Mining goal refinement patterns: Distilling know-how from data

Metta Santiputri[1,3], Novarun Deb[2], Muhammad Asjad Khan[3], Aditya Ghose[3], Hoa Dam[3], and Nabendu Chaki[2]

[1] Department of Informatics, State Polytechnic of Batam, Batam 29461, Indonesia metta@polibatam.ac.id
[2] Department of Computer Science and Engineering, University of Calcutta, Kolkata, India novarun@acm.org, nabendu@ieee.org
[3] Decision Systems Lab, School of Computing and IT, University of Wollongong, Australia NSW 2522 aditya@uow.edu.au, hoa@uow.edu.au

Abstract. Goal models play an important role by providing a hierarchic representation of stakeholder intent, and by providing a representation of lower-level subgoals that must be achieved to enable the achievement of higher-level goals. A goal model can be viewed as a composition of a number of *goal refinement patterns* that relate parent goals to subgoals. In this paper, we offer a means for mining these patterns from enterprise event logs and a technique to leverage vector representations of words and phrases to compose these patterns to obtain complete goal models. The resulting machinery can be quite powerful in its ability to mine *know-how* or *constitutive norms*. We offer an empirical evaluation using both real-life and synthetic datasets.

Keywords: Goal model mining, goal refinement, know-how

1 Introduction

Goal models play a critical role in requirements engineering, by providing a hierarchic representation of statements of stakeholder intent, with goals higher in the hierarchy (parent goals) related to goals lower in the hierarchy (sub-goals) via AND- or OR-refinement links. Goal models encode important knowledge about feasible, available alternatives for realizing stakeholder intent represented at varying levels of abstraction. A number of prominent frameworks leverage goal models, including KAOS, i* and Tropos[8].

There is a growing realization that data analytics (this term being liberally interpreted to denote a broad repertoire of machine learning, data mining and natural language processing techniques) have an important role to play in software engineering in general, and requirements engineering in particular. In that spirit, this paper addresses the question: *can enterprise goal models be mined from readily available enterprise data?*. It is useful to distinguish, at this point, the exercise of mining *goal models* from the exercise of mining *goals*. That latter problem is arguably more difficult, since user goals or stakeholder intent are often never manifested in enterprise data, and are often not explicitly articulated either. Knowledge about how a goal might be refined into lower-level sub-goals is a different matter altogether. Goal refinements that have been deployed before (either explicitly or implicitly) are ultimately manifested in operational data. Our intent in this paper is to leverage data of this form.

Mining goal models adds value in a number of ways. *First*, it offers a way around the *model acquisition bottleneck* (where the high investments associated with careful modeling often prevents businesses from leveraging the full value of goal modeling). While our approach does not guarantee that all models mined will be correct and accurate, it does ensure that the goal models (or model fragments) that are mined can be quickly deployed with minimal editing (the requirement for oversight and editing by analysts remains). Overall, the approach improves the productivity of modelers/analysts; instead of starting with a "blank sheet", our machinery generates "first draft" models or model fragments that can be composed to obtain usable models. *Second*, our approach could potentially improve model quality, by mining from execution histories from which "undesirable" executions have been filtered out. *Third*, model *anti-patterns* can be mined from "undesirable" execution data. *Fourth*, this machinery can be used for *goal conformance* checking.

Goal models can also be viewed as statements of *know-how*, where an AND-decomposition provides the know-how for achieving a parent goal by satisfying a set of sub-goals. Mining know-how patterns is independently useful. In particular, it permits us to use goal models as *effectors*, where a goal model is used to specify the desired state of the enterprise while decomposition via a sequence of know-how patterns enables us to identify the operational interventions which would help realize the desired state of the enterprise.

AND-refinement patterns can also be viewed as *constitutive norms* [4]. A constitutive norms specifies how the act of achieving conditions c_1, c_2, c_n *counts as* achieving condition c (we can also, without loss of generality, replace conditions with goals or actions). For instance, the acts of putting a tea bag in a cup followed by puring hot water into the cup *counts as* making tea. The account we offer in this paper can thus be also viewed as an account of constitutive norm mining.

