverses of path lengths to it from changed nodes it absorbs the change from $A_i = \sum \frac{1}{PL_{j \rightarrow i} \mid \{(i,j) \in \{1,2,3,\dots,n\}, i \neq j\}}$ Both R_j and R_i changed
Transmittance (T)	Sum of multiplicative inverses of path lengths from it to nodes it transmits the change to $T_i = \sum \frac{1}{PL_{i \rightarrow j} \mid \{(i,j) \in \{1,2,3,\dots,n\}, i \neq j\}}$ R_i not changed. R_j changed
Robustness (R)	Sum of multiplicative inverses of path lengths to it from changed nodes it is not affected by $R_i = \sum \frac{1}{PL_{j \rightarrow i} \mid \{(i,j) \in \{1,2,3,\dots,n\}, i \neq j\}}$ R_i not changed. R_j changed

initial changed requirement (source node) colored in green, changed requirements (intermediate nodes) colored in gray, and final changed requirement (sink node) colored in red. It may contain up to (n) number of requirement nodes and $(n - 2)$ intermediate nodes with $(n - 1)$ edges connecting them. Each extracted path-length-four relationship has $(n = 5)$ total requirement nodes, $(n - 1 = 4)$ edges or path lengths connecting them, and $(n - 2 = 3)$ intermediate nodes between the source and the sink nodes. These potential

relationships can be divided into three groups: relationships stemming from the source node and ending at the sink node, relationships stemming from the source node and ending at a changed node which is not the sink, and relationships stemming from the source node and ending at an unchanged node. It should be noted that these are three different path occurrences and are not necessarily subnetworks as they may make up the entire network. Each group has $2^{n-2} = 2^3$ potential types of relationships based on the relative positions

Table 4 Example volatility scoring of a change instance

of changed and unchanged intermediate nodes. These relationships are checked against the actual change propagation instance to score each requirement volatility class metrics.

Group 1 (source-to-sink) contains the relationships that stem from the source node and end at the sink node. The intermediate nodes may or may not be changed nodes, but they propagate the change to the sink node. Therefore, they are either multipliers or transmitters of change. The source node is not given a volatility score as the initial change can be due to any reason and cannot be predicted. Path 1C is taken here as an example to explain the volatility metric scoring. In Path 1C, the first intermediate node is a transmitter as it is not changed, but it propagates change to the second intermediate

node and to the sink. The second intermediate node is a multiplier as it changed and it propagates change to the sink. The third intermediate node is a transmitter as it is not changed, but propagates change to the sink. The nodes on the remainder of the paths in the first group are scored similarly. Note that the term PL in the equations refers to path length.

Volatility scores of requirement nodes on path 1C

1st node (T)	$\frac{1}{PL(T \rightarrow M)} + \frac{1}{PL(T \rightarrow A)} = \frac{1}{1} + \frac{1}{3} = 1.33$
2nd node (M)	$\frac{1}{PL(M \rightarrow A)} = \frac{1}{2} = 0.5$
3rd node (T)	$\frac{1}{PL(T \rightarrow A)} = \frac{1}{1} = 1.0$
The sink (A)	$\frac{1}{PL(S \rightarrow A)} + \frac{1}{PL(M \rightarrow A)} = \frac{1}{4} + \frac{1}{2} = 0.75$

Group 2 (source-to-change nodes) contains the relationships that stem from the source node and end at a changed node which is not the sink. This changed end node is a change absorber and the intermediate nodes may either be multipliers or transmitters. This group is scored the same as the first group.

Group 3 (source-to-unchanged nodes) contains the relationships that stem from the source node and end at an unchanged node. This end node is a robust node, since it is not changed and does not propagate change further. The intermediate nodes may be robust to change or may transmit, multiply, or absorb change based on their relative positions along with the relationships. Again, the source node is not given any volatility score. Path 3F is taken here as an example to explain the volatility metric scoring. In Path 3F, the first intermediate node is a multiplier as it changed and propagates change to the third intermediate node. The second intermediate node is a transmitter, because it is not changed, but propagates change to the third intermediate node. The third intermediate node is an absorber as it changed, but does not propagate change further. The nodes on the remainder of the paths in the third group are scored similarly.

Volatility scores of requirement nodes on path 3F

1st node (M)	$\frac{1}{PL(M \rightarrow A)} = \frac{1}{2} = 0.5$
2nd node (T)	$\frac{1}{PL(T \rightarrow A)} = \frac{1}{1} = 1.0$
3rd node (A)	$\frac{1}{PL(S \rightarrow A)} + \frac{1}{PL(M \rightarrow A)} = \frac{1}{3} + \frac{1}{2} = 0.83$
End node (R)	$\frac{1}{PL(S \rightarrow R)} + \frac{1}{PL(M \rightarrow R)} + \frac{1}{PL(A \rightarrow R)} = \frac{1}{4} + \frac{1}{3} + \frac{1}{1} = 1.58$

It is important to note that only modified requirement changes are analyzed and changes resulting in addition or deletion of requirements are not considered. While each requirement node will have its own volatility class metrics, they will all be compiled into the same form where $\bar{x}_i = (x_{i1}, \dots, x_{iD})$ is the vector of D complex network metrics of requirement $i \in \{1, \dots, n\}$, and $\bar{v}_{ij} = (M_i, T_i, A_i, R_i)_j$ is the vector of volatility scores of requirement i for change instance $j \in \{1, \dots, m\}$. Based on its relative position on

multiple requirement relationships, a requirement node may have multiple volatility scores simultaneously; in other words, a changed requirement may serve as a multiplier or an absorber, and similarly, an unchanged requirement may serve as a transmitter or a robust requirement. For change instance j , each volatility class metric of requirement i is summed over all extracted potential relationships $r \in \{1, \dots, k\}$ and normalized by the maximum value with respect to the other requirements. Consider the normalized multiplicity score of requirement i for change instance j as an example:

$$\widehat{M}_{ij} = \frac{\left(\sum_{r=1}^k M_{ij} \right) - \min \left(\sum_{r=1}^k M_{ij} : i \in \{i, \dots, n\} \right)}{\max \left(\sum_{r=1}^k M_{ij} : i \in \{i, \dots, n\} \right) - \min \left(\sum_{r=1}^k M_{ij} : i \in \{i, \dots, n\} \right)}.$$

This operation is performed for all change instances of a case study to determine a normalized score for each class metric. Each volatility class metric of requirement i is summed over all change instances and normalized by the maximum value with respect to the other requirements to produce final volatility scores values ranging from 0 to 1. Consider the final normalized multiplicity score of a requirement i as an example:

$$M_i = \frac{\left(\sum_{j=1}^m \widehat{M}_{ij} \right) - \min \left(\sum_{j=1}^m \widehat{M}_{ij} : i \in \{i, \dots, n\} \right)}{\max \left(\sum_{j=1}^m \widehat{M}_{ij} : i \in \{i, \dots, n\} \right) - \min \left(\sum_{j=1}^m \widehat{M}_{ij} : i \in \{i, \dots, n\} \right)}.$$

Since the aforementioned volatility metric scoring scheme is derived from the three groups of path lengths, as shown in Table 4, this measurement has been specifically designed for the requirement network at a cursory glance. However, the path lengths utilized in the scoring scheme are considered as separate snapshots of unidirectional individual paths (only the paths from the source to the sink, changed nodes, and unchanged nodes) activated for extraction based on the number of actual changes in the projects, independent of the whole connected network topology. Thus, the network-based RCV classes can cause high predictability for each result. Further analysis of any bias for centrality calculation should be examined to improve the prediction accuracy. Currently, predicting each RCV class metric can be treated as the classical AI black box problem, and the fundamental reason behind it is still yet to be investigated.

Given the observed sparsity of the volatility metric scores, they are divided into multiple ordered categorical

levels to improve the granularity of the data. For example, multiplicity scores can be divided into three intervals at two thresholds to classify requirements as having low, medium, or high multiplicity of change. An example excerpt of requirement volatility class metrics data is shown in Table 5, where $\vec{v}_i = (M_i, A_i, T_i, R_i)$ is the vector of final volatility class metrics of requirement i . The volatility class metrics serve as categorical dependent output variables in the explored machine learning techniques. By measuring each requirement by its volatility class metrics, RQ1 is addressed and summarized in Table 6.

