Automatically Mapping Natural Language Requirements to Domain-Specific Process Models

Uthayasanker Thayasivam
Department of Computer Science
University of Georgia
Athens,Georgia 30605, USA

Kunal Verma, Alex Kass and Reymonrod Vasquez
Accenture Technology Labs
Accenture
San Jose, California 95113, USA

Abstract

For large scale enterprise implementations, a key problem, that has not been tackled much, is the ability to automatically map users' requirements to reference process models. We present a tool called Process Model Requirements Gap Analyzer (ProcGap), which uses a combination of natural language processing, information retrieval and semantic reasoning to automatically match and map textual requirements to industry-specific process models. We present the results of mapping requirements from an industry project to an existing process model. We compare our approach to two previously implemented approaches and show that our approach outperforms them. In a case study, we also found that a user group with ProcGap had better performance than a user group that performed the same task manually.

Introduction

The requirements gathering exercise for large enterprise software implementations, especially those based on packaged software such as SAP, is often based on gap analysis between the users' requirements and the out-of-the-box capabilities of the package. Previous research (for e.g. (Daneva 2004) and (Pnina et al. 2001)) has found that basing implementations on standard offerings reduces costs and risks as it enables leveraging past experience and reusing existing artifacts such as code and test-scripts. Our experience analyzing the requirements process in the software industry has shown that gap analysis is typically a manual task that is often time-taking and error prone because process models typically have thousands of capabilities and most projects typically have hundreds of requirements. As a result, projects often fail to build upon experiences from previous projects or reuse artifacts that would have saved them time and effort.

To enable reuse of knowledge and artifacts, a recent trend in the enterprise software industry has been the emergence of reference process or capability models. Examples of such models include IBM's Web Industry Content Packs(IBM 2011), ARIS Reference models for various industries and domains(ARIS 2011) and Accenture's Business Process Repository (ABPR 2011). These models represent the reference capabilities and processes for a domain. A sample process model snippet is shown in the left pane of Figure 1. Large scale implementors such as Accenture, are placing a large emphasis on leveraging these models in project execution. Under this approach, a central activity is to determine the mapping between the requirements requested by stakeholders and the capabilities defined in the reference model. Doing so supports a number of key analysis objectives: 1)highlighting common capabilities from the reference model not specified in requirements and flagging them as potentially missing requirements, 2) determining which portions of the reference model correspond to requirements, 3) identifying requirements that do not map to elements of the reference process model and classifying them as potentially risky requirements and 4) making any asset associated with that element available to support other analysis and design activities.

In spite of the clear benefits of mapping the requirements to reference models, there is no automated tooling support for creating this mapping. In this paper, we will present a tool called Process Model Requirements Gap Analyzer (ProcGap), which uses a combination of natural language processing (NLP), information retrieval (IR) techniques and semantic reasoning to automatically match and map textual requirements to process models. Automatically mapping natural language requirements to process models is a challenging research problem because of the following factors:

Many similar capabilities: Many of the capabilities in a process model are about similar objects. For example, the object "invoice" appears in 95/3116 capabilities in one of the process models and it is often the relationship of the object to other parts of sentence such as verb and the prepositional phrase, which make the phrase unique (*create an invoice* vs. *delete an invoice*). Unlike traditional IR techniques, the high-frequency words objects cannot be ignored or deemphasized either, since they are important for the matching.

Use of domain specific terms: Requirements-writers often use domain-specific terms such as *debit memo* to refer to something that may be referred to by a term like *invoice* in the process model.

Implicit details: When a user talks about *entering a discount code*, he/she may be implying the capability of *creating a purchase order* without actually mentioning it.

Lack of standardization of requirement syntaxes Users

may use different syntaxes to express the same intent. For example, the requirements, *System shall allow the user to create a purchase order* and *Purchase order shall be created by the user*, should both be mapped to the capability *create purchase order* in the process model.

