Preprint version of chapter published in: Jorge Cardoso, Wil van der Aalst (eds.), *Handbook of Research on Business Process Modeling*, pp. 492-513, Information Science Reference, IGI Global, April 2009

Applied Sequence Clustering Techniques for Process Mining

Diogo R. Ferreira
IST – Technical University of Lisbon, Portugal

ABSTRACT

This chapter introduces the principles of sequence clustering and presents two case studies where the technique is used to discover behavioral patterns in event logs. In the first case study, the goal is to understand the way members of a software team perform their daily work, and the application of sequence clustering reveals a set of behavioral patterns that are related to some of the main processes being carried out by that team. In the second case study, the goal is to analyze the event history recorded in a technical support database in order to determine whether the recorded behavior complies with a predefined issue handling process. In this case, the application of sequence clustering confirms that all behavioral patterns share a common trend that resembles the original process. Throughout the chapter, special attention is given to the need for data preprocessing in order to obtain results that provide insight into the typical behavior of business processes.

1. INTRODUCTION

The field of process mining (van der Aalst & Weijters, 2004) is a new and exciting area of research, whose purpose is to develop techniques to gain insight into business processes based on the behavior recorded in event logs. There are a number of process mining techniques already available and most of them focus on discovering control-flow models (van der Aalst et al, 2003). There are also techniques to take into account data dependencies (Rozinat et al, 2006), and techniques to discover other kinds of models such as social networks among workflow participants (van der Aalst et al, 2005).

Process mining techniques such as the α-algorithm (van der Aalst et al, 2004), the inference methods proposed by (Cook & Wolf, 1995), the directed acyclic graphs of (Agrawal et al, 1998), the inductive workflow acquisition by (Herbst & Karagiannis, 1998), the hierarchical clustering of (Greco et al, 2005), the genetic algorithms of (Alves de Medeiros et al, 2007) and the instance graphs of (van Dongen & van der Aalst, 2004), to cite only a few, are all techniques that aim at extracting the control-flow behavior of a business process and representing it according to different kinds of models. All these techniques take an event log as input and as the starting point for the discovery of underlying process.

In many practical applications, however, the events that belong to a particular process can only be found among the events of other processes that are running within the same system. For example, events recorded in a CRM (Customer Relationship Management) system may belong to different processes such as creating a new customer or handling a claim submitted by an existing customer. Furthermore, even when focusing on a single process, the behavior in set of instances may be so diverse that it becomes appropriate to study different behaviors as separate workflows. Either way, the amount and diversity of activities recorded in an event log may be such that it becomes necessary to sort out the different existing processes before applying one of the above process mining techniques.

Sequence clustering is a particularly useful technique for this purpose, as it provides the means to partition a number of sequences into a set of clusters or groups of similar sequences. Although the development of sequence clustering techniques has been an active field of research especially in the area of bioinformatics – see for example (Enright et al, 2002), (Jaroszewski & Godzik, 2002) and (Chen et al, 2006) – its principles are equally applicable to other kids of sequence data. For example, in applications such as user click-stream analysis it is possible to use sequence clustering to discover the typical navigation patterns on a Web site (Cadez et al, 2003). The same approach can be used to discover the typical behavior of different processes, or to distinguish between different behaviors within a single process, for example to identify what is considered to be the normal flow and what is deemed to be exceptional behavior.

The use of clustering algorithms in association with process mining techniques has received increased attention in recent years: in (Greco et al, 2004), the authors represent each trace in a vectorial space in order to make use of the *k*-means algorithm to cluster workflow traces; (Alves de Medeiros et al, 2008) make use of a similar approach in order to perform hierarchical clustering; (Jung et al, 2008) also address hierarchical clustering by means of a special-purpose algorithm based on a cosine similarity measure; in (Song et al, 2008) the authors make use of several clustering algorithms, including *k*-means and self-organizing maps; (Ceglowski et al, 2005) make use of self-organizing maps in order to cluster hospital emergency data. This means that there are several techniques available for clustering workflow traces. In this chapter we focus specifically on the use of sequence clustering techniques.