3.2 Complex network metrics

Since this study is exploratory in nature, it uses a comprehensive list of complex network metrics to determine which metrics are useful in predicting the volatility of requirements. There are two categories, namely, *global descriptive metrics* and *pairwise descriptive metrics* considered. Global descriptive metrics describe the network topology

considering contributions from all nodes in the network. In global descriptive metrics, there are three subcategories of metrics termed *distance and clustering*, *centrality*, and *subgraph and community detection*. As their names imply, distance and clustering metrics refer to a node's distance to others, centrality metrics refer to a node's importance, and subgraph and community metrics refer to a node's properties

Table 5 Example excerpt of volatility class metrics

Req	Volatility $\vec{v}_i = (M_i, A_i, T_i, R_i)$
1	$\vec{v}_1 = (0, 0, 0.65, 0.20)$
2	$\vec{v}_2 = (0, 1, 0, 0)$
3	$\vec{v}_3 = (1, 0.67, 0, 0)$
...	...
n	$\vec{v}_n = (0.85, 0.25, 0.14, 0.37)$

Table 6 Summary for RQ 1

RQ 1	Can RCV be measured through four classes namely, (1) multiplier, (2) absorber, (3) transmitter, and (4) robust, during all instances of change?
Answer	RCV can be measured through four classes, (1) multiplier, (2) absorber, (3) transmitter, and (4) robust, in respective metric values during all instances of change utilizing requirement network relationships obtained from the R-ARCPP tool and the actual requirement change data of industrial case studies

in relation to the subgraph or community it belongs to. For example, one of the distance and clustering metrics, average neighbor degree of a node, is the average relationships or edges of its immediate neighbor nodes describing the connectedness of a node based on its neighbors (Barrat et al. 2004). One of the centrality metrics, closeness centrality of a node, indicates the node's importance based how reachable other nodes are from a give node by the mean geodesic distance between the node and any other node j averaged over all nodes in the network (Newman 2010; Dangelchev 2006; Freeman 1978; Valente et al. 2008). One of the subgraph and community detection metrics, communities metric or modularity, measures the quality of the communities or modules representing subnetworks of nodes having more intertwined relationships with each other (Newman 2006; Kantarci and Labatut 2013; Clauset et al. 2004; Leicht and Newman 2008). Since requirement networks studied in this paper are directed networks, metrics that account for directions of relationships are also considered. For example, metrics such as in-degree and out-degree are employed to study the connectedness of a requirement node in terms of the number of relationships directed to it from other requirement nodes and the number of relationships originating from it to other requirement nodes. Pairwise descriptive metrics—for example, walk length and path length—quantify interactions between change initiated and propagated node pairs. In this study, pairwise descriptive metrics are modified as the pairwise descriptive metrics in Menon's work measured only the interactions of change initiated and change propagated node pairs (Menon 2015). Conversely, this research determines four RCV classes of all requirement nodes from their corresponding complex network metrics which are dependent on all requirement network relationships. Therefore, up to path-length-four, up to walk-length-four, and the maximum number of paths from individual requirement nodes to all other requirement nodes. Table 7 details the comprehensive list of selected complex network metrics. The detailed explanations of these metrics are not given here for brevity and can be explored in the corresponding references.

The complex network metrics, except the metrics with categorical values, are normalized by the maximum value with respect to the other requirements to produce final scores ranging from 0 to 1. Table 8 illustrates an example excerpt of the complex network metrics data, where $\vec{x}_i = (x_{i1}, \dots, x_{iD})$ is the input variable vector of requirement i with D complex

network metrics. The complex network metrics serve as continuous or discrete independent input variables in the explored machine learning techniques. By incorporating complex metrics to measure requirements, RQ2 is addressed and summarized in Table 9. It is important to note that the answer to RQ 2 is not trivially true as graph theory metrics could be explored and computed for various change instances, but not reveal any meaningful results that could be used to explain or understand change propagation occurrence or volatility phenomenon.

3.3 Multilabel learning methods

To address the third research question, which seeks to explore if machine learning techniques can be used to predict requirement volatility metrics using complex network metrics, multilabel learning (MLL) methods are explored. MLL methods are the supervised machine learning techniques employed in real-world applications such as gene classification, medical diagnosis, document classifications, music and video annotation, image recognition, text categorization, and more (Tawiah and Sheng 2013). In MLL, the task of learning is a mapping from each data point, $x \in X$, to a set of outputs or labels $y \subseteq L$ (Tsoumakas et al. 2007; Madjarov et al. 2012). The labels are not assumed to be mutually exclusive, that is, multiple labels may be associated with a single data point, meaning that a data point can be a member of more than one class. MLL methods can therefore model all four volatility classes simultaneously. They employ machine learning classification algorithms and are considered a promising approach for requirement volatility classification problem. Table 10 illustrates an example multilabel data in which the label space of requirement i will be $L = \{\lambda_{1i}, \lambda_{2i}, \dots, \lambda_{Qi}\}$ with $Q = 4$ for four volatility classes, meaning $\{\lambda_{1i}, \lambda_{2i}, \lambda_{3i}, \lambda_{4i}\} = \{M_i, A_i, T_i, R_i\}$ with discrete values 0 or 1, indicating if a requirement belongs to a volatility class or not. If each volatility class metric is partitioned at multiple intervals to represent different severity levels, Q will be equal to four times the number of levels. For example, if each volatility class metric is divided into three levels (low, medium, and high), there will be 12 labels in the output label space. The example input space will be $\forall x_i \in X$, with input variable vector $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ for requirement i with D complex network metrics. Throughout this paper, the terms learning, learner, inputs, and outputs may be

Table 7 Selected network metrics

ID	Category	Name	Source
1	Distance and clustering metrics	Eccentricity	Dankelmann et al. (1999)
2		Efficiency	Latora and Marchiori (2001)
3		Vulnerability	Costa et al. (2007)
4		Clustering Coefficient	Newman (2003)
5		degree	Newman (2010)
6		In—Degree	Newman (2010)
7		Out—Degree	Newman (2010)
8	Centrality metrics	Average Neighbor Degree	Barrat et al. (2004)
9		In—Closeness Centrality	Newman (2010)
10		Out—Closeness Centrality	Newman (2010)
11		Delta Centrality	Latora and Marchiori (2007)
12		Bridging Centrality	Hwang et al. (2006)
13		Bridging Coefficient	Hwang et al. (2006)
14		Random Walk Centrality	Bollobás (2001)
15		Betweenness Centrality	Newman (2010)
16		Counting Betweenness	Bloechl et al. (2010)
17		Flow Coefficient	Honey et al. (2007)
18		Total Flow Paths	Honey et al. (2007)
19		Brokering Coefficient	Konganti et al. (2013)
20		Right Eigenvector Centrality	Newman (2010)
21		Left Eigenvector Centrality	Newman (2010)
22		Alpha Centrality	Bonacich and Lloyd (2001)
23		Katz Centrality	Newman (2010)
24		Subgraph Centrality	Estrada and Rodríguez-Velázquez (2005)
25		Coreness Centrality	Newman (2010)
26		PageRank	Newman (2010)
27		HITS: Authority Centrality	Newman (2010)
28		HITS: Hubs Centrality	Newman (2010)
29	Subgraph and community metrics	Participation Coefficient	Guimerà and Amaral (2005), Rubinov and Sporns (2010)
30		Within Module Degree	Guimerà and Amaral (2005), Rubinov and Sporns (2010)
31	Modified pairwise descriptive metrics	Communities	Newman (2006)
32		Out—Path Length $\subseteq \{1,2,3,4\}$	Newman (2010)
33		In—Path Length $\subseteq \{1,2,3,4\}$	Newman (2010)
34		Out—Walk Length $\subseteq \{1,2,3,4\}$	Rucker and Rucker (2000)
35		In—Walk Length $\subseteq \{1,2,3,4\}$	Rucker and Rucker (2000)
39		Out—Maximum Number of Paths	Newman (2010)
40		In—Maximum Number of Paths	Newman (2010)

Table 8 Example excerpt of complex network metrics

Req	Complex network metrics
	$\vec{x}_i = (x_{i1}, \dots, x_{iD})$
1	$\vec{x}_1 = (0.56, \dots, 1)$
2	$\vec{x}_2 = (0.43, \dots, 4)$
3	$\vec{x}_3 = (0.14, \dots, 2)$
...	...
n	$\vec{x}_n = (0.08, \dots, 5)$

used interchangeably with the terms classification, classifier, independent variables (or features), and dependent variables (or labels), respectively. We do this to remain consistent with the terms used by the literature for each type of machine learning method.

Multilabel learning methods can be divided into two categories: Problem Transformation Methods (PTMs) and Algorithm Adaptation Methods (AAMs) (Tsoumakas et al.