There has been some previous work on using NLP and IR techniques for mapping requirements to other artifacts. Most of the approaches, such as (Jirapanthong and Zisman 2009) and (Zachos and Neil A. M. 2008), have used a set of rules over the output of syntactic/shallow parsing to extract relevant constituents from the requirements. (Hayes, Dekhtyar, and Sundaram 2006) used IR techniques to map low level requirements to high-level requirements. There is a also vast body of work in natural language processing for judging similarity between two sentences. (Mihalcea, Corley, and Strapparava 2006) proposes using both corpus-based similarity and lexical similarity using WordNet between words for matching. (Li et al. 2006) propose using WordNet and order of the words in the sentences for calculating sentence similarity. These approaches are focused on general-purpose similarity between sentences. There are three key differences between our approach and previous approaches: 1)we use a combination of NLP and IR techniques so that we can benefit from both types of approaches. ProcGap can handle both well written, easily parsed requirements or poorly written and complex requirements, that cannot be parsed easily, 2) a differentiator in our NLP approach is that we have developed a rule based approach that leverages dependency parsing (de Marneffe, MacCartney, and Manning 2006). This allows handling some linguistic complexity that previous approaches based on syntactic/shallow parsing could not handle and 3) another differentiator from previous work is that in addition to WordNet, we also use a semi-automatically generated semantic graph representing domain knowledge to assist with the matching.

We have evaluated our approach on requirements from a industry project, using a process model created by another group in our organization. We compared the matching approach used by ProcGap to a number of previous approaches; ProcGap outperforms all of them. We also conducted a user evaluation where users were asked to map requirements with or without ProcGap. Our results show that the users who used ProcGap outperformed the users who did not.

Overview of ProcGap Application

A screen-shot of ProcGap after the user has used the automatic mapping feature is shown in Figure 1. The pane on the left depicts some of capabilities of the ERP process model for the chemical industry. The pane on the right shows some of the user requirements. A leaf capability, with at least one requirement mapped to it, has a green indicator box in front of it, otherwise it has a red indicator box. For non-leaf capabilities, the color of the indicator box depends on the number of children that are mapped. A requirement that is mapped to at least one capability has a green "'R" in front of it. Arrows are used to depict mappings. ProcGap also allows users to manually edit or remove the mappings and view reusable assets associated with a capability.

Matching

We have designed a matching approach to deal with the challenging problem of mapping requirements to the capabilities in a process model. The matching algorithm used by ProcGap uses two distinct measures - 1) similarity between extracted Verb, Object and Prepositional-object (VOP) triples and 2) cosine similarity. A number of extracted VOP triples are shown in Table 1. We ignore the noun phrase because capabilities (for e.g., *create invoice for sales order*) typically do not have a noun phrase. To deal with the issue of non-standard syntaxes, we leverage a set of rules that operate on the output of a dependency parser (de Marneffe, MacCartney, and Manning 2006) to extract the VOP triples. We also leverage knowledge from WordNet and a domain-specific semantic graph based on the process model (discussed in next section) to find related concepts. This allows the ProcGap matching algorithm to deal with issues mentioned earlier such as implicit details in the requirements. For dealing with poorly written or long sentences that are hard to parse, we leverage cosine similarity. A high-level overview of our approach is shown in Figure 2.

Extracting VOP triples

We leveraged the Stanford Parser to extract VOP triples because it uses dependency parsing that is less susceptible to syntax of a sentence. For example, the generated dependency lists for capabilities *System creates an invoice* and *Invoice shall be created*, contain the relationship between *create* and *invoice*, using *dobj* and *nsubjpass* relationships respectively. For more details about dependency parsing, please refer to (de Marneffe, MacCartney, and Manning 2006). ProcGap uses a simple set of rules to extract the VOP triples from the dependency lists.

VOP Matching Algorithm

This section defines the algorithm for matching VOP triples of requirements and capabilities. The similarity score ($VOP(R, C)$) between a requirement(R) and a capability(C) is defined as:

$$VOP(R,C) = \max_{\substack{(< V_R, O_R, P_R > \in R, \\ < V_C, O_C, P_C > \in C)}} \left\{ \begin{array}{l} SIM(V_R, V_C) \\ \times SIM(O_R, O_C) \\ \times SIM_P(P_R, P_C) \end{array} \right\} \quad (1)$$

Where, $< V_R, O_R, P_R >$ is a VOP triple extracted from Requirement R and $< V_C, O_C, P_C >$ is a VOP triple extracted from Capability C.