The chapter is organized as follows: Section 2 explains how sequence clustering works in order to find a set of clusters of similar sequences. Section 3 provides a word of caution regarding the need for preprocessing before actually applying sequence clustering to a given dataset. Section 4 presents a case study on the application of sequence clustering to an activity log that has been collected manually during the daily work of a software development team. Section 5 presents a second case study on the application of sequence clustering to the history recorded in a technical support system, in order to determine to what extent the recorded behavior complies with a standard incident management process. Section 6 concludes the chapter by highlighting how the case studies illustrate both the potential and limitations of sequence clustering as a process mining technique.

2. SEQUENCE CLUSTERING

The general purpose of clustering algorithms is to organize a given set of objects into a set of clusters, where each cluster contains objects that are similar by some kind of measure. This measure depends on the kind of objects or data being used. For example, if the objects are data points in two-dimensional space, then the measure of similarity can be formulated as the proximity between data points. In this case, points that are close together in space are more likely to belong to the same cluster than points that are farther apart.

The same concept can be extended to sequence data with some adaptations. As illustrated in figure 1, given a set of input sequences and a set of clusters, the goal is to assign each sequence to one of the available clusters based on some similarity measure. In sequence clustering, each cluster is associated with a probabilistic model, usually a Markov chain. If the Markov chains for all clusters are known, then each input sequence is assigned to the cluster that can best produce such sequence. In general there can be more than one possible cluster, so the sequence is assigned to the cluster which can produce the input sequence with higher probability.

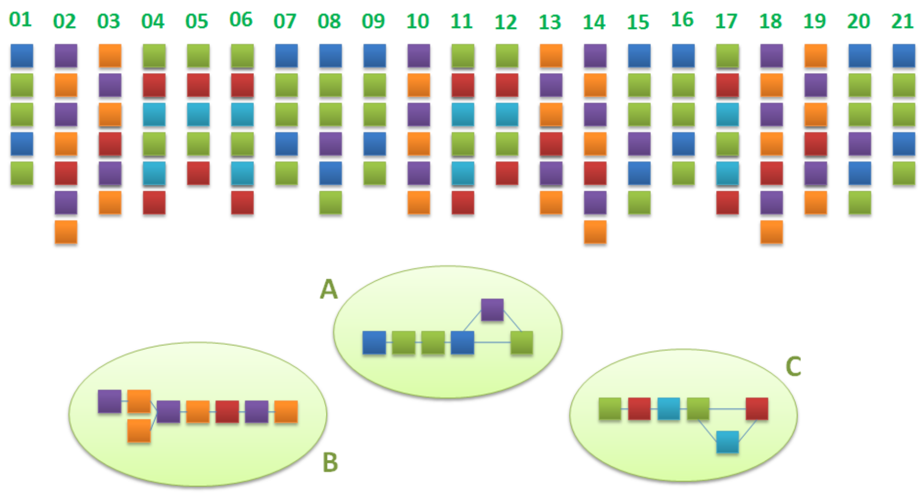


Figure 1. Basic concepts in sequence clustering

Since each cluster has its own Markov chain, the probability that an observed sequence belongs to a given cluster is the probability that the observed sequence was produced by the Markov chain associated with that cluster. For a sequence $\mathbf{x} = \{x_0, x_1, x_2,\ldots,x_{L-1}\}$ of length L this can be formalized as:

$$p(\mathbf{x} \mid c_k) = p(x_0; c_k) \cdot \prod_{i=1}^{L-1} p(x_i \mid x_{i-1}; c_k)$$

where $p(x_0;c_k)$ is the probability of x_0 occurring as the first state in the Markov chain associated with cluster c_k and $p(x_i|x_{i-1};c_k)$ is the transition probability of state x_{i-1} to state x_i in the same Markov chain.

Unfortunately, the Markov chains associated with each cluster [i.e. the probabilities $p(x_0;c_k)$ and $p(x_i|x_{i-1};c_k)$ in the formula above] are not given, because they are the actual result being sought. These cluster models represent the behavioral patterns found in the input dataset. Sequence clustering can be seen as an approach to discover these behavioral patterns. To explain how this is achieved, we will focus on the particular algorithm proposed by (Cadez et al, 2003).

