Extracting Relations Between Organizational Patterns Using Association Mining

Shakirullah Waseeb
Software Engineering Department, Faculty of Computer Science, Nangarhar University
Afghanistan

Waheedullah Sulaiman Khail
0000-0003-1494-2499 Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava
Slovakia

Haji Gul Wahaj
Database and Information System Department, Faculty of Computer Science, Nangarhar University
Afghanistan

Valentino Vranić
0000-0001-9044-4593 Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava
Slovakia

ABSTRACT

Patterns are powerful when used in combinations. Identifying relationships between patterns is challenging. The existing approaches and pattern formats reflect the relationships with other patterns in a very informal and traditional way. We are proposing an automatic approach which discover such relationships from the patterns descriptive text using text mining and natural language processing techniques. In this work, we demonstrate how it contributes in inference of relationships and its strength among patterns.

CCS CONCEPTS

• **Software and its engineering → Agile software development**; **Collaboration in software development**; • **Information systems → Association rules**.

KEYWORDS

patterns, pattern relationship, organizational patterns

ACM Reference Format:
Shakirullah Waseeb, Waheedullah Sulaiman Khail, Haji Gul Wahaj, and Valentino Vranić. 2020. Extracting Relations Between Organizational Patterns Using Association Mining. In *European Conference on Pattern Languages of Programs 2020 (EuroPLoP '20), July 1–4, 2020, Virtual Event, Germany*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3424771.3424817

1 INTRODUCTION

Patterns exist from late 70's [2, 3]. Organizational patterns of agile software development were published in 2004 by Coplien and Harrison. These are linked documents describing proven strategies for those problems that frequently occurs in a certain context [9]. They are used to build an organization from start or solve an organizational issue within an organization [9]. Patterns are considered a problem solution pair. A pattern then describes how a solution solves a specific problem [6]. The pattern solution has to be described in such a way that the user of the pattern can decide if this solution has created an added value. Each solution has syntax. This syntax is a description that shows where the solution can fit in a larger, more comprehensive design and which other solution can improve this design. This is where the relationships between a solution and other solutions are created [6]. For example, a larger solution might be the *Size The Organization* pattern [9] where the organization will build a start up team. Growing the organization graciously is part of building the team. One way to grow the team is through the *Phasing It In* pattern [9]. Another way is through the *Apprentice Ship* pattern [9].

According to Coplien and Harrison [9], a pattern language is a language that comprises patterns and the rules to put those patterns together in meaningful ways creating pattern sequences. Patterns in a pattern language can be combined in different ways like words in a sentence. In order to make a correct sentence, one needs to follow specific rules. Not all combinations of words will result in meaningful sentences. A pattern language tells how to build a whole, a system. Although isolated application of one or several patterns is not uncommon in practice, building a whole system with patterns requires understanding the pattern language. Organizational patterns can be applied to create new organizations from scratch or they can be applied to existing organizations [9]. Organizational patterns play a crucial role in agile software development [13, 21].

Patterns are not a new phenomenon, but there are still difficulties in selecting the appropriate pattern for a particular existing problem [7, 11]. To achieve a better result and fully utilize an appropriate organizational pattern, we must thoroughly understand not only the pattern itself, but also the pattern language it belongs to [2, 16, 19]. Furthermore, pattern languages in general do not always reveal every reliable connection with other patterns [10]. Most of the authors refer to other related patterns in the pattern description. However there are many hidden relationships between patterns in the pattern description. These hidden relationships can

EuroPLoP '20, July 1–4, 2020, Virtual Event, Germany
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7769-0/20/07…$15.00
https://doi.org/10.1145/3424771.3424817

be described by area's of interest. To have a better understanding, organizational patterns and their relationships are a difficult task by itself; and thus application of appropriate patterns might be challenging.

Patterns are powerful and effective when they are combined to solve a problem. Combining multiple patterns requires knowledge about the relationships they hold. Different approaches are being taken for defining and eliciting such relationships [6, 10, 18, 19]. Identifying relationships between patterns are difficult because existing formats reflect the relationships in a very informal way.

Patterns connected with each other form a network where patterns are typically applied in combination with each other [2, 3, 8, 16, 22]. These connections are used to select the right pattern for the problem in hand and navigate to other relevant patterns to establish a corresponding sequence of patterns. Pattern authors put a lot of effort in writing their patterns. However, if readers do not understand patterns or the links between them, then it will be difficult to apply them.

The relationships between patterns are defined on semantic level. Usefulness of pattern sequences solely depends on the nature of these relationships. These relationships can be extracted automatically using text mining and natural language processing techniques.

It is crucial to note that there are many patterns available and have some relationships between them. Furthermore, new patterns will be observed. For the time being, it might be clear that few patterns could be combined through some predefined pragmatics. However, there is still the need for an automatic method or algorithm to discover such relationships in patterns. Therefore, an automatic method is required to extract similarities and connection links which exist between organizational patterns. This is the method we propose in this paper. It involves text mining techniques, such as bags-of-words model, n-gram model, tf-idf, and association mining. This method extracts relationships/associations that exist between organizational patterns. The proposed method helps not just in navigating and selecting a pattern to be applied in a given context, but it also helps in identifying which pattern is best suited after a given pattern. This way, we can compose patterns meaningfully and create useful and appropriate pattern sequences.