We address two problems in this paper. First, we address the *goal refinement pattern mining problem*, where a goal refinement pattern is of the form $sg_1, sg_2, \ldots sg_n \rightarrow G$ where G is the parent goal while each sg_i is a sub-goal, and where the statement is that the act of achieving each sub-goal conjointly leads to the achievement of the parent goal. These latter are referred to as AND-refinement patterns, and are the main focus of this paper (OR-refinement patterns can be mined via small variants of the techniques discussed here, but a full discussion is omitted due to space constraints). Second, we address the problem of composing individual goal refinement patterns into *goal trees* (more generally goal graphs) which describe not only how a goal into subgoals, but also how these subgoals can be further refined into sub-subgoals and so on.

We present the general approach in Section 2. The identification of goal refinement patterns involves mining event logs (partitioned by levels of abstraction) that leverage *temporal correlation patterns* between goals and subgoals (recall that an *event log* is a collection of time-stamped events). The composition of goal refinement patterns relies on matching subgoal in one refinement pattern with the parent goal of another such pattern - we use word2vec [5] to identify *semantic similarity* between words and phrases that appear in the goals and subgoals for this purpose. We present a preliminary empirical evaluation of these proposals in Section 3.

2 General approach

Temporal correlation patterns relating goals and subgoals: A goal and its subgoals are typically related via *temporal correlation patterns* which impose temporal constraints on the achievement of the parent goal relative to the achievement of the subgoals. One such pattern (and the one we will leverage in the empirical evaluation in this paper) requires that event denoting the achievement of the parent goal occur immediately or soon, after the events denoting the achievement of the subgoals. We shall call these *sequential correlations.* Other examples of temporal correlation patterns leverage relations from Allen's Interval Algebra [2]. In some settings, we might require the interval over which each subgoal is achieved be included entirely (using the **during** relationship from the Interval Algebra) in the interval over which the parent goal is achieved. In some settings it might make sense to relate these intervals using the **meets, finishes** or **is equal to** relations from Interval Algebra.

Mining goal refinement patterns from multi-layered event logs: Independent of which temporal correlation pattern applies in a given setting, it is critical that the input event logs are partitioned into layers based on different levels of abstraction. A key assumption underpinning this proposal is that events denoting the achievement of parent goals appear in a log of more abstract events, while events denoting the achievement of subgoals appear in logs of more refined (or lower-level) events. In other words, we assume a hierarchy of levels $L_1, L_2, \ldots$ such that L_i is always at a higher level of abstraction than L_{i+1}. The idea is that goal refinement always occurs between goals manifested by events in adjacent levels in this hierarchy. The key question to address now is: How do we obtain this partitioning/hierarchy? Possible strategies include:

- Leveraging part-whole relationships between objects: We know that a photo, a front page, an embedded chip, a visa or an expiry date are parts of a more abstract object called a passport. Any event involving the passport photo, or a visa etc. will belong to a lower level in the hierarchy than any event involving the passport.
- Leveraging the source of the data: We know that any event from a process log is likely to be lower in an abstraction hierarchy than any event in a message log. Similarly, events that manifest in the IT infrastructure are typically lower in abstraction than events that involve applications, which in turn are lower level than events concerning business services.
- Leveraging the organizational hierarchy: We know that events associated with roles lower in the organizational hierarchy will likely be lower in the abstraction hierarchy than events associated with roles higher in the organizational hierarchy. The intuition is that employees in a business unit are usually

tasked with achieving lower-level goals than the manager of that business unit. Indeed, the goals of the manager rely on the achievement of the sub-goals that the employees in that unit are tasked to achieve. The employee-level goals can thus be viewed as AND-refinements of the manager-level goals.

With the abstraction hierarchy of events thus obtained, our task in now to mine (temporal) sequential correlations between events in adjacent levels of the abstraction hierarchy. Thus a passport photo check, a passport validity check, a visa check and a passport stamping event would be followed soon after by a higher-level event indicating that an immigration check has been completed. We would expect to see this pattern repeated frequently. If this frequency meets a user-specified threshold, we conclude that it is indicative of a goal refinement pattern.