$SIM(T_R, T_C)$ is the similarity score between any two terms T_R and T_C is calculated as:

$$SIM(T_R, T_C) = \max \left\{ \begin{array}{l} SIM_{str}(T_R, T_C), \\ SIM_{sem}(T_R, T_C), \\ SIM_{Lin}(T_R, T_C) \end{array} \right\} \quad (2)$$

$SIM_{str}(T_R, T_C)$ is the string similarity score between any two terms T_R and T_C and is based on exact string comparison between the stemmed (Martin F. 1980) version of the terms.

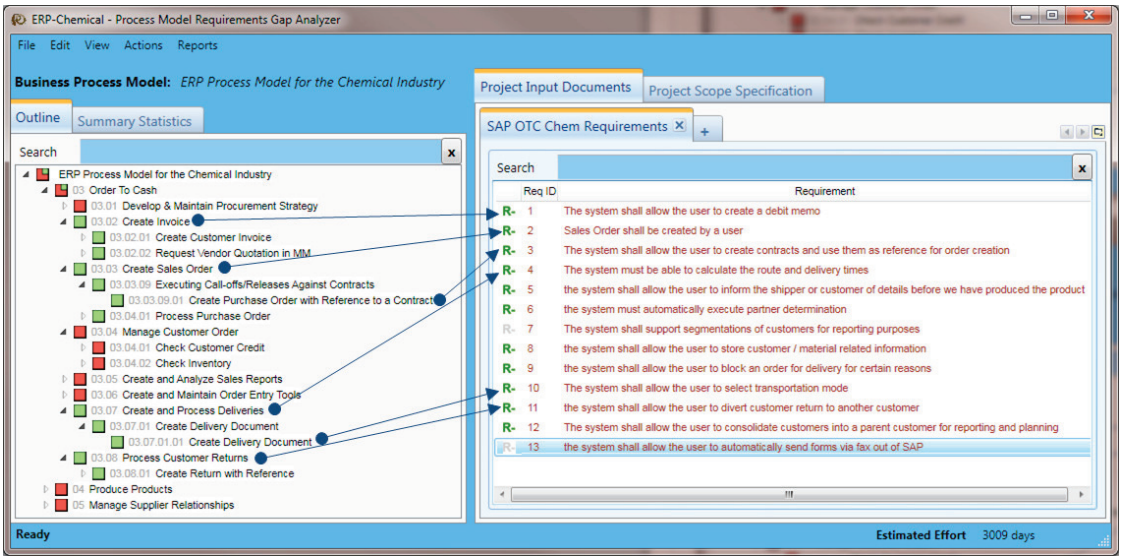

Figure 1: Screen shot of ProcGap User Interface.

$SIM_{sem}(T_R, T_C)$ is the semantic similarity score between any two terms T_R and T_C and is based on semantic relationship between the words. It is derived using the extracted semantic graph discussed in the next section and is defined as:

$$SIM_{sem}(T_R, T_C) = {}^{1} \begin{cases} 1 & \text{if } sameAs(T_R, T_C) \\ \alpha_1 & \text{if } subClassOf(T_R, T_C) \\ \beta_1 & \text{if } partOf(T_R, T_C) \\ \alpha_2 & \text{if } subClassOf(T_C, T_R) \\ \beta_2 & \text{if } partOf(T_C, T_R) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Where, $sameAs(T_R, T_C)$ is a function that returns true if two elements in the semantic graph are equivalent. $subClassOf(T_R, T_C)$ is a function that returns true if T_R is a sub-class of T_C in the semantic graph. $partOf(T_R, T_C)$ is a function that returns true if T_R is a part-of T_C in the semantic graph.

We represent the semantic graph using Web Ontology Language (OWL)(W3C 2004) and all three functions are implemented using the Jena reasoning engine(Jena 2009). While $sameAs$ and $subClassOf$ are natively supported by Jena and OWL, we augmented the Jena reasoner with some simple rules to implement the $partOf$ function. Some examples of calculating semantic similarity score are shown in rows 2 and 3 of Table 1.

$SIM_{Lin}(T_R, T_C)$ is the information theoretic similarity between two terms based on the approach proposed in (Lin 1998). It calculates the similarity between two terms based on their relative positions in a taxonomy created from WordNet.

$$SIM_{Lin}(T_R, T_C) = \frac{2.IC(lcs(T_R, T_C))}{IC(T_R) + IC(T_C)} \tag{4}$$

Where the information content of a term t, $IC(t)$ is defined as $-ln\left(\frac{freq(t)}{freq(r)}\right)$. Where $freq(t)$ and $freq(r)$ are, respectively, the frequencies of the term t and the root r of the taxonomy. The *lowest common subsumer* (lcs) of two terms T_R and T_C, that is the lowest node in the taxonomy that subsumes both the terms.