The rest of the paper is structured as follows. Section 2 presents an overview of the state of the art in expressing relationships between patterns. In Section 3, we propose how the association rules mining can be applied to infer the relationships between organizational patterns by using n-grams as fuel for driving such relationships. Section 4 explains how we evaluated our approach. Section 5 relates our approach to the work done by others. Section 6 concludes the paper.

2 EXPRESSING RELATIONSHIPS BETWEEN PATTERNS

Patterns are organized in pattern languages, which connect patterns together. However, pattern languages do not reveal all the information on how the relevant patterns are connected to each other. Pattern languages are primarily described as being networks of patterns, which does not provide a clear and unambiguous foundation to reveal their nature [10].

Relationship between patterns are the links between them which connect them with each other. Relationships can be created by referencing a pattern within another pattern's description. The simplest way of finding these relationships is by finding references to other patterns without considering their context. These relationships point to similar patterns or to other patterns that could be applied next, thus indicating a pattern sequence. Such relationships are limited to known patterns.

Other types of relationships are by describing areas of interest of patterns in a pattern language. There are many hidden relationships between patterns in the pattern description.

Pattern languages are documented by different authors. It might be necessary to identify interdependencies and overlaps between these patterns and pattern languages as well. The selection of an appropriate solution requires sufficient expertise for both the relevant patterns and domain in which they are applied.

3 TEXT PROCESSING AND ASSOCIATION MINING

Agrawal et.al [1] introduced an efficient algorithm that generates all significant associations between the items in a database. The algorithm mines a large collection of basket data type transactions for association rules between sets of items with some minimum specified confidence [1]. We use the bag-of-words model for mining the association rules between the sets of patterns with a specified minimum confidence.

Terms distribution information are often observed considerably effective for achieving high precision [14]. We use tf-idf to weight and score the bag-of-words vocabulary for discovering relationships between organizational patterns. Relationship strength is determined from the co-occurrence of vocabulary among the patterns. The complete process is shown in Figure 1. The method of extracting such relationships is described in the following sections.

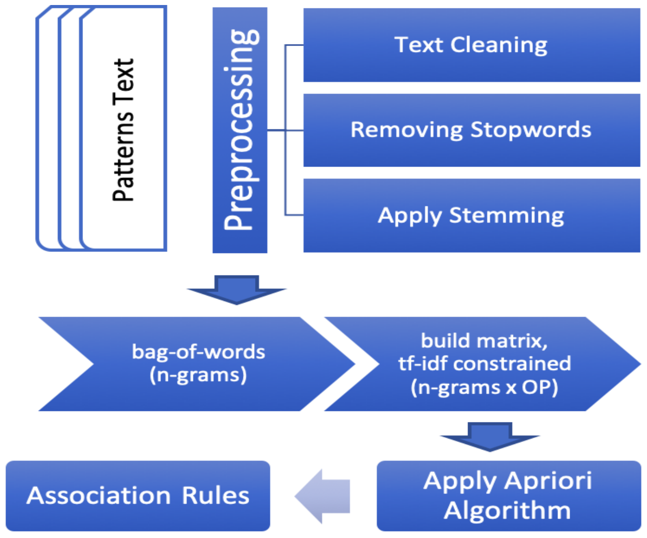

Figure 1: Process of Mining Association Rules

3.1 Preparing the Dataset

Our dataset was prepared by taking the description text of thirty organizational patterns from the Organizational Patterns website.[1] We kept the size small just for the simplicity. This approach can be applied on any catalogue of patterns with any size. Each pattern was put into a separate text document. These organizational pattern descriptions are just plain natural language text, i.e., they are unstructured data. However, the association mining algorithm requires the data to be in a relational form. Therefore, the bag-of-words (BoW) model is used for extracting features for use in association mining.

Text preprocessing is performed by cleaning the text (ignoring case, eliminating punctuation), removing stop-words, and applying stemming. Stop-words are frequent words which don't contain much information, e.g., *is*, *the*, *in*, or *are*). Stemming is reducing words to their stems (e.g., *develop* from *developing*). The scope of vocabulary is changed by using n-grams (uni-gram; vocabulary of one-word, and bi-grams; vocabulary of two-word pairs) model to allow bag-of-words to capture more meaning from the text.

Term Frequency – Inverse Document Frequency (tf-idf) is used to scale the frequency of words by how often they appear across the documents. The words with document frequency one are disregarded. By having BoW in place, we build the required matrix for mining association rules using the Apriori[1] algorithm. The columns of the matrix represent organizational patterns, while rows represent the vocabulary of the BoW, where each row indicates the existence of the vocabulary word against each organizational pattern. The existence criteria is constrained by the tf-idf parameter. The matrix sample is shown in Table 1.

3.2 Frequent Text Patterns

As we said, organizational patterns are described using text. The text is either about context, problem, solution and sometimes even consequences. Our approach is to use text mining techniques to find relationships between organizational patterns.

Apriori is an efficient algorithm for the problem of mining a large collection of basket data transactions for association rules between sets of items with some specified minimum confidence [1]. An example of such an association is the statement that 90% of transactions that purchase bread and butter also purchase milk. The antecedent of the rule consists bread and butter and the consequent consists of milk alone. The number 90% is the confidence factor for the rule. An association rule is an expression of the form $X \implies Y$, where X and Y are sets of items. The intuition of such a rule reveals that the transactions of the database which contains X tend to contain Y [1].