Composing goal refinement patterns: The challenge in composing goal refinement patterns to obtain goal models (or goal trees) is the difficulty in relating semantically similar, but syntactically highly distinct, specifications of goals and subgoals. For instance, a subgoal might be represented in natural language as: *log labour hours for billing*. Quite separately, we might find a mined goal refinement pattern for a parent goal represented textually as: *track technician time for charging the customer*. Human intuition suggests that these two goals are semantically quite similar, and any available know-how for the latter would also be useful for the former. Our strategy is to use a state-of-the-art vector encoding of words and phrases, called *word2vec* [5] which is effective in identifying semantic similarity. Word2vec learns vector representations of words and phrases such that semantically similar ones are projected in close proximity to each other in the vector space. Given a pair of phrases, word2vec returns a real-valued measure of semantic similarity (the higher the value, the more similar the phrases are). By setting an appropriate threshold for the similarity measure (this will require domain-specific tuning), we can connect a phrase describing a subgoal in one goal refinement pattern with a phrase describing a parent goal in another goal refinement pattern.

3 Evaluation

The sequential patterns of interest are those where a sequence of events at level L_{x+1} are immediately followed by an event at level L_x, where L_x is always at a higher level of abstraction than L_{x+1}. Therefore, in both setting, to determine the correlation between events in adjacent levels of the abstraction hierarchy, we first create a joined log of the form: $\langle\langle\langle\langle\langle T_{11}\rangle, \ldots, \langle T_{1n}\rangle\rangle, T_1\rangle, \ldots, \langle\langle\langle T_{i1}\rangle, \ldots, \langle T_{im}\rangle\rangle, T_i\rangle, \ldots$
$\langle\langle\langle T_{j1}\rangle, \ldots, \langle T_{jk}\rangle\rangle, T_j\rangle$ where each $\langle T_i, T_{i+1}\rangle$ pair represents contiguous events in the higher level event log L_x and each T_{ij} represents the j-th event in the lower level L_{x+1} related to event T_i and executed during T_i's execution time. This joined log represents the sequence database provided as input to the frequent closed sequential pattern miner.

In our instance, we will apply BIDE+ algorithm, but other similar algorithms could be used instead. The miner then returns the sequence $\langle\langle T_{i1}, \ldots, T_{in}\rangle, T_i\rangle$

that satisfy the threshold $min_{support}$ to represent the correlation of event T_i of the higher level abstraction and event sequence $\langle T_{i1}, \ldots, T_{in} \rangle$ of the lower level abstraction. The threshold ($min_{support}$) is bounded by the number of distinct cases which a sequence with an event suffix T_i occurs. As with any association rule mining technique, $min_{support}$ represent the support—higher values of this can give us more reliable results but rule out potentially interesting rules and vice versa.

Evaluation with synthetic data: We perform the evaluation with an event log of a telephone repair process[4]. This example describes a business process in a telephone repair company. The event log consists of seven activities: *Register, Analyze Defect, Repair (Complex), Repair (Simple), Test Repair, Restart Repair, Inform User*, and *Archive Repair*.

We consider this event log as the event log in the higher level, namely the events in the business layer. This event log of the higher level contains 7152 entries. For sequence pattern mining, SPMF pattern mining library [9] is used to run frequent sequential pattern mining algorithm BIDE+ [10].

We then generated the joined log with each record represents a sequence of function invocation events from the simulated lower level event log followed by a complete event of the task in the higher level event log. Afterwards, we used the sequence pattern miner to mine the task-function correlation. The result is shown in Table 1.

Table 1. Result

Goal	Sub-Goals
Register	create new customer service ticket, print repair receipt for the customer, email repair order confirmation
Analyze Defect	perform a series of standard diagnostic tests to identify fault, assign a (simple/complex) label to the ticket after defect assessment,contact customer and seek approval for complex repairs
Repair	Order replacement parts if repair type is complex,log labour hours for billing, Change ticket status to ‘fixed’
Test Repair	Perform series of standard tests, Send phone to repair department if not approved and restart repair
Restart Repair	Update ticket status to ‘in diagnosis’
Inform User	Send out email/text to customer, receive payment based on hours and repair type
Archive Repair	Close and archive ticket

Evaluation with real-life data: We use the data from BPI Challenge 2015 (BPIC’15)[5] which features building permit application process in five Dutch municipalities from year 2010 until 2015. The permit application consists of three main process: (1) the municipality receives permit application, (2) the municipality performs a number of checks on the request requirements (information