The similarity between two prepositional objects, $SIM_P(P_R, P_C)$ is defined as:

$${}^{2}SIM_P(P_R, P_C) = \begin{cases} 1 & \text{if } P_R = null \\ \gamma & \text{if } P_C = null \\ SIM(P_R, P_C) & \text{otherwise} \end{cases} \tag{5}$$

The prepositional object similarity is designed based on the following hypothesis - if a text has a preposition and the other does not then the prior one is more specific than the second one. For e.g., *create contract for e-commerce* is considered to be more specific than *create contract*.

Weighted Cosine Similarity

Cosine similarity models a sentence as a bag of words. It does not depend on the structure of the sentence and is particularly useful for poorly formed or complex sentences where it is hard to extract the VOP triples. We have designed our approach to benefit from extracted VOP triples and terms in the semantic graph. The cosine similarity of two vectors is defined as the dot product between them. The weighted

[1]We empirically came up the following values for the constants $\alpha_1 = 0.95$, $\beta_1 = 0.85$, $\alpha_2 = 0.85$, $\beta_2 = 0.75$. They are designed to give penalize requirements that are more general than capabilities and reward requirements that are more specific that capabilities.

[2]We empirically came up with $\gamma = 0.8$. The aim is to penalize differences between verbs or nouns more than the prepositional objects.

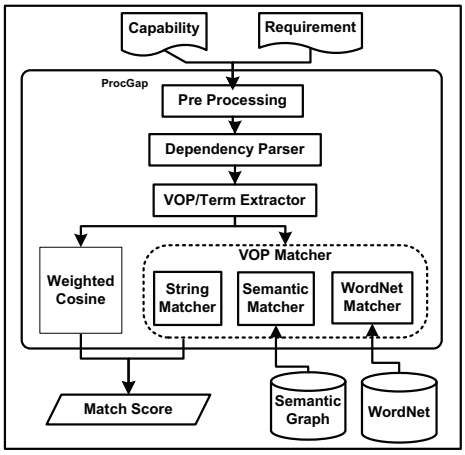

Figure 2: Matching Process

cosine similarity between a requirement R and capability C is defined as:

$$\cos(R, C) = \vec{R}.\vec{C} \tag{6}$$

Where $\vec{R}$ and $\vec{C}$ are term vectors representing the requirement R and the capability C respectively. Term vector $\vec{T} = \{f_1 t1, f_2 t_2, ..., f_n t_n\}$; where, f_i is the weighted frequency of term t_i in the term vector $\vec{T}$ and is defined as:

$$f_i = \begin{cases} 0 & \text{if } t_i \in \{StopWords\} \\ \tilde{f}_i + 1 & \text{if } t_i \in \{< V, O, P >\} \\ \tilde{f}_i + 1 & \text{if } t_i \in SemanticGraph \\ \tilde{f}_i & \text{otherwise} \end{cases} \tag{7}$$

Here $\tilde{f}_i$ is the frequency of word W_i in sentence S. Some example word vectors and match scores calculated using weighted cosine similarity are shown in rows 4, 5 and 6 of Table 1.

Calculating the final match score

Finally, the overall similarity score $M(R, C)$ between a requirement and a capability is calculated by taking the maximum score from VOP matching (Equation 1) and weighted cosine similarity (Equation 7).

$$M(R, C) = \max\{cos(R, C), VOP(R, C)\} \tag{8}$$

The match scores for a number of requirements and capabilities are shown in Table 1. We experimentally determined the threshold of similarity score to be 0.625.

Semi-Automatically Generating A Semantic Graph

Based on our empirical evaluation, we noticed that WordNet, which is a general purpose thesaurus, did not have enough relationships about the process model that we were using. For example, it is not possible to use WordNet to derive that *debit memo* is a special type of *invoice* or whether *transportation mode* is a typical attribute of a *delivery document*. However, since users often refer to different common attributes of objects such as *invoice* and *purchase order*, our approach was to model the relationships between commonly used terms in the domain of the process model.