The sequence clustering algorithm described by (Cadez et al, 2003) is a model-based clustering technique (Han & Kamber, 2006) that relies on an iterative Expectation-Maximization procedure (Dempster et al, 1977). The idea can be described as follows. If the cluster models (i.e. the Markov chains) were known, then we could assign sequences to clusters in the way described above. Once the sequences have been assigned, it is possible to re-estimate the cluster models based on the actual population of each cluster, i.e., from the set of sequences that belong to a cluster it is possible to re-estimate the Markov chain for that cluster. After that estimation, we can re-assign the sequences to clusters and again improve the estimate for the cluster models. By repeating this procedure over and over again, the cluster models will eventually converge to a set of Markov chains that no longer change. These are the desired behavioral patterns.

Two difficulties arise from this approach. One is how to obtain a first estimate for the cluster models so that this iterative procedure can be applied. The simplest solution to this problem is to randomize the cluster models, i.e. using a random guess as a starting point. The second issue is whether such procedure will actually converge. Fortunately, this has been proved by (Dempster et al, 1977) for the general framework of Expectation-Maximization (EM), which is one of the cornerstones of model-based clustering.

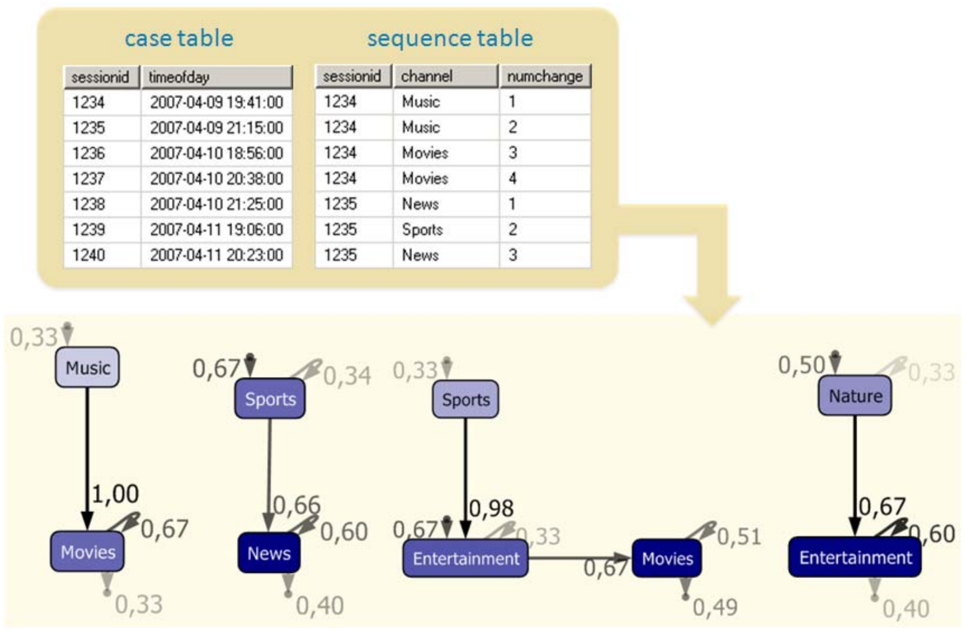
The algorithm of (Cadez et al, 2003) can therefore be described as follows:

1. Initialize the cluster models (i.e. the Markov chain for each cluster) randomly.
2. Assign each input sequence to the cluster that is able to produce it with higher probability (by the equation above).
3. Estimate each cluster model from the set of sequences that belong to that cluster.
4. Repeat steps 2 and 3 until the cluster models, and hence the assignment of sequences to clusters, do not change.

This algorithm has been implemented in Microsoft SQL Server 2005® Analysis Services, where it is known as Microsoft Sequence Clustering (MSC), and is readily available for use either programmatically via a Data Mining API or manually via a user-friendly interface in Microsoft Visual Studio 2005®.

The MSC algorithm must be provided two input tables: a case table and a sequence table. The case table contains one record for each sequence; it conveys the number of sequences in the input dataset together with some descriptive information about each sequence. The sequence table contains the steps for all sequences, where each step is numbered and labeled. The number is the order of occurrence within the sequence, and the label is a descriptive attribute that denotes the state in a Markov chain. The case and sequence tables have a one-to-many relationship: each sequence in the case table is associated with several steps in the sequence table by means of a case id.