Table 9 Summary for RQ 2

RQ 2	Can complex network metrics be explored and computed for each requirement?
Answer	The most relevant and up-to-date taxonomy of complex network metrics is explored, modified, and adopted, and the metrics are calculated from requirement networks obtained from the R-ARCPP tool using novel tools and state of the art existing tools in the literature

Table 10 Example of multilabel dataset

Example	D input variables	Q output labels
1	$x_1 = (x_{11}, x_{12}, \dots, x_{1D})$	$\{\lambda_1, \lambda_4\}$
2	$x_2 = (x_{21}, x_{22}, \dots, x_{2D})$	$\{\lambda_3, \lambda_4\}$
3	$x_3 = (x_{31}, x_{32}, \dots, x_{3D})$	$\{\lambda_1\}$
...	...	...
n	$x_n = (x_{n1}, x_{n2}, \dots, x_{nD})$	$\{\lambda_2, \lambda_3\}$

2007; Madjarov et al. 2012). Others (Madjarov et al. 2012; Gibaja and Ventura 2014) also categorize another method termed Ensemble Methods (EMs). PTMs are the algorithm independent multilabel learning methods that transform the multilabel learning problem into one or more single-label learning problems, on which a variety of single-label classification algorithms or classifiers can be applied (Tsoumakas et al. 2007; Gibaja and Ventura 2014). Machine learning algorithms such as decision trees (DT), Naïve Bayes (NB), support vector machines (SVM), nearest neighbors (NN), and artificial neural networks (ANNs) are such common classification techniques. AAMs are the multilabel learning methods that adapt, extend, and customize existing classifiers to directly perform multilabel classification on multilabel datasets (Tsoumakas et al. 2007; Madjarov et al. 2012; Gibaja and Ventura 2014). They are the extensions or customizations of the machine learning classifiers that can be taxonomized into such categories as decision trees, support vector machines, nearest neighbors, artificial neural networks, generative and probabilistic models, associative classifications, bio-inspired approaches, and ensembles (Gibaja and Ventura 2014). EMs are the multilabel learning methods developed on top of PTMs and AAMs, which can be divided into EMs of PTMs and EMs of AAMs (Madjarov et al. 2012). In EMs, the machine learning is a paradigm in which multiple classifiers are used to learn a set of hypotheses and combine them, in contrast to ordinary machine learning which only learns one hypothesis (Zhou 2015). An ensemble consists of several learners named base learners and the generalization of an ensemble is stronger. It is important to note that the MLL methods in this paper are employed due to their widespread acceptance in the literature. Widely used MLL methods are selected to cover diverse types of methods and comprehensively explore their computational capabilities, as shown in Fig. 5: nine PTMs

on which four base classifiers are applied and three AAMs. In addition, four feature selection filter methods as proposed in Spolaôr et al. (2013) are applied to the complex network metric inputs before running MLL method analyses. Using these four feature rankings, the number of complex network metrics is varied based on decreasing order of scores. *MULAN*, an open-source Java library for multilabel learning, is employed to perform the MLL analyses (Tsoumakas et al. 2011). The detailed descriptions of these MLL methods along with their parameters and the feature selection filter methods are given in Appendix 1.

3.4 Evaluation measures of multilabel methods

The performance evaluation measures for single-label learning methods and multilabel learning methods are similar; however, multilabel classification prediction models require slightly different measures to account for multiple labels (Tsoumakas et al. 2007). Unlike single-label prediction, in which the prediction is either correct or incorrect based on the confusion matrix or contingency table, multilabel predictions may be partially correct or incorrect according to labels (Santos et al. 2011). Comprehensive taxonomy and mathematical details of multilabel evaluation measures can be found in review papers such as (Gibaja and Ventura 2014). Only the most common measures used in studies such as Tawiah and Sheng (2013, Madjarov et al. 2012; Santos et al. 2011; Abdallah et al. 2015; Kavitha 2016), Pakrashi et al. (2016), Dimou et al. (2009), Nair-Benrekia et al. (2015), Heath et al. (2010), Kafrawy et al. (2015) are deployed here.

Tsoumakas, et al. suggested performance evaluation measures for multilabel learning that can be divided into two groups (Tsoumakas et al. 2009a). The first group is named bipartitions-based measures and the second is rankings-based measures. The bipartitions-based measures are calculated based on the comparison of the predicted relevant labels with the ground truth relevant labels (Madjarov et al. 2012; Kafrawy et al. 2015). This group can be further divided into example-based measures and label-based measures. The example-based measures are calculated from the average differences of the actual and the predicted sets of labels over all examples (Madjarov et al. 2012; Kafrawy et al. 2015; Tsoumakas et al. 2009a). There are six example-based measures deployed: hamming loss, subset accuracy, precision, recall, accuracy, and *F1* score (Madjarov et al.

Fig. 5 Employed multilabel learning methods

2012; Santos et al. 2011; Kafrawy et al. 2015; Tsoumakas et al. 2009a). The example-based measures range between 0 and 1 with higher values indicating higher performance, with the exception of hamming loss. The label-based measures decompose the process into separate evaluations for each label and then average over all labels (Madjarov et al. 2012; Kafrawy et al. 2015; Tsoumakas et al. 2009a). For binary classification confusion matrix, there are true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Any known measure for binary prediction such as

accuracy, precision, recall, $F1$ score, and the area under ROC curve (AUC)—the tradeoff between sensitivity ($f(TP, FN)$) and specificity ($f(TN, FP)$)—can be used as label-based measures and can be achieved through two operations: macro-averaging and micro-averaging (Tsoumakas et al. 2009a). Macro-averaging computes one measure for each label and then averages over all labels, whereas micro-averaging considers all examples of all labels together (Gibaja and Ventura 2014). Some, such as accuracy, have the same macro- and micro-averaged values, while others (precision, recall, $F1$,

and AUC score) are different in macro- and micro-averaged values (Tsoumakas et al. 2009a). The label-based measures range between 0 and 1 with higher values indicating higher performance.

The rankings-based measures compare the predicted ranking of the labels with the ground truth ranking (Madjarov et al. 2012; Kafrawy et al. 2015; Tsoumakas et al. 2009a). There are four rankings-based measures: one-error, coverage, ranking loss, and average precision (Tsoumakas et al. 2009a). The rankings-based measures range between 0 and 1 with smaller values indicating higher performance, with the exception of average precision (Madjarov et al. 2012; Santos et al. 2011; Heath et al. 2010; Tsoumakas et al. 2009a). One-error evaluates how many times the top-ranked label is not in the set of relevant labels of the example. Coverage evaluates the distance on average to go down the list of ranked labels to cover all the correct labels of the example. Ranking loss is the number of times that an incorrect label is ranked higher than a correct label. Average precision evaluates the average fraction of labels ranked above a label $\lambda \in Y_i$ that are actually in Y_i .

3.5 Statistical comparisons of MLL models

To compare these models, the distributions of their evaluation measures values are first analyzed, and it is observed that the evaluation measures do not follow normality for all volatility metrics. Figure 6 presents an example evaluation measure that does not follow normality. Therefore, a typical analysis of variance—which assumes normality of the data—cannot be used; rather, the Friedman test is performed to identify which model differs from which other models. The Friedman test (1937, 1940), a nonparametric

test without distributional assumptions, is used to test the null hypothesis of no difference in performance between all compared models (Salkind 2006). The test statistic can also be approximated by the Chi-squared distribution with $(K - 1)$ degree of freedom and corresponding p value (Salkind 2006, 2010). If the p value is less than the significance level, the null hypothesis is rejected, and the alternative hypothesis that there is at least one model different from at least one other model is accepted (Salkind 2006, 2010).

When the null hypothesis is rejected by the Friedman test, it is necessary to identify which model is different. Statistical tests that can determine this are post hoc multiple comparison tests which compare the models pairwise for all pairs. This paper employs the Nemenyi test (Demšar 2006), which is the most common post hoc multiple comparison test after the nonparametric Friedman test. Any two models are different significantly if their mean ranks differ by at least one Nemenyi's test statistic, critical difference (CD) (Demšar 2006). Another approach for multiple comparisons is any classical tests, such as the Friedman post hoc test for multiple comparisons (Derrac et al. 2011; García et al. 2010). The Friedman post hoc test and its statistic, widely considered to be a more statistically powerful approach, is performed in addition to the Nemenyi test. Additionally, the p values of the test are adjusted using Shaffer's static procedure making use of the logical relations of hypotheses as described in Shaffer (1986).

3.6 Case study background

Requirement and requirement change propagation data documented from three industrial case studies in varying format by different stakeholders are employed. This first study is a project that involves the creation of yarn on a spool or comb using an automated creel system (Morkos 2012). The second study pertains to an assembly line design for the development of an exhaust gas recirculation bypass flap (Morkos 2012). The last case study is the development of a threading station for pipes for a customer with annual capacity of 150 kilotons of pipe made of varying material grades (Morkos 2012). The projects' timelines range from 9 months to 2 years and costs range from 700,000 to 2,500,000 USD. There are total of 575 requirements and 16 requirements change propagation instances. The validity for the use of these projects is proven by their heterogeneity across problem domains, relative technical complexity, time and monetary investments, and overall success. Moreover, the studies can point out the models' ability to assist designers and engineers in managing the engineering change propagations in the requirement domain along the design process. The firm's data management system also provides readily available information for each project such as initiation documentation, requirement specifications, tasks and activities

Fig. 6 An Example of distribution of micro-AUC of a multilabel model

allocations, and budget updates. Engineering changes were found documented in engineering change notification documents which contained the data needed for performing requirement change propagation analysis.