[4] http://www.processmining.org/_media/tutorial/repairexample.zip
[5] https://www.win.tue.nl/bpi/doku.php?id=2015:challenge

Table 2. The sequential correlations between events from different groups of resources

Events	Correlated to event
01_HOOFD_01, 01_HOOFD_02	01_HOOFD_03
01_HOOFD_01, 01_HOOFD_02	01_HOOFD_04
01_HOOFD_01, 01_HOOFD_02	01_HOOFD_05
01_HOOFD_05	01_HOOFD_06
01_HOOFD_06, 01_HOOFD_10	01_HOOFD_11
01_HOOFD_1 01_HOOFD_2 01_HOOFD_3 01_HOOFD_4	01_HOOFD_5
01_HOOFD_4, 01_HOOFD_5	05_EIND
01_HOOFD_4	01_HOOFD_5
01_HOOFD_5	06_VD

completeness) and determines procedure applicable, and (3) competent authority evaluates permit application and decides on permit. Among the five Dutch municipalities, the first municipality has the most well-structured organization [3, 7]. Hence, we use the dataset of only the first municipality in our exercise.

We take the event log for the first municipality (BPIC15_1.xes) and view the log using the Dotted Chart visualization provided by ProM [1]. The Dotted Chart helps in identifying the activities performed by the resources (as specified in the log). Using the Dotted Chart, we can group similar sets of activities and identify organizational roles for each such group. These organizational roles relate to the different hierarchic levels that exist within the Dutch municipality.

We generate event sequences from the event logs and use them as input for the sequential pattern mining. The result is shown in Table 2.

Table 3. The sequential correlations provided to the modeler

registration received, reception → change procedure, application forwarded
regular procedure, registration date received → subcases completed
procedure started, subcases checked, content assessed, permit issued → permit sent to stakeholder
permit issued, permit sent to stakeholder → case terminated
objection registered → procedure term extended

In our result, there are nine correlations identified. Seven of the correlations are between events from adjacent levels, however two correlations (identified by dashed lines), namely the correlation between 01_HOOFD_06, 01_HOOFD_10 and 01_HOOFD_11 and the correlation between 01_HOOFD_4, 01_HOOFD_5 and 05_EIND, are between events from non-adjacent levels. These two patterns were picked up by the sequential pattern miner from the input sequences since they pass the threshold. This situation indicates that the case handovers between resources in the municipality do not always follow the structure or restricted to adjacent levels, but may also happen between resources from non-adjacent levels.

We leveraged the help of an expert modeler to asses the correlations discovered through our techniques on both the soundness and the completeness. We provide the modeler with the dataset consists of *Case ID*, *ActivityNameEN* (which describes the activity in English, instead of Activity name/code) and *Resource*. We also provide them with the correlations discovered earlier. The correlations that we provide for the modeler is shown in Table 3. We use the "$\rightarrow$" symbol to make the correlations easier to understand. The context of the correlations given to the modeler is that the consequence (right of the arrow) is achieved if all the antecedents (left of the arrow) are achieved, or the antecedent has to be performed or completed first before the activities in the consequences can be performed.

The modeler concluded that all of the correlations are in accordance with the event log. They also noticed that there are correlations which are not included in the list, such as the phase *application received* $\rightarrow$ *confirmation receipt sent* and *procedure confirmation sent, senddate procedure confirmation entered* $\rightarrow$ *registration date published*. However, after we look closely into the event log, we noticed that these events are performed by resource(s) from the same group, therefore they do not included in our result. This is not to discount that the correlations are not valid, however these correlations are not relevant in our exercise. The modeler also noticed that there are activities that they deemed to be important but do not shown in the result, such as *ask stakeholders views* and *request further information*. We argued that we only picked up main activities which occur frequently and we also did not include detailed activities, thus there are activities which do not included in our result.

Evaluating goal-subgoal similarity: For evaluation we rely on Google's pre-trained word2vec model which they have publicly made available. It includes word vectors for a vocabulary of 3 million words and phrases that has been trained on approximately 100 billion words from a Google News dataset. Although for this evaluation, we used a pre-trained model, training a model with a smaller but more targeted and domain-specific corpora is not hard. We have done this but have not achieved results thus far that surpass the results we have obtained using the pre-trained model. We took the goal refinement patterns obtained in the evaluation using the phone repair scenario described above (8 in total), and extended these with a repertoire of 40 additional goal refinement patterns (this was necessary to be able to further refine the sub-goals initially obtained from the mining of a 2-level event log).