Figure 2 illustrates a simple example where the members of a family have different ways of zapping through TV channels according to their own interests. Let us assume that each member always finds the TV switched off, and after turning it on, goes through a set of channels before turning it off again. Every time it is turned on, the TV generates a session identifier (case id) and records the sequence of channels (by channel type). Figure 2 shows part of the case and sequence tables, together with the cluster models found by running MSC on this dataset. Each cluster model shows the transition probabilities between states, as well as the entry and exit probabilities that are determined from the beginning and ending states of the sequences that belong to that cluster.


case table

sessionid	timeofday
1234	2007-04-09 19:41:00
1235	2007-04-09 21:15:00
1236	2007-04-10 18:56:00
1237	2007-04-10 20:38:00
1238	2007-04-10 21:25:00
1239	2007-04-11 19:06:00
1240	2007-04-11 20:23:00

sequence table

sessionid	channel	numchange
1234	Music	1
1234	Music	2
1234	Movies	3
1234	Movies	4
1235	News	1
1235	Sports	2
1235	News	3

Figure 2. Input tables and cluster models for the simple TV usage scenario

Assuming that the case and sequence tables are already available in an existing database, the main steps to run such an example in Microsoft SQL Server 2005® Analysis Services are the following:

1. In the “Business Intelligence Development Studio”, create a new “Analysis Services Project” – This creates a project that will contain several artifacts, namely data sources, mining models, etc., as described in the following steps.
2. Create a new “Data Source” and connect to the existing database – This step connects to the database that contains the input sequence data to be processed.
3. Create a new “Data Source View”, add the case and sequence tables to that view, and create a one-to-many relationship based on the case id attribute – This step is necessary in order to specify which tables are the case table and the sequence table in the database that we connected to the previous step.
4. Create a new “Mining Structure” by specifying the table columns that will be the input for MSC – This step will specify which columns of the case and sequence tables will serve as input data for the algorithm to be chosen in the next step.
5. Create a new “Mining Model” and specify the use of sequence clustering – From the set of data mining algorithms available, the one to be chosen is Microsoft Sequence Clustering (MSC).
6. Configure the algorithm parameters, such as number of clusters and the minimum number of sequences in each cluster (minimum support) – Each data mining algorithm may have several parameters which can be adjusted to produce best results.
7. Run the mining model and wait until the processing is complete – This will take from a couple of seconds to a couple of minutes, depending on the size of the input dataset.
8. Browse through results using the “Mining Model Viewer” – With this component it is possible to study the results from a number of views, and to get the Markov chain for each cluster.

Each of these steps is explained in thorough detail in the product documentation[1]. The same steps can be done programmatically (for example in C#) by resorting to a class library known as Analysis Management Objects (AMO)[2] to create, configure and run the objects listed above. It is also possible to write a set of DMX (Data Mining Extensions) queries for the same purpose. The DMX language[3] is an extension to SQL that can be used to create and query mining models.

A key parameter to define when applying MSC (step 6) is the number of clusters to use, and this can be set either manually or automatically. In the later case, the algorithm will make use of heuristics to find the ideal number of clusters for the given data. On the other hand, even if the number of clusters is specified manually, the MSC algorithm may still increase or decrease this number slightly according to other parameters such as *minimum support*, i.e. the minimum number of sequences to be placed in each cluster.

3. DATA PREPROCESSING

The MSC algorithm described above relies on a database as its source of data and therefore it can deal with very large event logs. On the other hand, the algorithm is robust to noise in the sense that the probabilistic model associated with each cluster can accommodate several variations of the same sequence. It should be noted, however, that every given sequence will eventually be assigned to one of the available clusters. This means that if a sequence is very different or atypical and hardly fits any cluster, it will nevertheless be assigned to one of them. This, in turn, will have an effect on the probabilistic model

[1] A tutorial covering these steps is available at: http://msdn.microsoft.com/en-us/library/ms167167.aspx
[2] The documentation for AMO is available at: http://msdn.microsoft.com/en-us/library/ms124924.aspx
[3] A reference for DMX is available at: http://msdn.microsoft.com/en-us/library/ms132058.aspx

estimated for that cluster. The effect is that an unusual sequence may actually distort the cluster model when, without that sequence, the cluster model would be simpler and easier to understand.