4 Results and discussion

As presented in Fig. 5 and Appendix 1, four preprocessing feature selection methods—with seven steps of complex network metrics input, nine PTMs using four base classifiers, and three AAMs (total seven models: BPMLL, AdaBoost, MH, and five k values in MLkNN)—result in the total of 1204 $((4 \times 7) \times ((9 \times 4) + 7))$ models. Since there are 1204 models and 18 evaluation measures, the model comparisons for MLL methods are not performed based on individual evaluation measures, but rather across all evaluation measures ranging from 0 to 1. For each feature selection method, from three AAMs and nine PTMs using four base classifiers, only the highest performing model out of seven steps of complex network metrics inputs is selected, resulting in 39 $((9 \times 4) + 3) = 39$ models. Only the top 20% or 8 of these 39 models are selected from each feature selection method. For four feature selection methods, the total number of final models selected to compare is 32; only the results of these models are presented.

The Friedman test p value and Nemenyi test's CD of the compared models are shown in Fig. 7. The horizontal axis represents the average ranks of the models with higher ranks to lower ranks from left to right. The models with the same ranks are connected by a thin line. The models do not differ significantly if connected by the bold crossing line. The p value is close to zero, rejecting the null hypothesis. The seven bold lines of the Nemenyi test's CD plot also indicate that there are significant differences between the models that are not connected by the same line. For example, the first ten models connected by the first bold line, starting from model 24 to model 14, are not significantly different from each other. However, model 24 is different from model 11 as there is no CD line connecting them. Considering all seven CD lines and all models, most of the models are not significantly different.

The Friedman post hoc test with Shaffer's adjusted p values produces similar results. As shown in Fig. 8, the first ranked model (model 24) is different, as it has p values less than the significant level of 0.05 from model 10 to model 26. The second model and the third model, model 21 and model 19, are different from models 12 to 26. Model 23 is different from models 9 to 26. Models 17, 18, 20, 22, and 30 are different from models 31, 25, and 26.

The top 20% of these 32 models are selected as the highest performing MLL models, as shown in Table 11. To compare the models, the following naming scheme is used:

Fig. 7 Friedman test and Nemenyi test of top-performing MLL models

Fig. 8 Friedman post hoc test of top-performing MLL methods **Table 11** Evaluation measures of top seven MLL models

Evaluation measures	Model ID						
	M24: CLR_ NN_LPIG_7	M21: PS_NN_ LPIG_7	M19: ECC_ NN_LPIG_7	M23: RAKEL_ NN_LPIG_7	M17: BR_ NN_LPIG_7	M18: CC_ NN_LPIG_7	M20: LP_NN_ LPIG_7
Hamming Loss	0.028	0.029	0.030	0.028	0.028	0.028	0.028
Sub.Accuracy	0.905	0.900	0.893	0.903	0.903	0.903	0.903
Ex.Accuracy	0.947	0.945	0.942	0.946	0.946	0.946	0.946
Ex.Precision	0.961	0.961	0.961	0.961	0.961	0.961	0.961
Ex.Recall	0.962	0.961	0.956	0.961	0.961	0.961	0.961
Ex.F1	0.962	0.961	0.958	0.961	0.961	0.961	0.961
Mic.Precision	0.961	0.961	0.961	0.961	0.961	0.961	0.961
Mic.Recall	0.962	0.961	0.956	0.961	0.961	0.961	0.961
Mic.F1	0.962	0.961	0.958	0.961	0.961	0.961	0.961
Mac.Precision	0.839	0.839	0.873	0.812	0.812	0.812	0.812
Mac.Recall	0.866	0.820	0.845	0.865	0.865	0.865	0.865
Mac.F1	0.849	0.818	0.837	0.831	0.831	0.831	0.831
Mic.AUC	0.984	0.985	0.983	0.970	0.968	0.968	0.968
Mac.AUC	0.864	0.829	0.894	0.888	0.862	0.862	0.862
Avg.Precision	0.981	0.980	0.974	0.963	0.963	0.963	0.963
Coverage	3.281	3.316	3.406	3.531	0.531	3.531	3.531
One.Error	0.005	0.005	0.023	0.051	0.051	0.051	0.051
Ranking Loss	0.017	0.019	0.024	0.033	0.033	0.033	0.033

$\langle \text{MLLMethod} \rangle_ \langle \text{Classifier} \rangle_ \langle \text{FeatureSelectionMethod} \rangle_ \langle \# \text{ of ComplexNetworkMetrics} \rangle$.

For example, model 24 is CLR_NN_LPIG_7, meaning that it is the model using CLR PTM, NN classifier, LP-IG feature selection method, and the top seven complex network

metrics selected by the feature selection method. All the models use the NN classifier and the first seven complex network metrics by LP-IG feature selection method. However, the PTMs used are CLR, PS, ECC, RAKEL, BR, CC, and LP from model 24 to model 20, respectively. Out of the employed PTMs, both binary methods (BR and CC), two (PS and LP) out of three label combination methods, one pairwise method (CLR), and two (ECC and RAKEL) out of three EMs of PTMs rank as the top seven MLL methods. The evaluation measures are also similar across the models with slight differences. Surprisingly, there is no AAM in the top seven models. The bold values in the table represent the highest performing evaluation measures. Instances where there are multiple bold values for a single evaluation measure indicate that several models performed well and are tied for highest performance. The seven complex network metrics used by the models are in decreasing order of the LP-IG feature selection method: Katz centrality, brokering coefficient, authority centrality, left eigenvector centrality, outgoing walks of length three, PageRank, and incoming walks of length three. The example-based measures (hamming-loss, subset accuracy, precision, recall, accuracy, and *F1* score) range between 0 and 1 with higher values indicating higher performance, except for hamming loss. The label-based measures (accuracy, precision, recall, *F1* score, and AUC) in both macro-averaging and micro-averaging) range between 0 and 1 with higher values indicating higher performance. The rankings-based measures compare the predicted ranking of the labels with the ground truth ranking (Madjarov et al. 2012; Kafrawy et al. 2015; Tsoumakas et al. 2009a). The rankings-based measures (average precision, coverage, one-error, and ranking loss) range between 0 and 1, with lower values indicating higher performance, with the exception of average precision (Madjarov et al. 2012; Santos et al. 2011; Heath et al. 2010; Tsoumakas et al. 2009a).

It is important to note that the above complex network metrics in the top seven MLL models are the top seven useful metrics selected by the LP-IG feature selection method employed with the NN classifier in the seven different multilabel PTMs to predict multiple RCV classes. The ability of each MLL model to predict the levels of all four RCV classes is dependent on all the respective complex network metrics. It is difficult to interpret how the complex network metrics influence the severity levels of the volatility classes of requirements: for example, the prediction of RCV classes of a requirement data point using an MLL model that employs the NN classifier and LP-IG feature selection method is based on the distance of the values of the top seven complex network metrics in multidimensional space. Also, the results of this study are based on the current data, parameter settings, and classifiers of the MLL methods, and complex network metrics selected by LP-IG feature selection method. However, these seven complex metrics are found in

this paper to be the necessary contributing network properties of a requirement node, determining its volatility class metrics levels simultaneously. These properties are based on the physical component domain information flowing on requirement relationships modeled in the R-ARCPP tool.

Katz centrality is the extension of the right eigenvector centrality and measures a requirement node's importance by how much it is being pointed to by close and distant nodes. A requirement node's brokering coefficient compares its degrees with its clustering coefficient to measure its ability to connect many other nodes that would not be connected otherwise. Authority centrality of a requirement node is proportional to the sum of the hub centralities of requirement nodes that point to it. Left eigenvector centrality is proportional to the sum of the degrees of the neighbors of a node that it points to. It measures a requirement node's importance by the extent to which it points to many requirements or many requirements that themselves point to many others. Outgoing walks of length three represents the number of walks of length three along which a requirement node points to other requirements. PageRank centrality of a node is derived as a variation of Katz centrality and is proportional to its neighbors' centrality divided by their out-degree, meaning that each node pointed to by highly central nodes pointing to many others is endowed only a fraction of their centrality. Therefore, similar to Katz centrality, it measures a requirement node's importance by how much it is being pointed to. Incoming walks of length three represents the number of walks of length three that point to a requirement node from other requirements.

In simpler terms, Katz centrality, authority centrality, and PageRank centrality commonly represent how much a requirement node is being pointed to or the extent of physical domain information a requirement node receives. Left eigenvector centrality represents how much a requirement node points to others or the extent of physical domain information it transmits. Brokering coefficient represents a requirement node's ability to connect many other nodes that would not be connected otherwise or the extent to which it facilitates the transfer of physical domain information in the neighborhood. Outgoing walks of length three and incoming walks of length three represent the indirect connections of a requirement node; they represent requirement's indirect outgoing and incoming connections for transmittance and reception of physical domain information. There are other complex network metrics that represent the same or similar network properties as described in Table 7; however, they are the metrics that consistently yield the highest performance in predicting RCV classes when employed in MLL machine learning techniques.