The same notion is used to find the association rules between organizational patterns. Thus, an association rule for organizational patterns is an expression $X \implies Y$, where X and Y are sets of patterns, which describe the fact that occurrence of words constrained by a predefined parameter tf-idf appeared in pattern X tends to appeared in pattern Y. The strength of the relationship is determined by the frequent words appearance constrained by tf-idf in relative patterns.

[1]http://www.orgpatterns.com/Organizational-Patterns-of-Agile-Software-Developm/bookoutline

3.3 Mining the Association Rules

Let $X = \{op_1, op_2, ..., op_n\}$ be a set of organizational patterns. Let D, the task relevant data, bag-of-words where each vocabulary word appears in a pattern set P such that $P \subseteq X$. A set of patterns $S \subset X$ is called pattern-set. We say that a vocabulary word appeared in pattern-set S if $S \subseteq P$. An association rule is an implication of the form $A \implies B$, where $A \subset X$, $B \subset X$, and $A \cap B \neq \emptyset$. The rule $A \implies B$ holds in set D with support s, where s is the percentage of vocabulary words in D that occur in $A \cup B$ (i.e., the union of sets A and B) This is taken to be the probability, $P(A \cup B)$:

$$support(A \implies B) = P(A \cup B) \tag{1}$$

The $A \implies B$ rule has confidence c in set D, where c is the percentage of vocabulary words in D occurs in A that also occurs in B. This is taken to be the conditional probability, $P(B|A)$. That is,

$$confidence(A \implies B) = P(B|A) = \frac{support(A \cup B)}{support(A)} \tag{2}$$

Equation (2) shows that the confidence of rule $A \implies B$ can be easily derived from the support counts of A, B, and $A \implies B$. That is, once the support counts of A, B, and $A \cup B$ are found, it is straightforward to derive the corresponding association rules $A \implies B$ and $B \implies A$ and check whether they are strong. Thus, the problem of mining association rules can be reduced to that of mining frequent pattern-sets.

In general, association rule mining can be viewed as a two-step process:

(1) Find all frequent pattern-sets. By definition, each of these pattern-sets will occur at least as frequently as a predetermined minimum support count (min-sup).
(2) Generate strong association rules from the frequent pattern-sets. By definition, these rules must satisfy the minimum support and minimum confidence.

Support and confidence are two important measures for extracting rules. Support reflects the usefulness, while confidence reflects the certainty of discovered rules. The support metric is defined for discovering/building pattern-sets. Antecedent support is the support for pattern A which is the proportion of the vocabulary words appeared in given pattern. Consequent support computes the support for consequent pattern B. Similarly, support metric computes the combined support of pattern-set $(A \cup B)$. These metrics are used for computing the confidence of a rule $(A \implies B)$, which is the probability of seeing the n-gram words appeared in the consequent also appeared in the antecedent.

Consider the association rule between *Architect Controls Product* (ACP) and *Architect Also Implements* (AAI). The bag-of-words has 1,537 vocabulary. ACP has 254 vocabulary from the bag-of-words, and the AAI has 355. The ACP and AAI both has 116 vocabulary in common. Now the support and confidence is calculated for the rule ($ACP \implies AAI$) as:

$$support(ACP) = 254/1,537 = 0.17 \tag{3}$$

$$support(AAI) = 355/1,537 = 0.23 \tag{4}$$

Pattern	('develop','team')	('develop', 'role')	('totalitarian', 'control')
ArchitectControlsProduct (ACP)	1	1	1
DeveloperControlsProcess (DCP)	1	1	1
ArchitectAlsoImplements (AAI)	1	1	1
DomainExpertiseInRoles (DEIR)	0	1	0
SizeTheOrganization (STO)	1	0	0

Table 1: A matrix sample with (0,1) values for n-gram existence against organizational patterns.

Terms	ACP	DCP	AAI	DEIR	STO
('develop','team')	1	1	1	0	1
('develop', 'role')	1	1	1	1	0
('totalitarian', 'control')	1	1	1	0	0

Table 2: The transpose matrix of Table 1.

$$support(ACP \cup AAI) = 116/1,537 = 0.08 \tag{5}$$

$$confidence(ACP \implies AAI) = 0.08/0.17 = 0.46 \tag{6}$$

The *support* of 8% for association rule in Equation (5) means that 8% vocabulary of bag-of- words under analysis show that both DCP and ACP has the possibility of having relationship. The confidence signifies the strength of a pattern/pattern set in a relationship. The *confidence* of 46% in Equation (6) means that the ACP pattern with 46% of its vocabulary claims its strength in the relationship of (ACP, AAI). Some samples of the extracted rules are listed in Table 3.

4 EVALUATION

As we said in Section 3.1, as our dataset, we used the description of thirty organizational patterns of agile software development from the Organizational Patterns website.[2], which contained a total of 127 641 words. After the data-set text cleaning, a bag-of-words with a vocabulary size of 1 537 was constructed. The method was evaluated by changing tf-idf, support, and confidence parameters. The change of tf-idf from 0.1 toward 0.3 reduced the number of association rules while the quality of the rules improved. The change in the support value result in a number of pattern combination sets: two-pattern sets, three-pattern sets, and four-patterns sets. The value of 0.04 for the support resulted in the generation of very few pattern sets: few two-pattern sets, rare three-pattern sets, and zero four-pattern sets. The confidence value resulted in a change of the number of association rules that can be observed from the the confidence value change graph for the values 0.3 to 0.5, as shown in Figure 2.