For the algorithm it does not matter what the input sequences are, but for the end user or business analyst who will interpret the results, it will be easier to draw conclusions if the cluster models are a meaningful representation of the typical sequences contained in that cluster. Therefore, when preparing the case and sequence tables for MSC, some preprocessing steps must be performed in order to ensure that behavioral patterns that are hidden in the input data will be more easily discovered. In practical applications such as the case studies presented ahead, the following preprocessing steps are usually performed:

1. *Dropping states with low support* – While some states may occur very often and be present in most of the sequences in the dataset, on the other hand there may be states that are so infrequent that their occurrence can only be attributed to pure ad-hoc behavior or even mislabeling. For the purpose of identifying typical behavior, these very infrequent states are usually removed from the input sequences.
   Example: If state "B" is found to occur very rarely in the input dataset, then the input sequence A→B→C→D turns into A→C→D.
2. *Dropping consecutive repetitions of the same state* – Some systems record multiple events caused by changes in attributes other than state. These changes may or may not be interesting to study under a control-flow point of view. In general, only events that pertain to changes in state are considered, and therefore events that do not change this state are usually discarded
   Example: the sequence A→C→C→D becomes A→C→D.
3. *Dropping single-step sequences* – Some cases actually contain no sequential behavior, as they comprise only a single step. These cases are usually removed from the dataset.
   Example: "A→B" is a sequence, but sequences with a single state, such as "A" or "B", are discarded.
4. *Dropping unique sequences* – Often there are sequences that are unique in the sense that they never happen twice. If after all the previous preprocessing steps there are still such sequences, they should be considered for removal, unless the set of all unique sequences is a significant portion of the whole input data. In general, unique sequences are undesirable as the MSC algorithm is forced to assign them to some cluster, possibly changing the cluster models in an unpredictable way.
   Example: if B→A→D→C is a sequence that happens just once in the entire dataset, then it is discarded.

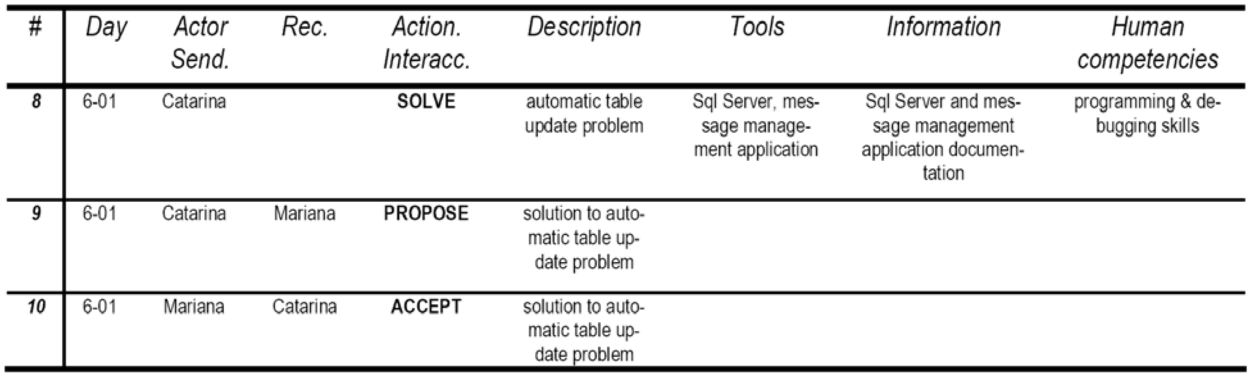
Clearly, the order of these preprocessing steps does matter. By removing some rare states (step 1) there may be more consecutive repetitions in the remaining states (step 2); for example, if B is found to be rare state in step 1 and is removed from A→C→B→C→D, then the resulting sequence will be A→C→C→D and then step 2 will be applied to produce A→C→D, something that would not happen if the two steps be applied in reverse order. Also, removing states (steps 1 and 2) may increase the number of single-step sequences (step 3); consider for example the sequence A→B with B as a rare state, or the repetitive sequence A→A→A, both of which will be collapsed to a single state. Finally, unique sequences should not be dropped until the very end (step 4) as previous steps may produce identical sequences. For example, the sequence A→B→C→D becomes identical to A→C→D after step 1 if B is found to be a rare state; however, it could be the case that A→C→D was a unique sequence before step 1, which would have been dropped had step 4 been applied immediately.