Due to the limitation of this empirical study, a majority of the top-performing metrics are centrality metrics, with the exception of the extensions of clustering coefficient or

Table 12 Summary of results

Summary of results by multilabel learning methods

The ability of an MLL model to predict the levels of all four volatility classes is dependent on all the complex network metrics of the model. It is difficult to interpret in the selected top MLL models how each complex network metric of a model influences the output levels of RCV classes.

Based on the employed data and the parameter settings of the top seven different multilabel PTMs and NN classifier algorithm, the top seven complex network metrics selected by the LP-IG feature selection method yield the highest performance in predicting RCV classes simultaneously.

indirect outgoing/incoming connections—agreeing with results shown in Braha and Bar-Yam (2004a, b, 2007), Braha (2016). Centrality metrics are essential in complex real-world systems represented by network models and focusing on strongly correlated centrality metrics (Braha and Bar-Yam 2004b; Valente et al. 2008), which convey useful information regarding requirement change volatility. These results provide evidence that the utilization of complex networks and machine learning computational approaches can identify a set of network metrics representing different network characteristics that indicate requirement change volatility. For example, given a requirement change in a project, a designer can consider these indicator metrics and predict how other requirements will react to that change in terms of four RCV classes. Subsequent studies in this research direction will set practical engineering design guidelines and strategies necessary to develop more formalized computational reasoning tools capable of managing and predicting requirement change volatility in the design process. Table 12 summarizes these takeaways of multilabel learning machine learning techniques, while Table 13 summarizes the answer for RQ 3. It is important to note that many of the solutions observed here are the classical AI black box problem (Castelvecchi 2016; Bathaee 2018), where a solution is realized through the analysis, but the authors have yet to recognize how each metric influences the output.

5 Conclusions, recommendations, and future work

This research measures RCV classes based on how requirements behave in response during every instance of change: (1) multiplier, (2) absorber, (3) transmitter, and (4) robust in terms of their respective metric values: multiplicity,

absorbance, transmittance, and robustness. The volatility class metrics of each requirement are determined from requirement change data of industrial case studies and requirement relationships of the network of the R-ARCPP tool. The multiple computational models utilizing widely used MLL methods, found in both problem transformation and algorithm adaptation methods from the literature, are selected to cover diverse types of methods to comprehensively explore their computational capabilities.

The MLL methods can predict the multiple levels of all four volatility classes simultaneously and therefore, there is only one model by each method. The models are dependent on the employed data and the combinations and parameter settings of the FS methods, PTMs, and base classifiers. For example, the base classifier employed in the top MLL models, the NN classifier is dependent on the choice of distance functions and k values, and limited by its requirement for large storage space. It is also important to note that these models are limited in interpretability and their ability to extrapolate outside the training data. Due to the increment of the number of employed complex network metrics based on the LP-IG FS method, the top seven complex network metrics in the top MLL models represent different network properties of a requirement node contributing to the models' predictive ability. The metrics that represent the extent of physical domain information a requirement node receives (Katz centrality, authority centrality, and PageRank centrality), the metric that represents the extent of physical domain information it transmits (left eigenvector centrality), the metric that represents the extent of it to facilitate the transfer of physical domain information in the neighborhood (brokering coefficient), and the metrics that represent indirect outgoing and incoming connections for transmittance and receipt of physical domain information (outgoing walks of length three and incoming walks of length three) contribute to the ability of the models to predict simultaneously RCV classes.

Table 13 Summary for RQ3

RQ 3	Can machine learning techniques be employed to predict RCV using complex network metrics?
Answer	Machine learning techniques, specifically, multilabel learning methods, can be employed to predict RCV using complex network networks

As mentioned in Sect. 4, a majority of the top-performing centrality metrics under the limited empirical scope of this paper is consistent with literature findings (Braha and Bar-Yam 2004a, b, 2007; Braha 2016). The results suggest that focusing on the centrality metrics will likely allow for communicating useful information regarding requirement change volatility. Given this premise, after further studies and validations, practical engineering guidelines and strategies with a combination of the top-performing complex network metrics, centrality metrics, and machine learning algorithms capable of characterizing requirement change volatility can be formalized. Consequently, computational software tools employing such guidelines and strategies can assist designers in the early phase of the design process in implementing engineering changes once the requirements are defined.

Based on the results of the analysis, several recommendations are provided that have an immediate impact on design practice—specifically how requirements are elicited and managed. By introducing RCV to stakeholders (designers, engineering, decision-makers, etc.), it affords an opportunity to determine which requirements possess the greatest volatility. Knowing this information may impact current design practices. Contrary to most industrial practices, requirements verification should be delayed as long as possible for requirements which possess high RCV values. Because of their high level of connection (relationships) with other requirements, there is a high likelihood that changes to other requirements of the system will result. As such, additional costs are necessary when re-verifying requirements which have already been verified. Thus, it is best practice to avoid verifying requirements which have a likelihood of changing downstream as it will result in higher costs and rework.

During analysis, many of the requirements were found to be coupled (a requirement that covers several components). While coupling is unavoidable at times (for instance, in system-level requirements), it is recommended that requirements are decoupled as they should pertain to only specific components. By doing so, the network model becomes more modular as requirements are grouped closely together and few requirements spanning multiple modules. Highly modular networks result in a lower likelihood for change to propagate to other requirements outside the cluster of the networks. Moreover, requirements written in this manner are easier to evaluate for change. While this may result in a higher number of overall requirements, it prevents the stakeholder from coupling requirements unnecessarily.

Due to the employed approach's ability to select the subsets of complex network metrics and its overall satisfactory performance, the selected top MLL models are concluded to be the generalized candidate models that can be further explored in the future research. In doing so, it is recommended to address the limitations observed in this research.

The first limitation is the amount of data used for training. Recall, this study has a sample size of 575 total requirements with 16 change propagation instances. Although the MLL models perform well with the case studies employed, there is a need for more validations for more statistically more significant results. Since it is essential in this type of empirical study that computational models with high prediction ability that perform superior to the rest of the developed experimental models be selected, it is important to employ as many change instances and requirement data points as possible. Therefore, it is suggested to collect more requirement change data through industrial partnerships. This will both help to distinguish superior models and allow for their validations against heterogeneous projects. The second limitation is the total number of possible combinations of algorithm-parameter settings to explore in the MLL models. The possible parameter combinations of the MLL models in employed feature selection methods, the number of complex networks produced by the feature selection methods, problem transformation or algorithm adaption MLL methods, and machine learning classifiers are so immense that it is considered worth exploring them using comprehensive search optimization methods to obtain the parameter settings once data that are satisfactory in both quantity and quality are collected. It is now possible more than ever to utilize online cloud computing resources to perform such optimization with relative ease for the authors in the future research or for researchers in closely related areas if they intend to reproduce and improve the findings of this paper. Addressing these limitations in the future work will allow for the development of more robust generalized computational models to help engineers assess RCV before committing to implement a change to avoid time and monetary losses and unanticipated propagating changes.

Acknowledgements This material is based on work supported by the National Science Foundation under Grant No. 1951989. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix 1: Multilabel learning methods (description and implementation)

The *problem transformation methods (PTMs)* are the algorithm independent multilabel learning methods that transform the multilabel learning problem into one or more single-label learning problems on which a variety of single-label classifiers can be applied (Tsoumakas et al. 2007; Gibaja and Ventura 2014). Any of the single-label classification algorithms may be applied to PTMs. In this paper, four base classifiers, NB, DT with J48 algorithm, SVM with SMO algorithm, and NN with IBk algorithm using Euclidean distance function, and $k = 1$ are applied with default parameters.

Binary Methods The *Binary Relevance (BR)* method is a one-versus-all (OVA) approach which learns one binary data for each label and outputs the union of their predictions (Gibaja and Ventura 2014). BR ignores label dependence which may lead to failure to predict label combinations or ranking (Gibaja and Ventura 2014). *Classifier Chains (CC)* method is a similar extended approach to BR, but considers label dependence. Individual binary dataset for each label is linked in a chain along which any binary classifier can incorporate the predicted labels in previous datasets as additional features.