The strength of an association rule depends on the tf-idf value: the higher the tf-idf value, the stronger the association rule. Based on the evaluation, we further investigated the method results for the parameters tf-idf set to 0.2, min-support for frequent pattern-set set to 0.02, and min-confidence threshold set to 0.3. With this parameter configuration, a total of 372 association rules with sampling of pattern-set size 4 were mined. A pattern set with size 2 had 88 rules: 276 rules with size 3, and 8 rules with size 4. The min-support 0.02 discovered all possible (k-patterns-set; k=2,040) with at least 2% of support. The 0.3 min-confidence confirmed at least 40% of strength in a relationship. The value of 0.2 for the tf-idf to consider the most important words.

Figure 3 shows the resulting support and confidence for association rules mined for tf-idf set to 0.2, support set to 0.01, and confidence set to 0.2. Rules carrying higher confidence with lower support and high support and lower confidence reveal the dominance of a given pattern over other patterns, which means that the other patterns have very low influence in the association and vice versa. Low confidence and low support shows dissimilarities between patterns. Therefore, we considered 0.02 support to ensure significant similarities between patterns. Similarly, we considered 0.3 confidence to have association rules for patterns having essential strength in the relationship. Once we gained these associations, we started to observe the patterns for finding the type of association.

As depicted in Figure 4, mined organizational patterns association rules can be visualized using a network graph. The nodes represents organizational patterns, and the edges represent association rules. The color and style of an edge reflects the strength of the relationship. A solid blue edge represent stronger, dashed edge green represents normal, and red dotted represents a weaker relationship. Such a diagram reveals a way of navigating between organizational patterns. Recall that in the introduction we mentioned how the *Size The Organization*, *Phasing It In*, and *Apprenticeship* organizational patterns are related. This association can be clearly seen in Figure 4 between the nodes STO (Size The Organization), PII (Phasing It In), and A (Apprenticeship). This means that this approach not only mines associations between patterns, but also shows the navigation between them.

4.1 Qualitative Evaluation

Once we extracted the rules, we started to explore the patterns to uncover the types of the relationships between them. Based on the strength of associations visualized in Figure 4, we selected *Developer Controls Process* (DCP), *Architect Controls Product* (ACP), and *Architect Also Implements* (AAI) as patterns with stronger associations, and *Size The Organization* (STO), *Phasing It In* (PII), and *Apprenticeship* (A) as patterns with normal strength associations.

[2]http://www.orgpatterns.com/Organizational-Patterns-of-Agile-Software-Developm/bookoutline

patt(B|A)	$\sup(A \cup B)$	$\text{conf}(A \implies B)$
Architect Controls Product | *Architect Also Implements*	0.08	0. 46
Architect Controls Product | *Developer Controls Process*	0.07	0. 41
Developer Controls Process | *Architect Controls Product*	0.07	0.42

Table 3: Pattern association rules with the corresponding support and confidence.