Since the same process model is used over a number of projects, creating a domain-specific semantic graph seemed a reasonable approach to capture such relationships. This section will describe approach used by ProcGap to create the semantic model. It is loosely based on existing approaches for automatic ontology extraction such as (Dahab, Hassan, and Rafea 2008) and (Alani et al. 2003). To start with, the top twenty frequently occurring nouns (for e.g., order, contract) and verbs (for e.g., maintain, create) in a process model are added as nodes of the graph. Then, the most frequently occurring phrases are extracted using a high pass filter with a lower cutoff (5) and added to the graph. Finally, relationships are added manually with some suggestions from the tool. ProcGap either uses simple heuristics (e.g. sales order is a subclass of order) or information from external sources such as Wikipedia to suggest the relationships (e.g. Debit Memo is a synonym of Invoice) between the nodes. A snapshot of a semantic graph is shown in Figure 3.

Evaluation

In this section, we will present an evaluation based on a large industry project from the chemical industry. The evaluation was performed on a requirements document from a project team who wanted to map their requirements to their industry-specific process model. The scope of the project included implementing different aspects of their multi-national supply chain process such as accepting customer orders, fulfilling the orders, invoicing for the orders and shipping the orders. We used ProcGap to automatically map their requirements to the standardized "ERP Process Model for the Chemical Industry" created by a different group in our organization. That particular process model has 3116 capabilities. The project team gave us a document with 189 requirements. We used the requirements provided by the team as-is, therefore some are well-formed, while others are not.

To evaluate our approach, we performed two kinds of experiments. For the first experiment, we compared our matching approach to a number of other approaches. For the second experiment, we conducted a case study with 6 potential users of ProcGap.

Creating the Gold Standard

To help us evaluate the automatic mapping, a team of three project members created a Gold Standard by manually mapping the requirements. For a mapping to be accepted in the Gold Standard, agreement of at least two of the three members was needed. The gold standard contains 334 mappings between the requirements and the capabilities. Out of the 189 requirements, 42 requirements were not mapped to any capability, because they did not have any corresponding capabilities in the process model. Some of the requirements map to more than one capability.

	Capability (C)	Mapped Requirement (R)	Score	Details
1.	Create sales order	Sales order shall be created by a user.	1.0000	V: SIM_{str}(created, create)=1 ; O: SIM_{str}(sales order, sales order)=1
2.	Create invoice	The system shall allow the user to create a debit memo.	0.8500	V: SIM_{str}(create, create)=1 ; O: SIM_{sem}(debit memo, invoice) =0.85 $\cdot$ hasSubClass(invoice, debit memo)= T
3.	Create delivery document	The system shall allow the user to select transportation mode.	0.7225	V: SIM_{sem} (select, create) = 0.85 $\cdot$ partOf(select, create) = T ; O: SIM_{sem}(transportation mode, delivery document) = 0.85 $\cdot$ partOf(transportation mode, delivery document) = T
4.	Create Purchase Order with Reference to a Contract	The system shall allow the user to create contracts and use them as reference for order creation	0.8366	$\vec{C}$ {purchas=2, order=2, refer=2, creat=2, contract=2} ; $\vec{R}$ {creation=1, order=2, refer=1, creat=2, contract=2}
5.	the system allow the user to create contract	the system shall allow the user to cancel the contract	0.0000 0.0000	V: SIM(create, cancel) = 0 ; O: SIM_{str} (contract, contract) = 1 $\vec{C}$ {creat=2, contract=2} ; $\vec{R}$ {cancel=2, contract=2}
6.	The system shall allow the user to create Purchase Order for ERP	the system shall allow the user to create Purchase Order for SRM	0.5000 0.0000	$\vec{C}$ {purchas=2, order=2, creat=2, erp=2} ; $\vec{R}$ {srm=2, purchas=2, order=2, execut=2} V: SIM_{str} (create, create) = 1 ; O:SIM_{str} (Purchase Order, Purchase Order) = 1 ; P: SIM(ERP, SRM) = 0

Table 1: This table depicts a set of mappings, associated mapping scores and their calculation details.