We ran our program on a large amazon EC2 instance with 16GB RAM using 64-Bit Python. For querying the model, our program first loads the 3.6 GB pre-trained model in memory using the word2vec module of the Gensim Library [6]. Then given two input phrases it computes the average vector for both and calculates the cosine similarity between the two phrase vectors. Common English words (stop words) are removed during the similarity calculation.

The Word2Vec metric tends to place two words close to each other if they are semantically similar. In the results table below we can see that 'Print repair receipt for the customer' and 'Print customer service repair order' have a high similarity score even though the phrases use different vocabulary to explain the same sub-goal. The notion of similarity used here is just cosine distance (dot product of vectors). It's closer to 1 if the phrases are semantically similar and for two completely dissimilar phrases, the similarity is close to 0. e.g 'update issue status to 'in repair' and 'dissemble the phone components' refer to two different goals and are very far apart semantically thus receiving a score of 0.130887. In some cases like 'log labor hours for billing' and 'Track technician time for charging the customer' the score is neither too high nor too low. We can use a certain threshold e.g (0.60) to filter cases where we not fully confident of a semantic match.

Table 4. Goal Text Similarity Results

Sub-Goal Text	Sub-Goal(Alternative Text)	Similarity Score
create new customer service ticket	open new repair issue for the customer	0.623894
Print repair receipt for the customer	Print customer service repair order	0.863028
Send out courtesy product check-in confirmation email	Email repair order confirmation	0.742585
Perform a series of standard diagnostic tests to identify fault	troubleshoot the problem by following a step-by-step testing methodology	0.635703
Assign a label to the issue after defect assessment	specify fault type by adding a tag to the issue after diagnosing the problem	0.637491
Seek Customer approval for complex repairs	Ask permission from customer if a part replacement is required	0.640503
Order replacement parts if repair type is complex	Send out replacement request for new components	0.645564
Log labor hours for billing	Track technician time for charging the customer	0.469291
Change issue status to 'fixed'	update issue status to 'ready to review'	0.819758
Perform series of standard tests	Change issue status to either 'approved' or 'not fixed'	0.233664
Send phone to repair department if not approved and restart repair	Close and archive issue	0.277631
Update issue status to 'in repair'	dissemble the phone components	0.130887

4 Conclusion

The ability to mine goal models has important implications for requirements engineering, as well as a wide variety of other settings that benefit from goal modeling. The machinery that we present can therefore provide useful directions for future research and development. This machinery can also be used to mine know-how which can support enterprise innovation strategies in significant ways. The empirical evaluation presented in the paper is preliminary in nature, but provides evidence that suggests that there is merit in pursuing this general approach.

References

1. van der Aalst, W.: Process mining software. Process Mining Part V, 325–352 (2016)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
3. Blevi, L., den Spiegel, P.V.: Discovery and analysis of the Dutch permitting process. In: Fifth International BPIC. BPIC'15 (2015)
4. Boella, G., Broersen, J., van der Torre, L.: Reasoning about constitutive norms, counts-as conditionals, institutions, deadlines and violations. In: Pacific Rim International Conference on Multi-Agents. pp. 86–97. Springer (2008)
5. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
6. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. Valletta, Malta (2010)
7. Teinemaa, I., Leontjeva, A., Masing, K.O.: BPIC 2015: Diagnostics of Building Permit Application Process in Dutch Municipalities. In: Fifth International BPIC. BPIC'15 (2015)
8. Van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: Proceedings of the 5th IEEE International Symposium on RE. pp. 249–262 (2001)
9. Viger, P.F., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: SPMF: A Java Open-Source Pattern Mining Library. Journal of Machine Learning Research 15, 3389–3393 (2014)
10. Wang, J., Han, J.: BIDE: Efficient Mining of Frequent Closed Sequences. In: Proceedings of the 20th International Conference on Data Engineering. pp. 79–90 (2004)