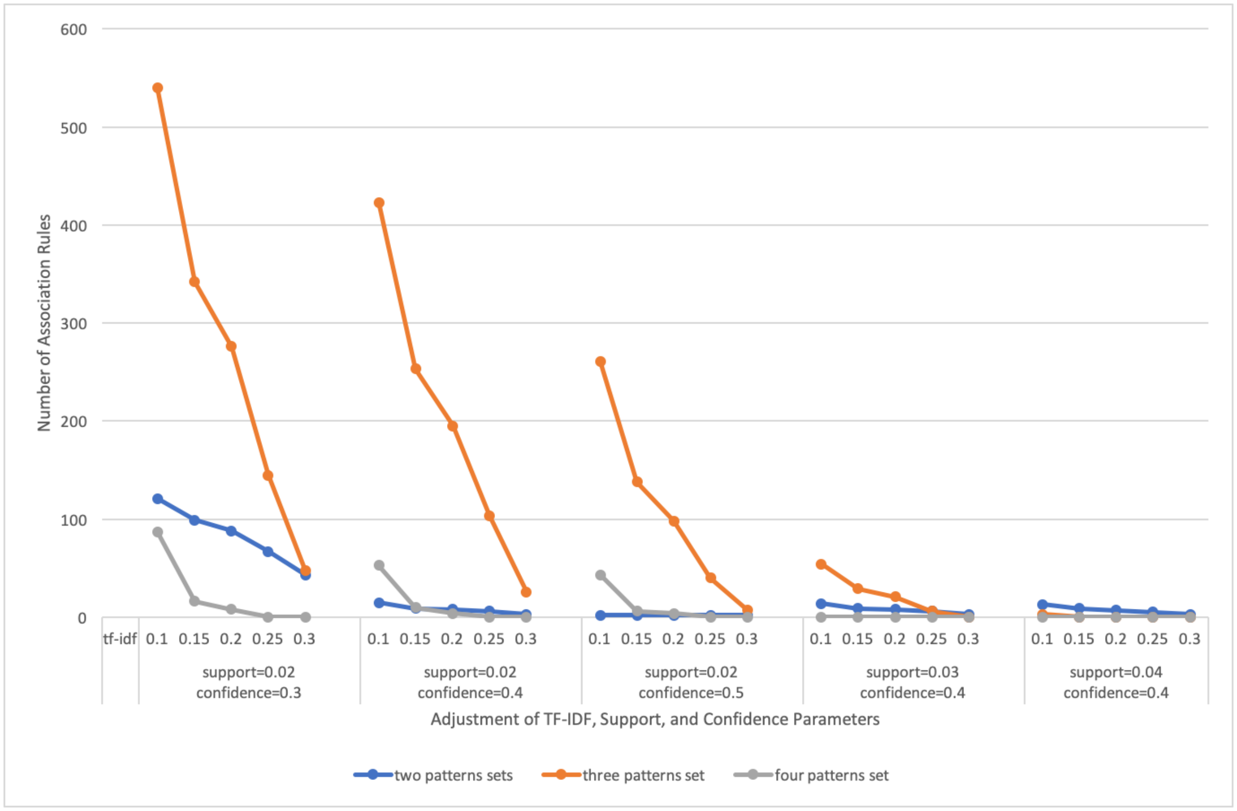

Figure 2: Confidence value change graphs.

We studied these patterns according to common n-gram words shown in Figures 5 and 6. It was observed from the sentences in which these words occur that these patterns support each other. Consider these examples for DCP, ACP, and AAI:

- *develop, team*:
  - DCP: Totalitarian control is viewed by most *development teams* as a draconian measure.
  - ACP: The architect is the principal bridge-builder between *development teams* members.
  - AAI: But even if that was possible, totalitarian control is viewed by most *development teams* as a draconian measure.
- *inform, flow*:
  - DCP: The right *information must flow* through the right roles.
  - ACP: However, the right *information must flow* through the right roles.
  - AAI: The right *information must flow* through the right roles.
- *develop, process*:
  - DCP: The developer is central to all activities of this end-to-end software *development process*.
  - ACP: Architectural control must balance developer authority… their ownership of the code *development process*.
  - AAI: The architects also learn by seeing, first-hand, the results of their decisions and designs: an important place for feedback in the *development process*.
- *control, product* and *control, process*:
  - DCP: While the developer *controls the process*, the architect *controls the product*.
  - ACP: While the developer *controls the process*, the architect *controls the product*.

The first bi-gram word (develop, team), which appears in sentences of DCP, ACP, and AAI, speaks about a development team in which different roles are established by resolving the central/-totalitarian control. The second bi-gram word (inform, flow) also reveals the importance of the information flow through the right

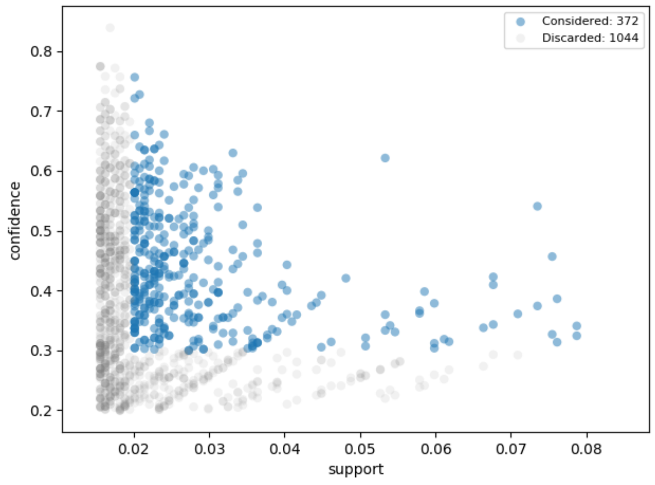

Figure 3: 1 416 rules among thirty organizational patterns.

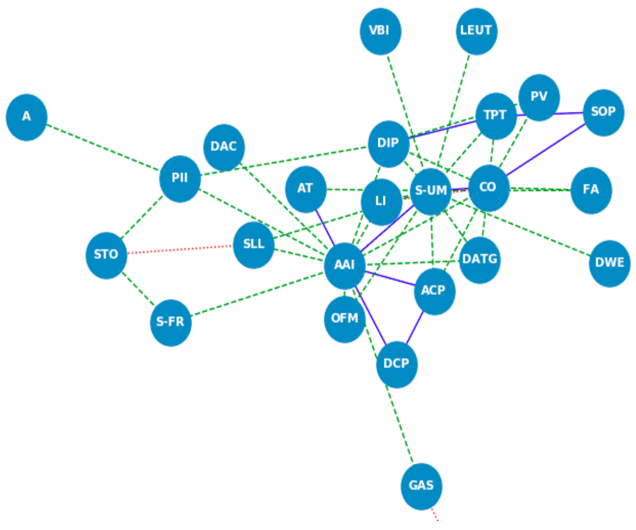

Figure 4: A network graph for minded association rules.

roles, which could be considered as a solution for the problems raised due to the central control. The last bi-gram word (control, product) provide interesting information about roles of an architect and developer.

Consider these examples for STO, PII, and A:

- *hire*:
  - STO: To decide whom to *hire* into a nascent organization, use patterns like *Domain Expertise In Roles* and *Architecture Team*.
  - PII: Growing projects must figure take into account out how to grow long-term staff: whom to *hire*, how many to *hire*, and when to *hire* them.
  - A: Turn new *hire* into experts (see DomainExpertiseInRoles) through an apprenticeship program.
- *critic, mass* :
  - STO: In addition, if an organization is too small, the team won't have a *critical mass* and productivity will suffer.
  - PII: You need enough people for a *critical mass*.
  - A: You need enough people for a *critical mass*.
- *people*:
  - STO: Adding *people* late to a project rarely helps complete that project on time and within budget.
  - PII: The right set of initial *people* (*Size The Organization*) sets the tone for the project, and it's important to hire the key *people* first.
  - A: It is better to apprentice *people* than to put *people* through a "trial by fire" that may damage the project.
- *domainexpertiseinrol*:
  - STO: To decide whom to hire into a nascent organization, use patterns like *Domain Expertise In Roles* and *Architecture Team*.
  - PII: Key project players have been hired or otherwise brought into the project and cover the necessary expertise (*Domain Expertise In Roles*), but the project needs more staff.
  - A: Turn new hires into experts (see *Domain Expertise In Roles*) through an apprenticeship program.