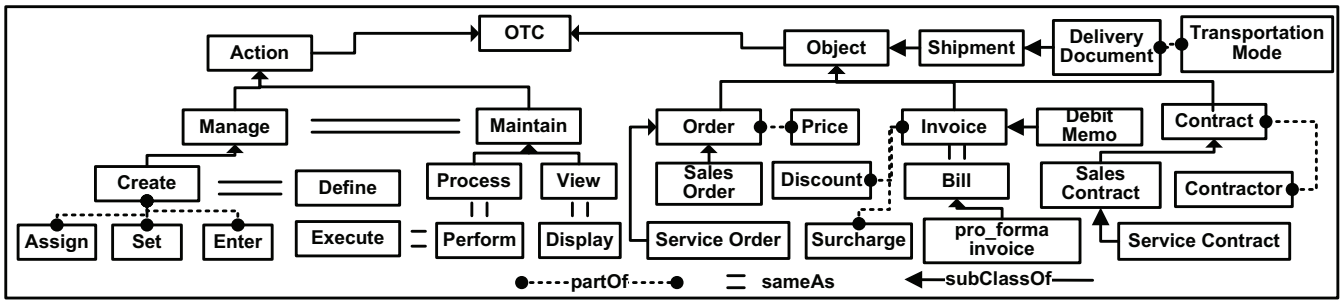

Figure 3: Snapshot of semi-automatically created semantic graph

User Group with ProcGap	Participant 1			Participant 2			Participant 3			Average		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
	0.87	0.68	0.76	0.77	0.72	0.74	0.78	0.73	0.75	0.81	0.71	0.75
User Group with manual approach	Participant 4			Participant 5			Participant 6			Average		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
	0.29	0.04	0.08	0.37	0.08	0.13	0.37	0.04	0.08	0.34	0.06	0.09

Table 2: Case Study: ProcGap Effectiveness

Experiment I - Comparison with other techniques

Table 3 shows precision, recall and f-measure(F1) scores for different matching strategies. This clearly shows that the fifth approach (Weighted Cosine + VOP + Semantic Graphs), which is used by ProcGap for matching requirements to capabilities yields a better F1 score than any other strategy. The second approach, based on our implementation of TF-IDF (Term Frequency Inverse Document Frequency) based cosine similarity gives very low scores in both precision (0.16) and recall (0.03) for two reasons -1) It does not use domain-specific information and 2) Using IDF decreases the weight of frequently used terms such as *invoice*. The second approach based on our implementation of the sentence similarity approach proposed in (Li et al. 2006) has a higher recall (0.33) because it uses WordNet, however, it suffers from poor precision (0.11). This is because a general purpose lexical database like WordNet is not enough to disambiguate between domain specific entities (objects), such as the ones used in the requirements and the process model. The third approach, weighted cosine with semantic graphs has high precision (0.87) and slightly lower recall (0.27) than the Sentence Similarity based approach. This shows the value of the domain-specific semantic model. This approach is analogous to the approach proposed in (Hayes, Dekhtyar, and Sundaram 2006). In the fourth approach (Weighted Cosine + VOP), the introduction of VOP improves the recall (0.37) significantly while keeping a higher precision (0.74). VOP gives importance to the structure of the sentence unlike first three approaches , hence it helps to find more correct mappings. Finally, the fifth approach (Weighted Cosine + VOP + Semantic Graphs) combines the benefits of using a Semantic Graph along with rule based parsing and co-

	Approach	Precision	Recall	F1
1	TF-IDF	0.16	0.03	0.05
2	Sentence Similarity (Li et al. 2006)	0.11	0.33	0.17
3	Weighted Cosine + Semantic Graph (Hayes, Dekhtyar, and Sundaram)	0.87	0.27	0.41
4	Weighted Cosine + VOP	0.74	0.37	0.49
5	Weighted Cosine + VOP + Semantic Graph (ProcGap)	0.73	0.62	0.67

Table 3: ProcGap matching vs. other approaches

sine similarity and maintains a high precision (0.73) with the highest recall (0.62).

Experiment II - Case Study

The second experiment was a case study to verify if there is statistical evidence that demonstrates that users find more mapping (higher recall) that are more accurate (higher precision) with the help of ProcGap than with a manual approach. We used 6 Accenture employees who had 2-10 years of experience in the IT industry with varying amount of expertise about the chemical industry. Three of them were randomly chosen to manually perform the task of mapping the requirements document to the process model. They were given the requirements document and the process model in two tabs of a Microsoft Excel sheet and were free to use keyword based search provided by Excel. The other three were asked to perform the same task with the help of ProcGap, i.e., their starting point was a set of mappings generated by ProcGap and they used the interface provided by ProcGap (shown in Figure 1).