Label Combination Methods The *Label Powerset (LP)* method (Kafrawy et al. 2015) considers each unique combination of multiple labels as one class for transforming into a single multiclass learning problem with 2^L possible class values (Pakrashi et al. 2016). Although it performs well and considers label dependency, as the number of labels increases, the number of possible combinations grows exponentially with imbalanced data sets (Santos et al. 2011; Pakrashi et al. 2016). *Pruned Sets (PS)* method mitigates the complexity drawback of the LP by focusing on the most important combinations of labels by pruning data points with labels of low frequency. To compensate such information loss, it reintroduces the pruned data points along with subsets of their labels that exist more times than p by keeping the top b of subsets, or by keeping all subsets of size greater than b , after ranking the subsets by the size of examples they belong to (Tsoumakas et al. 2009b). The number of data points required for a label set to be included is set to $p = 1$, and the strategy for processing infrequent label sets is set to option B to keep all subsets greater than parameter $b = 2$ as recommended in (Kafrawy et al. 2015). *Hierarchy of Multilabel Classifiers (HOMER)* computes efficiently by dividing the multilabel dataset into smaller subsets of labels in hierarchical manner, so that it organizes all labels into a tree shaped hierarchy with smaller set of labels at each node (Madjarov et al. 2012). At each non-leaf node, a multiclass LP classifier is then applied. It uses even distribution of a

set of labels into k disjoint subsets to place similar labels together and the highest performance is reported using a balanced k means algorithm customized for HOMER (Tsoumakas et al. 2008). In HOMER, balanced clustering is used with the parameter $k = 3$ as recommended in (Kafrawy et al. 2015).

Pairwise methods The *Ranking by Pairwise Comparison (RPC)* method transforms a data set with q labels into $\frac{q(q-1)}{2}$ binary data sets, one per each pair of label, and follows an OVA approach as in BR (Gibaja and Ventura 2014). A binary classifier is built for each data set and the prediction is performed by invoking all models from which a ranking is obtained by counting votes for each label (Gibaja and Ventura 2014). *Calibrated Label Ranking (CLR)* extends RPC by introducing an additional virtual label used to separate the relevant and irrelevant ones, obtaining a consistent ranking and partitioning (Gibaja and Ventura 2014; Nair-Benrekia et al. 2015). It is assumed that the virtual label is preferred over all irrelevant labels and all relevant labels are preferred over it Madjarov et al. (2012). Binary classifiers are then applied and, at the prediction, a ranking over $q + 1$ labels is obtained through majority voting (Madjarov et al. 2012).

EMs of PTMs EMs of PTMs are referred to as ensemble methods, strictly in the sense of methods developed on top of PTMs, as they involve multiple binary models. Like PTMs, the machine learning classification algorithms may be applied to EMs of PTMs. *Ensembles of Classifier Chains (ECC)* is an ensemble multilabel learning method that uses CC as base classifiers with a random chain ordering and a random subset of data set sampled with replacement (Gibaja and Ventura 2014). Each label receives a number of votes when the predictions are summed by label and a threshold value is used to choose the most popular labels, forming the final predicted label set (Madjarov et al. 2012; Santos et al. 2011). In ECC, the number of models is set to 10 as recommended in Madjarov et al. (2012). *Ensembles of Pruned Sets (EPS)* is a method that combines pruned sets in an ensemble scheme. PS is not able to predict label sets that are not in the training data set similar to LP. To solve this issue, the results of several classifiers in an ensemble are combined (Gibaja and Ventura 2014). In the ensemble, a subset of the training set (63%) is sampled without replacement for each classifier and a PS classifier is trained (Gibaja and Ventura 2014). In EPS, the parameter sets in PS are kept the same with the number of models set to 10 and the voting threshold set to 0.5 as recommended in Kafrawy et al. (2015). *Random k-label Set (RAkEL)* is a method that takes label dependency into account and avoids the computational complexity of LP (Gibaja and Ventura 2014). It is an ensemble method of LP which draws m random subsets of labels with size k from all labels and trains an LP classifier on each of them (Madjarov et al. 2012; Kafrawy et al. 2015). The final set

of labels is determined by labels with averaged votes larger than a given threshold value (Santos et al. 2011; Kafrawy et al. 2015). 2^L possible labels are reduced to 2^k , simplifying computation (Tsoumakas et al. 2009b, 2006). The number of models is set to $\min(2 * Q, 100)$, $\min(2 * 12, 100) = 24$, and the size of the label set is set to $Q/2, \frac{12}{2} = 6$ as recommended in Madjarov et al. (2012).

The *algorithm adaptation methods* (AAMs) are the multilabel learning methods that adapt, extend, and customize existing classifiers to directly perform multilabel classification on multilabel datasets (Tsoumakas et al. 2007; Madjarov et al. 2012; Gibaja and Ventura 2014). AAMs are the extensions or customizations of the machine learning classifiers that can be taxonomized into such categories as decision trees, support vector machines, nearest neighbors, artificial neural networks, generative and probabilistic models, associative classifications, bio-inspired approaches, and ensembles (Gibaja and Ventura 2014). However, nearest-neighbor AAMs, artificial neural-network AAMs, and ensemble methods of AAMs are found to be widely deployed in many studies, and thus only these types of AAMs are employed in this paper.

Nearest-neighbor AAMs Multilabel K-Nearest Neighbor (MLKNN) is one of the most widely used binary relevance algorithms acting on labels individually, and extends the lazy learning algorithm KNN using a Bayesian approach to deal with multilabel dataset (Nair-Benrekia et al. 2015; Kafrawy et al. 2015). After determining the k nearest neighbors, instead of applying the standard k -nearest-neighbor algorithm directly, it uses the maximum a posteriori principle (MAP) to determine a label set for a new input based on prior and posterior probabilities for the frequency of each label within the k -nearest neighbors (Santos et al. 2011; Pakrashi et al. 2016). In MLKNN, the value of k is varied from one to the square root of the number of data points as recommended in Tiwari (2016), at 5 steps increment of 6 ($k = \{1, 6, 12, 18, 24\}$).

Artificial neural-network AAMs Backpropagation for Multilabel Learning (BPMLL) is the most widely used neural-network AAMs and is an adaptation of the popular feed-forward backpropagation (BP) algorithm (Gibaja and Ventura 2014), with the error function of the backpropagation algorithm modified to obtain a new error function that takes multiple labels into account (Dimou et al. 2009; Sorower 2010). The network has one input unit per input independent variable, one output unit per label or dependent variables, and the hidden layer is fully connected to the input and output layers (Gibaja and Ventura 2014). A threshold function is then provided to determine which label should be included in the output for a new input data (Heath et al. 2010). In BPMLL, as recommended in Dimou et al. (2009), the number of hidden neurons is set to 20% of the number

of complex network metrics inputs with other parameters kept constant.

EMS of AAMs EMs of AAMs are the ensembles, in the sense of ensemble learning (EL), base learners of which themselves are the algorithms of AAMs. Each base learner makes multilabel prediction, and then, the predictions are combined using a voting scheme such as probability distribution voting (Madjarov et al. 2012). *AdaBoost.MH* and *AdaBoost.MR* are boosting based algorithms designed for text categorization (Gibaja and Ventura 2014). The purpose of these algorithms is to identify the accurate learner by combining many base learners. AdaBoost.MH tried to minimize the number of misclassified labels for which it maintained a set of weights over the training data points and over the labels (Gibaja and Ventura 2014). After each round of training, the training data points and labels that are harder to classify get higher weights incrementally, while those that are easier get lower. It mapped the original multilabel problem into a binary problem and solves it using the traditional AdaBoost algorithm with one-level decision trees as base learners (Gibaja and Ventura 2014). Conversely, AdaBoost.MR tried to minimize the number of misordered labels, so that relevant labels would rank above irrelevant ones (Gibaja and Ventura 2014). AdaBoost.MH runs with default settings in *MULAN*.

As recommended in Spolaôr et al. (2013) using four feature selection filter methods, the multilabel dataset is first transformed using two PTMs: BR method that transforms the data into single-label datasets, one for each individual label without accounting for dependence within labels; and LP method that directly transforms the data into one single-label dataset by taking each combination of labels as a distinct label value accounting for label dependence. After BR transformation, the contribution of each complex network metric input is measured and averaged across labels using two feature selection filter methods, Relief (RLF) and Information Gain (IG). A similar procedure is performed after LP transformation, but without averaging, since LP transformed data use the combination of labels as distinct values. This results in four feature selection methods for MLL analyses: BR-RF, BR-IG, LP-RF, and LP-IG. Using these four feature rankings, the number of complex network metrics is varied based on decreasing order of scores starting from the top 7 metrics with an increment of 7, with the last step including all 49 metrics.