The exploration of the n-grams shows that all the three patterns are about how to manage the size of the organization. It leads us how to start, who to hire, and how to hire the best.

4.2 Discussion

Organizational patterns of agile software development are linked document, describing proven strategies for those problems that frequently occur in certain contexts. As patterns are solutions to problems, they should be presented and described in a way that their user can decide if the solution created by them adds value. The strength of patterns is in their combination, and this is where the relationships between them is created. These relationships are defined and visualized using different approaches [2, 3, 6, 8, 10, 16, 18, 19, 22]. Mostly, these approaches are traditional and the relationships are annotated by the user explicitly. We use an automated approach to find such relationships/associations with text mining and natural language processing techniques. However, there is still room to improve this approach in order to mine semantically sound relationships/associations.

We used n-grams without considering their context semantics. Words can have multiple meanings causing ambiguity, which is a common phenomenon in natural languages [12]. The meaning of the word in a text cannot be judged only on its own, but depends on the context the word appears within. Consider the word *status*, which in example (1) means the utterance of someone, while in example (2) it means a territory. Such words are called homonyms. Similarly, the words *talk* and *state* are synonyms, but in the context of examples (1) and (3) have to different meaning. Hence, the association rules we mined can be further improved by incorporating the word sense disambiguation techniques into our current approach. Using sentence level interpretation or discourse analysis could impart the relationship types between patterns.

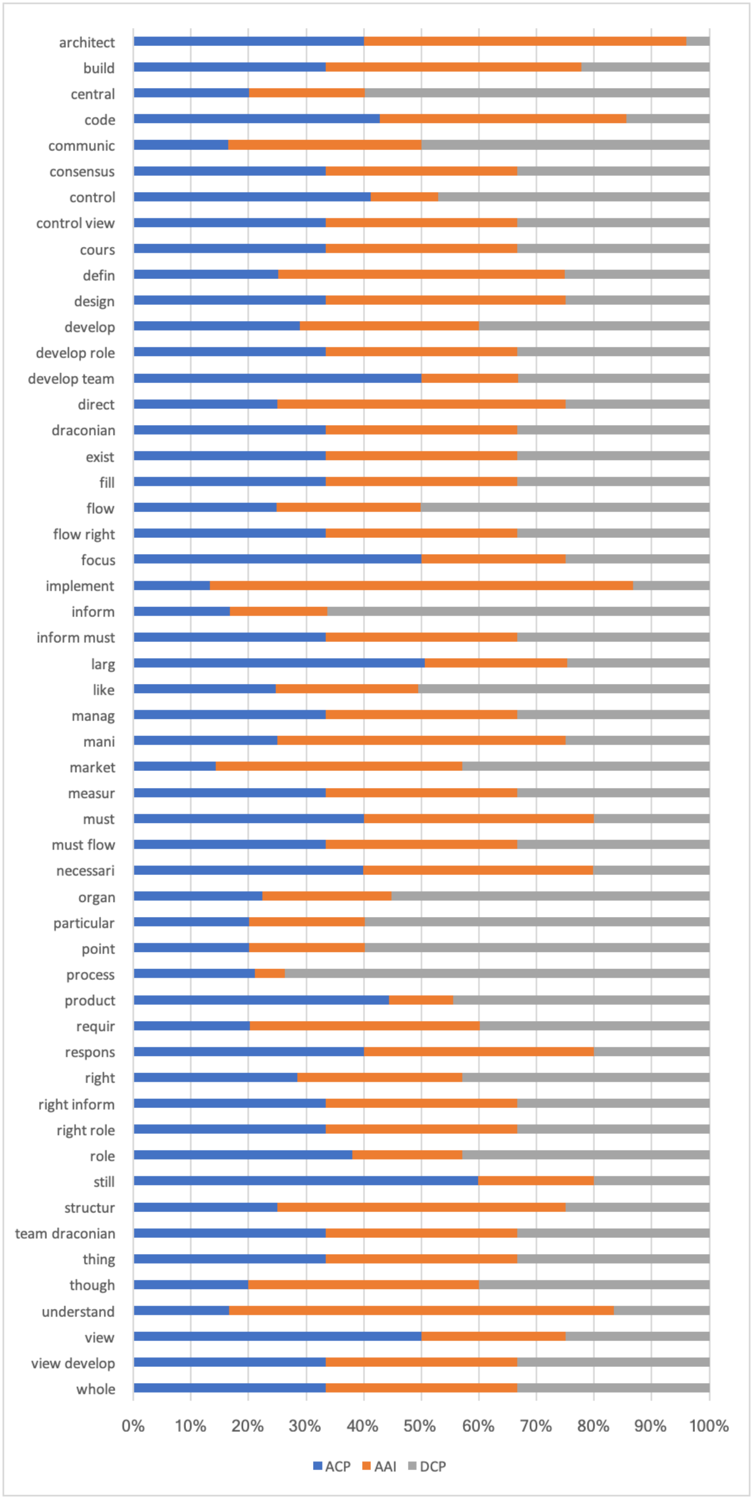

Figure 5: Stacked plot for tf-idf of common n-grams among ACP, AAI, and DCP.