Data collected from Case Study Table 2 shows the precision, recall and F1 scores for all 6 participants from Experiment II. The average recall by the group that uses ProcGap is 0.71 and the average recall by the group that created the mappings manually is 0.06. This shows that users were able to find significantly more mappings with the help of ProcGap. This is because of two reasons - 1) users are overwhelmed by the size of the requirements document and capability model and 2) they are not able to get beyond simple keyword search and only get the most obvious matches. The average precision by the group that uses ProcGap is 0.81 and the average precision by the group that created the mappings manually is 0.34. This can be explained by the fact that the users go for the most obvious match and do not apply any thresholds like ProcGap. The Wilcoxon-mann Whiteney test showed that the group using ProcGap outperformed the manual group, both in terms of precision and recall with the level of significance at 0.05.

Conclusions and Future Work

We have presented a tool called ProcGap that helps users map natural language requirements to process models. We use a combination of IR and rule based techniques over dependency parsing that allow ProcGap to automatically match both well-formed or poorly formed requirements and capabilities. Our approach also leverages a semi-automatically generated semantic graph that is more suited for the domain-specific process models than a general purpose lexicon such as WordNet. We have evaluated our approach based on data from an industry project and demonstrated that ProcGap outperformed some previous approaches and also increased productivity of users. We have presented a detailed evaluation of a single project, since it was not possible to calculate the recall without creating a gold standard. ProcGap has been used at 6 other projects and the precision was reported to be over 0.7 for all of them. Our future work involves focusing on richer process models to handle dependencies and conflicts. In addition to further studying the usefulness of ProcGap, we will also be exploring using this mapping technology for other tasks such as mapping patient care notes to models such as SNOMED.

References

Accenture. 2011. Accenture business process repository. https://microsite.accenture.com/BPM/Documents/BPM-L%20Brochure-100410.pdf.

Alani, H.; Kim, S.; Millard, D. E.; Weal, M. J.; Hall, W.; Lewis, P. H.; and Shadbolt, N. R. 2003. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems* 18(1):14–21.

ARIS. 2011. Aris reference models. http://www.ids-scheer.com/us/en/ARIS/ARIS‘¨Reference‘¨Models/82854.html.

Dahab, M. Y.; Hassan, H. A.; and Rafea, A. 2008. Textontoex: Automatic ontology construction from natural english text. *Expert Syst. Appl.* 1474–1480.

Daneva, M. 2004. Erp requirements engineering practice: Lessons learned. *IEEE Softw.* 21(2):26–33.

de Marneffe, M.; MacCartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Hayes, J. H.; Dekhtyar, A.; and Sundaram, S. K. 2006. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Trans. Softw. Eng.* 32(1).

IBM. 2011. Ibm websphere industry content packs. http://www-142.ibm.com/software/products/gb/en/inducontpack/.

Jena. 2009. Jena api. http://jena.sourceforge.net/ontology/.

Jirapanthong, W., and Zisman, A. 2009. Xtraque: traceability for product line systems. *Software and Systems Modeling* 8:117–144.

Li, Y.; McLean, D.; Bandar, Z. A.; O'Shea, J. D.; and Crockett, K. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. on Knowl. and Data Eng.* 18(8):1138–1150.

Lin, D. 1998. An information-theoretic definition of similarity. In *ICML '98*, 296–304. Morgan Kaufmann Publishers Inc.

Martin F., P. 1980. An Algorithm for Suffix Stripping. *Program* 14(3):130–137.

Mihalcea, R.; Corley, C.; and Strapparava, C. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, 775–780.

Pnina, S.; Boaz, G.; Dov, D.; and Yair, W. 2001. Modelling off-the-shelf information systems requirements: An ontological approach. *Requir. Eng.* 6(3):183–199.

W3C. 2004. Owl web ontology language. http://www.w3.org/TR/owl-features/.

Zachos, K., and Neil A. M., M. 2008. Inventing requirements from software: An empirical investigation with web services. In *RE 08*, 145–154.