References

- Abdallah Z, El Zaart A, Oueidat M (2015) Experimental analysis and comparison of multilabel problem transformation methods for multimedia domain. In: Appl. Res. Comput. Sci. Eng. (ICAR), 2015 Int. Conf., 2015, pp 1–8

- Acar B, Benedetto-Neto H, Wright I (1998) Design change: problem or opportunity. In: Eng. Des. Conf., 1998, pp 445–454
- Barrat A, Barthélemy M, Pastor-Satorras R, Vespignani A (2004) The architecture of complex weighted networks. *Proc. Natl. Acad. Sci. USA* 101:3747–3752. <https://doi.org/10.1073/pnas.0400087101>
- Bathae Y (2018) The artificial intelligence black box and the failure of intent and causation. *Harv J Law Technol* 31:889–938
- Bloechl F, Theis FJ, Vega-Redondo F, Fisher EO (2010) Which sectors of a modern economy are most central? In: CESifo Work. Pap. Ser., pp 1–13
- Bollobás B (2001). Random Graphs. <https://doi.org/10.1214/aoms/1177706098>
- Bonacich P, Lloyd P (2001) Eigenvector-like measures of centrality for asymmetric relations. *Soc Netw* 23:191–201. [https://doi.org/10.1016/S0378-8733\(01\)00038-7](https://doi.org/10.1016/S0378-8733(01)00038-7)
- Braha D (2016) The complexity of design networks: structure and dynamics. *Exp Des Res Approach Perspect Appl* 20(16):129–152. https://doi.org/10.1007/978-3-319-33781-4_8
- Braha D, Bar-Yam Y (2004a) Topology of large-scale engineering problem-solving networks. *Phys Rev E* 69:7. <https://doi.org/10.1103/PhysRevE.69.016113>
- Braha D, Bar-Yam Y (2004b) Information flow structure in large-scale product development organizational networks. *J Inf Technol* 19:244–253. <https://doi.org/10.1057/palgrave.jit.2000030>
- Braha D, Bar-Yam Y (2007) The statistical mechanics of complex product development: empirical and analytical results. *Manag Sci* 53:1127–1145. <https://doi.org/10.1287/mnsc.1060.0617>
- Castelvecchi D (2016) Can we open the black box of AI? *Nature* 538:20–23. <https://doi.org/10.1038/538020a>
- Cheng H, Chu X (2012) A network-based assessment approach for change impacts on complex product. *J Intell Manuf* 23:1419–1431. <https://doi.org/10.1007/s10845-010-0454-8>
- Clark K, Fujimoto T (1991) Product development and performance: strategy, organisation and management in the world auto industries, Boston, 1991
- Clarkson PJ, Simons C, Eckert C (2004) Predicting change propagation in complex design. *J Mech Des* 126:788–797. <https://doi.org/10.1115/1.1765117>
- Clauset A, Newman M, Moore C (2004) Finding community structure in very large networks. *Phys Rev E* 70:10. <https://doi.org/10.1103/PhysRevE.70.066111>
- Colombo EF, Cascini G, De Weck OL (2015) Impact of architecture types and degree of modularity on change propagation indices. In: *Proc. Int. Conf. Eng. Des. ICED*, Politecnico Di Milano, Italy, 2015
- Costa LDF, Rodrigues FA, Travieso G, Boas PRV (2007) Characterization of complex networks: a survey of measurements. *Adv Phys* 56:167–242. <https://doi.org/10.1080/00018730601170527>
- Dangalchev C (2006) Residual closeness in networks. *Phys A Stat Mech Its Appl* 365:556–564. <https://doi.org/10.1016/j.physa.2005.12.020>
- Dankelmann P, Goddard W, Henning MA, Swart HC (1999) Generalized eccentricity, radius and diameter in graphs. *Networks* 34:312–319
- Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7:1–30. <https://doi.org/10.1016/j.jecp.2010.03.005>
- Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol Comput* 1:3–18. <https://doi.org/10.1016/j.swevo.2011.02.002>
- Dimou A, Tsoumakas G, Mezaris V, Kompatsiaris I, Vlahavas I (2009) An empirical study of multi-label learning methods for video annotation. In: *Seventh International Workshop on Content-Based Multimedia Indexing*. Chania, pp. 19–24. <https://doi.org/10.1109/CBML.2009.37>
- Eckert C, Clarkson PJ, Zanker W (2004) Change and customisation in complex engineering domains. *Res Eng Des* 15:1–21. <https://doi.org/10.1007/s00163-003-0031-7>
- El Kafrawy P, Mausad A, Esmail H (2015) Experimental comparison of methods for multi-label classification in different application domains. *Int J Comput Appl* 114:0975–8887
- Ernst NA, Mylopoulos J, Wang Y (2009) Requirements evolution and what (research) to do about it. *Lect Notes Bus Inf Process LNBP* 14:186–214. https://doi.org/10.1007/978-3-540-92966-6_11
- Estrada E, Rodríguez-Velázquez JA (2005) Subgraph centrality in complex networks. *Phys Rev E Stat Nonlinear Soft Matter Phys* 71:29. <https://doi.org/10.1103/PhysRevE.71.056103>
- Fei G (2011) A methodology for engineering design change analysis using system modelling and knowledge management technologies. *Int J Comput Integr Manuf* 24:6:535–551
- Freeman LC (1978) Centrality in social networks conceptual clarification. *Soc Netw* 1:215–239. [https://doi.org/10.1016/0378-8733\(78\)90021-7](https://doi.org/10.1016/0378-8733(78)90021-7)
- Fricke E, Gebhard B, Negele H, Igenbergs E (2000) Coping with Changes: causes, findings and strategies. *Syst Eng* 3:169–179
- Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J Am Stat Assoc* 32:675–701. <https://doi.org/10.1080/01621459.1937.10503522>
- Friedman M (1940) A comparison of alternative tests of significance for the problem of $\$m\$ rankings. *Ann Math Stat* 11:86–92. <https://doi.org/10.1214/aoms/1177731944>$
- García S, Fernández A, Luengo J, Herrera F (2010) Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf Sci (Ny)* 180:2044–2064. <https://doi.org/10.1016/j.ins.2009.12.010>
- Gibaja E, Ventura S (2014) Multilabel learning: a review of the state of the art and ongoing research. *WIREs Data Mining Knowl Discov* 4:411–444. <https://doi.org/10.1002/widm.1139>
- Giffin M, de Weck O, Bounova G, Keller R, Eckert C, Clarkson PJ (2009) Change propagation analysis in complex technical systems. *J Mech Des* 131:081001. <https://doi.org/10.1115/1.3149847>
- Guimerà R, Amaral LAN (2005) Cartography of complex networks: modules and universal roles. *J Stat Mech Theory Exp* 2005:P02001. <https://doi.org/10.1088/1742-5468/2005/02/P02001>
- Hamraz B, Caldwell NHM, Wynn DC, Clarkson PJ, Hamraz B, Caldwell NHM, Wynn DC, Clarkson PJ (2013a) Requirements-based development of an improved engineering change management method. *J Eng D* 24:765–793. <https://doi.org/10.1080/09544828.2013.834039>
- Hamraz B, Caldwell NHM, Clarkson PJ (2013b) A holistic categorization framework for literature on engineering change management. *Syst Eng* 16:473–505
- Heath D, Zitzelberger A, Giraud-Carrier CG (2010) A multiple domain comparison of multi-label classification methods In: *Work. Notes 2nd Int. Work. Learn. from multi-label data ICML/COLT*, 2010, pp 21–28
- Hein H, Morkos B, Sen C (2017a) Utilizing node interference method and complex network centrality metrics to explore requirement change propagation. In: *Vol. 1 37th Comput. Inf. Eng. Conf.*, 2017, p V001T02A081. <https://doi.org/10.1115/DETC2017-67930>
- Hein PH, Voris N, Morkos B (2017b) Predicting requirement change propagation through investigation of physical and functional domains. <https://doi.org/10.1007/s00163-017-0271-6>
- Honey CJ, Honey CJ, Kotter R, Kotter R, Breakspear M, Breakspear M, Sporns O, Sporns O (2007) Network structure of cerebral cortex