(1) Note that since ConwaysLaw states that organization and architecture are isomorphic, that the architecture must follow the market.

(2) The project was spread across three states and two countries, though most of the work centered in two states.

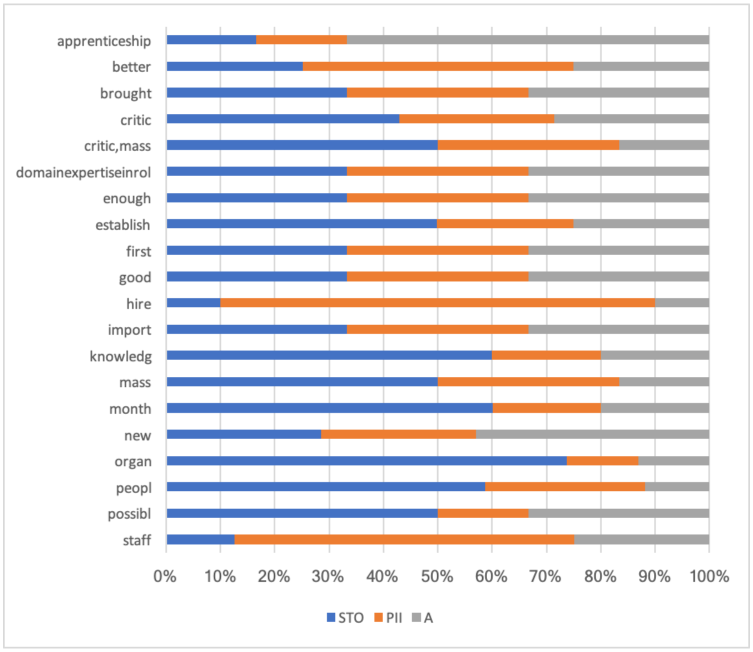

Figure 6: Stacked plot for tf-idf of common n-grams among STO, PII, and A.

(3) Yet the system must act as a system, and to do that the parts must talk to each other.

5 RELATED WORK

There exist works in the literature which helps in selection of patterns based on relationships between them. Some works used graphs to help in representing relationships between patterns hence make it easy in selection of patterns. While others used formal methods and grammars to express relationships between patterns.

Seidel in his work [16] analyzed the patterns interrelations using network analysis methods by considering the pattern language as a graph. The graph G is an ordered pair $G = (V, E)$ of vertices V and edges E. Each vertex represent a pattern and each edge is composed of relationship between two patterns. The pattern network of 45 vertex and 136 edges was built. The graph representation of a pattern language constitutes all pattern relationships in a pattern language. An overview of these relationship types and how to identify them in a pattern language has been given. The interrelations being used were: *contains*, *contained by*, and *similar*. The degree (number of patterns linked between patterns) of the pattern indicates how central or isolated a patterns is in the network.

Rouhi and Zamani [15] revised and extended the GEBNF meta modeling notation [4] and provided more abstraction and flexibility in the specification of rules. Then, they applied the revised notation to model class and sequence diagrams of UML which is the popular modeling language of design pattern solutions. Also, inspired by the notion of systematic pattern selection method of Zdun [22] and exploiting the induced functions of the UML class and sequence diagrams, they presented a formal model for a given pattern language.

Subburaj Ramasamy and co in their work [17] the impact of design pattern application have discusses how to select design patterns. In their work, they have proposed to keep in mind the Following points while selecting a pattern: scan the intent part of patterns, study other similar patterns, examine causes of redesign, and identify the relationship between patterns. However, they have not proposed any specific method which will help in applying the above mentioned steps.

Zdun proposed an approach to support the selection of patterns based on desired quality attributes and systematic design based on patterns [22]. The proposed a grammar annotated with effects on quality goals. Complex design decisions are further analyzed by using the design spaces covered by a set of related software patterns. The approach helps in finding and categorizing the appropriate software patterns systematically. However, this approach might not be applicable to solve very large design problems because the decision making on selecting a pattern variant or pattern alternative is very complex.

Sulaiman Khail [18] proposed a new pattern format in which the relationships were reflected in the pattern structure as keywords. These relationship keywords would be used to move between patterns and compose patterns.

Becker et al. [6] in their work focused on the relationships between context-patterns, which forms the syntax of a pattern language and can be derived from pattern relationship tables.

Alexander also noted that adjacent patterns should be applied as close in sequence as possible [3], but neither Alexander's theory nor language diagrams indicate how to order adjacent patterns.

Bayley and Zhu [4, 5]have proposed a formal language as well as a set of operators for composing related patterns. These operators have been used to compose the GoF design patterns [20] and sketch the pattern relationships.

The presented set of relationship and their strength in our work is more precise. Set of relationships and their confidence strength is drawn automatically, by using association rule mining approach and the basic natural language processing techniques such as stemming, stop-words, n-grams and frequent terms. The relationship set (k-patternset) is built based on the terms document frequency and inverse document frequency, minimum-support, and confidence-threshold parameters.

6 CONCLUSIONS AND FURTHER WORK

relationships forms base for patterns combinations. Therefore, to combine various organizational patterns and to create sequences of them, it is important to know how patterns are related with each other and how could be associated. In order to extract such relationships from a group of organizational patterns, we propose to use text mining techniques and algorithms. The proposed technique is an approach to discover relationships between organizational patterns, which are based on frequent vocabulary among patterns. The relationship strength is determined from the percentage of vocabulary appeared among the patterns. The proposed method will make patterns more readable and will cause to solve bigger problems. The future work will be to find relationship type of mined rules. To know the type of relationship, is it either solution of a problem or is a supportive type. We may add semantic information via NLP.

ACKNOWLEDGMENTS

We would like to thank Niels Seidel for being our shepherd and for his constructive remarks. Our sincere thanks also go to our writer's workshop group members: Michael Weiss, Eduardo Fernandez, and Rafik Tighilt.

The work reported here was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under grant No. VG 1/0759/19 the Operational Programme Integrated Infrastructure for the project Research and Development of Software Solution with the Application of Blockchain Technology in the Field of International Rail and Container Transport of Goods (ITMS code: 313022V816), co-funded by the European Regional Development Fund (ERDF).

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliundefinedski, and Arun Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*. Association for Computing Machinery, New York, NY, USA, 207–216. https://doi.org/10.1145/170035.170072

[2] Christopher Alexander. 1979. *The Timeless Way of Building.* Oxford University Press.

[3] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Joaquim Romaguera i Ramió, Max Jacobson, and Ingrid Fiksdahl-King. 1977. *A Pattern Language.* Gustavo Gili.

[4] Ian Bayley and Hong Zhu. 2008. On the Composition of Design Patterns. In *Quality Software, 2008. QSIC'08. The Eighth International Conference on.* IEEE, 27–36.

[5] Ian Bayley and Hong Zhu. 2010. Formal Specification of the Variants and Behavioural Features of Design Patterns. *Journal of Systems and Software* 83, 2 (2010), 209–221.

[6] Kristian Beckers, Stephan Fassbender, and Maritta Heisel. 2014. Deriving a Pattern Language Syntax for Context-patterns. In *In Proceedings of 19th European Conference on Pattern Languages of Programs, EuroPLoP 2014.* ACM, Irsee, Germany.

[7] Aliaksandr Birukou. 2010. A survey of existing approaches for pattern search and selection. In *Proceedings of the 15th European Conference on Pattern Languages of Programs.* 1–13.

[8] Frank Buschmann, Kelvin Henney, and Douglas Schimdt. 2007. *Pattern-Oriented Software Architecture: On Patterns and Pattern Language.* Vol. 5. Wiley.

[9] James O. Coplien and Neil B. Harrison. 2004. *Organizational Patterns of Agile Software Development.* Prentice-Hall.

[10] Michael Falkenthal, Uwe Breitenbücher, and Frank Leymann. 2018. The nature of pattern languages. *cit. on* (2018), 14.

[11] Tomáš Frťala and Valentino Vranić. 2015. Animating Organizational Patterns. In *Proceedings of 8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015, ICSE 2015 Workshop.* IEEE, Florence, Italy.

[12] Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, USA.

[13] Predrag Matkovic, Mirjana Maric, Pere Tumbas, and Marton Sakal. 2018. Traditionalisation of agile processes: Architectural aspects. *Computer Science and Information Systems* 15, 1 (2018), 79–109.

[14] Bo Pang and Lillian Lee. 2008. Using Very Simple Statistics for Review Search: An Exploration. In *COLING.*

[15] Alireza Rouhi and Bahman Zamani. 2016. Towards a formal model of patterns and pattern languages. *Information and Software Technology* 79 (2016), 1–16.

[16] Niels Seidel. 2017. Empirical evaluation methods for pattern languages: sketches, classification, and network analysis. In *Proceedings of the 22nd European Conference on Pattern Languages of Programs.* 1–24.

[17] R Subburaj, Gladman Jekese, and Chiedza Hwata. 2015. Impact of object oriented design patterns on software development. (2015).

[18] Waheedullah Sulaiman Khail and Valentino Vranić. 2019. Reflecting pattern relationships in a pattern format. In *Proceedings of the 24th European Conference on Pattern Languages of Programs.* 1–5.

[19] Waheedullah Sulaiman Khail and Valentino Vranić. 2017. Treating Pattern Sublanguages As Patterns with an Application to Organizational Patterns. In *Proceedings of the 22Nd European Conference on Pattern Languages of Programs, EuroPLoP 2017.* ACM, Irsee, Germany.

[20] John Vlissides, Richard Helm, Ralph Johnson, and Erich Gamma. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

[21] Hironori Washizaki, Suthinan Thanintranon, Masashi Kadoya, Yoshiaki Fukazawa, Takeshi Kawamura, and Joseph W. Yoder. 2014. Analyzing Software Patterns Network Obtained from Portland Pattern Repository. In *Proceedings of the 21st Conference on Pattern Languages of Programs (PLoP '14).* The Hillside Group, USA, Article 8, 6 pages. http://dl.acm.org/citation.cfm?id=2893559.2893567

[22] Uwe Zdun. 2007. Systematic Pattern Selection Using Pattern Language Grammars and Design Space Analysis. *Software: Practice and Experience* 37, 9 (2007), 983–1016.