- shapes functional connectivity on multiple time scales. PNAS 104:10240–10245. <https://doi.org/10.1073/pnas.0701519104>
- Htet Hein P, Morkos B, Menon V (2015) Exploring requirement change propagation through the physical and functional domain. In: Proc. ASME 2015 Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf. IDETC/CIE 2015, 2015, pp DETC2015–47746. DETC2015-47746
- Huang GQ, Yee WY, Mak KL (2003) Current Practice of Engineering Change Management in Hong Kong Manufacturing Industries. J Mater Process Technol 139:481–487
- Hull E, Jackson K, Dick J (2005) Requirements Engineering, 2nd edn. Springer NY, New York
- Hwang W, Cho Y, Zhang A, Remanathan M (2006) Bridging centrality : identifying bridging nodes in scale-free networks. In Kdd., pp 20–23
- Kantarci B, Labatut V (2013) Classification of complex networks based on topological properties. In Cloud Green Comput. (CGC), (2013)
- Kavitha CR, Mahalekshmi T (2016) A comparison of multi-label classification methods using MEKA on benchmark datasets. Int J Res Eng Technol 05:330–335. <https://doi.org/10.15623/ijret.2016.0509050>
- Kobayashi A, Maekawa M (2001) Need-based requirement change management. In: IEEE Int. Conf. Work. Eng. Comput. Based Syst., 2001, pp 171–178
- Konganti K, Wang G, Yang E, Cai JJ (2013) SBEToolbox: a matlab toolbox for biological network analysis. Evol Bioinform Online 9:355–362. <https://doi.org/10.4137/EBO.S12012>
- Kreimeyer M, Lindemann U (2011) The foundations of complexity metrics. <https://doi.org/10.1007/978-3-642-20963-5>
- Kulk GP, Verhoef C (2008) Quantifying requirements volatility effects. Sci Comput Program. <https://doi.org/10.1016/j.scico.2008.04.003>
- Langer S, Maier A, Wilberg J, Munich T, Lindemann U (2012) Exploring the differences between average and critical engineering changes: Survey Results from Denmark, (2012)
- Latora V, Marchiori M (2001) Efficient behavior of small-world networks. Phys Rev Lett 87:198701. <https://doi.org/10.1103/PhysRevLett.87.198701>
- Latora V, Marchiori M (2007) A measure of centrality based on network efficiency. New J Phys 9:7. <https://doi.org/10.1088/1367-2630/9/6/188>
- Leicht EA, Newman MEJ (2008) Community structure in directed networks. Phys Rev Lett 100:1–4. <https://doi.org/10.1103/PhysRevLett.100.118703>
- Leng S, Wang L, Chen G, Tang D (2016) Engineering change information propagation in aviation industrial manufacturing execution processes. Int J Adv Manuf Technol 83:575–585
- Madjarov G, Kocov D, Gjorgjevikj D, Deroski S (2012) An extensive experimental comparison of methods for multi-label learning. Pattern Recogn 45:3084–3104. <https://doi.org/10.1016/j.patcog.2012.03.004>
- Mcintosh K (1995) Engineering Data Management: A Guide to Successful Implementation. McGraw-Hill, London
- Morkos JD (2010) Summers, Implementing design tools in capstone design projects: requirements elicitation through use of personas. In: 2010 Capstone Des. Conf., 2010, p 50
- Morkos B (2012) Computational representation and reasoning support for requirements change management in complex system design, Ph.D. Diss. Clemson Univ
- Menon V (2015) Application of complex network metrics to support computational reasoning of requirement change propagation in complex system design, Florida Institute of Technology, 2015
- Morkos B, Summers JB (2009) Elicitation and development of requirements through integrated methods. In: Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf., ASME, San Diego, CA, 2009
- Morkos B, Palmer G, Summers JD (2010) A study of designer familiarity with product and user during requirement elicitation. Int J Comput Aided Eng Technol 5:139–158
- Morkos B, Shankar P, Summers JD (2012) Predicting requirement change propagation, using higher order design structure matrices: an industry case study. J Eng Des 23:905–926
- Nair-Benrekia N-Y, Kuntz P, Meyer F (2015) Selecting a multi-label classification method for an interactive system. In: Lausen B, Krolak-Schwerdt S, Böhrer M (eds) Data Sci. Learn. by Latent Struct. Knowl. Discov. Springer Berlin Heidelberg, Berlin, pp 157–167. https://doi.org/10.1007/978-3-662-44983-7_14
- Newman MEJ (2003) A Measure of betweenness based on random walks. ArXiv:Cond-Mat, pp 1–15
- Newman M (2006) Modularity and community structure in networks. Proc Natl Acad 103:8577–8582. <https://doi.org/10.1073/pnas.0601602103>
- Newman M (2010) Networks: an introduction. <https://doi.org/10.1093/acprof:oso/9780199206650.001.0001>
- Pakrashi A, Greene D, Mac Namee B (2016) Benchmarking multi-label classification algorithms. In: 24th Irish Conf. Artif. Intell. Cogn. Sci., CEUR Workshop Proceedings, 2016
- Pasqual MC, De Weck OL (2011) Multilayer network model for analysis and management of change propagation. In: Int. Conf. Eng. Des. ICED, 2011, pp 1–13
- Pikosz P, Malmqvist JV (1998) A Comparative Study of Engineering Change Management in Three Swedish Engineering Companies.
- Rouibah K, Kevin CR (2003) Change management in concurrent engineering from parameter perspective. Comput Ind 50:15–34
- Rubinov M, Sporns O (2010) Complex network measures of brain connectivity: uses and interpretations. Neuroimage 52:1059–1069. <https://doi.org/10.1016/j.neuroimage.2009.10.003>
- Rucker G, Rucker C (2000) Walk counts, labyrinthicity, and complexity of acyclic and cyclic graphs and molecules. J Chem Inf Comput Sci 40:99–106
- Salkind DNJJ (2006) Encyclopedia of measurement and statistics. SAGE Publications Inc, Thousand Oaks
- Salkind NJ (2010) Encyclopedia of research design. SAGE Publications Inc, Thousand Oaks
- Santos AM, Canuto AMP, Neto AF (2011) A comparative analysis of classification methods to multi-label tasks in different application domains. Int J Comput Inf Syst Ind Manag Appl 3:218–227
- Shaffer JP (1986) Modified sequentially rejective multiple test procedures. J Am Stat Assoc 81:826–831. <https://doi.org/10.1080/01621459.1986.10478341>
- Shankar P, Morkos B, Summers JD (2012) Reasons for change propagation: a case study in an automotive OEM. Res Eng Des 23:291–303. <https://doi.org/10.1007/s00163-012-0132-2>
- Sorower M (2010) A literature survey on algorithms for multi-label learning. Oregon State Univ Corvallis 18:1–25
- Spolaôr N, Cherman EA, Monard MC, Lee HD (2013) A comparison of multi-label feature selection methods using the problem transformation approach. Electron Notes Theor Comput Sci 292:135–151. <https://doi.org/10.1016/j.entcs.2013.02.010>
- Stark GE, Oman P, Skillicorn A, Ameen A (1998) An examination of the effects of requirements changes on software maintenance releases. J Def Softw Eng 11:16. [https://doi.org/10.1002/\(sici\)1096-908x\(199909/10\)11:5%3c293::aid-smr198%3e3.0.co;2-r](https://doi.org/10.1002/(sici)1096-908x(199909/10)11:5%3c293::aid-smr198%3e3.0.co;2-r)
- Suh E, DeWeck O, Change D (2007) Flexible product platforms: framework and case study. Res Eng Des 18:67–89. <https://doi.org/10.1007/s00163-007-0032-z>
- Tawiah CA, Sheng VS (2013) Empirical comparison of multi-label classification algorithms. In: AAAI Conf. Artif. Intell., 2013, pp 1645–1646
- Tiwari AK (2016) Prediction of G-protein coupled receptors and their subfamilies by incorporating various sequence features into

- Chou's general PseAAC. *Comput Methods Programs Biomed* 134:197–213. <https://doi.org/10.1016/J.CMPB.2016.07.004>
- Toutanova K, Klein D, Manning C, Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency newtwork. In: *Proc. HLT-NAACL*, 2003, pp 252–259
- Tsoumakas G, Katakis I, Vlahavas I (2006) A review of multi-label classification methods. In: *Proc. 2nd ADBIS Work. Data Min. Knowl. Discov. (ADMKD 2006)*, pp 99–109
- Tsoumakas G, Katakis I, Overview A (2007) Multi-label classification: an overview. *Int J Data Warehs Min* 3:1–13. <https://doi.org/10.1109/ICWAPR.2007.4421677>
- Tsoumakas G, Katakis I, Vlahavas I (2008) Effective and efficient multilabel classification in domains with large number of labels. In: *Proc. ECML/PKDD 2008 Work. Min. Multidimens. Data*, pp 30–44
- Tsoumakas G, Katakis I, Vlahavas I (2009a) Mining multi-label data. In: *Data Min. Knowl. Discov. Handb. Springer*, pp 667–685
- Tsoumakas G, Zhang M-L, Zhou Z-H (2009b) Learning from multi-label data (Tutorial). In: *Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases, Bled, Slovenia*
- Tsoumakas G, Spyromitros-Xioufis E, Vilcek J, Vlahavas I (2011) MULAN: A Java library for multi-label learning. *J Mach Learn Res* 12:2411–2414
- Valente TW, Coronges K, Lakon C, Costenbader E (2008) How correlated are network centrality measures? *Connect (Tor)* 28:16–26. <https://doi.org/10.1016/j.bbi.2008.05.010>
- Wang R, Huang R, Qu B (2014) Network-based analysis of software change propagation. *Sci World J* 2014:1–10
- Watts F (1984) Engineering changes: a case study. *Prod Invent Manag* 25:55–62
- Wright IC (1997) A review of research into engineering change management: implications for product design. *Des Stud* 18:33–42
- Zhou Z-H (2015) Ensemble learning, In: *Encyclopedia Biometrics, Li SZ and Jian A, Eds. New York, NY, USA: Springer, 2015*, pp. 411–416.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.