4. CASE STUDY: MINING HUMAN ACTIVITIES

This case study is based on the work (Zacarias et al, 2006) in a banking institution, where a software development team was observed for three weeks. The team comprises four software developers and a project leader that performs both system development and project management tasks. The team develops web applications and performs systems analysis, design, programming, test and maintenance activities. During the three-week period of observation, the team performed tasks on the following applications: (1) Suppliers, (2) Claims, (3) Customer Correspondence (called Mail application), (4) Evictions and (5) Marketing Campaigns.

The purpose of the study was to collect a record of all activities taking place in daily work of that team. Those data would then be used to analyze the structure of work within the team and to devise a set of collaboration tools. The team members were asked to manually register their actions and interactions in chronological order during the observation period. To reduce the burden of such task, they were asked to register their actions and interactions by means of a simple summarizing sentence.

After the data have been collected, these sentences were first parsed using grammatical rules to separate the subject and predicate. Synonym verbs were replaced by a single verb to avoid inconsistencies. Each action description was augmented with a set of application, information and human resources involved. The results were further structured as described in (Zacarias et al, 2005) into an event table as shown in figure 3. The table had 534 entries.



#	Day	Actor Send.	Rec.	Action. Interacc.	Description	Tools	Information	Human competencies
8	6-01	Catarina		**SOLVE**	automatic table update problem	Sql Server, message management application	Sql Server and message management application documentation	programming & debugging skills
9	6-01	Catarina	Mariana	**PROPOSE**	solution to automatic table update problem			
10	6-01	Mariana	Catarina	**ACCEPT**	solution to automatic table update problem			

Figure 3. Example of actions collected during observation

By analyzing this table it was possible to group events that belong to the same or to intimately related tasks. Given the chronological order of events for each team member, together with the interactions that took place between them, it was possible to determine the sequences of events that took place across actors. This led to a number of rather long sequences, which were then broken down into shorter, scope-delimited tasks. About 140 tasks were found.

A brief analysis of these task sequences revealed some issues. A first issue was that some of these tasks were not actually sequences, but just arbitrary repetitions of the same action. For example, all team members had at least one sequence in which they repeated the action “program” from 2 to 20 times. Therefore, consecutive repeating steps within each sequence were eliminated, and sequences ending up with just one single step were discarded. Figure 4 shows the total number of occurrences of each action, both before and after repeating steps were eliminated.

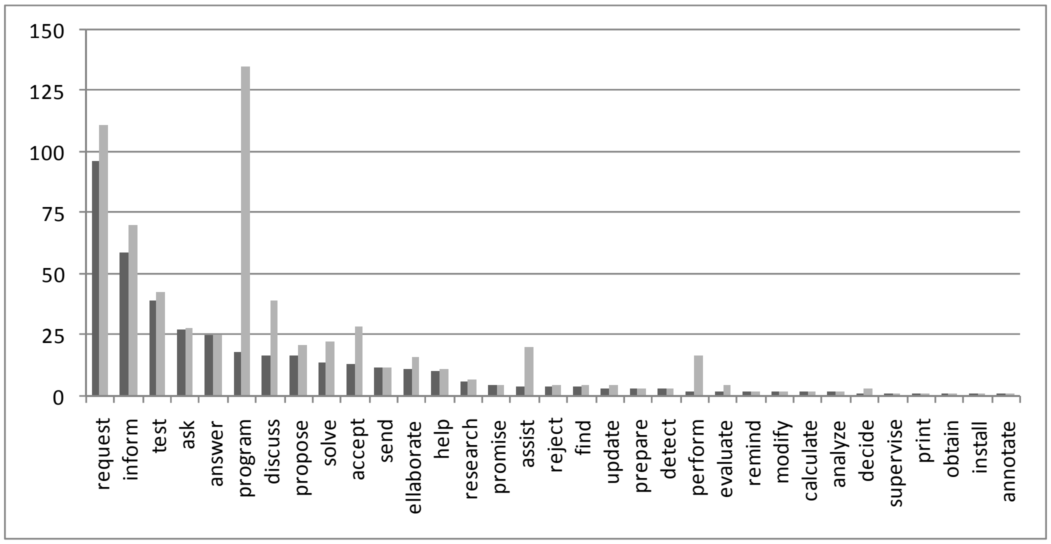


Figure 4. Total number of occurrences for each action, before (light column) and after eliminating repeating steps (dark column) ordered by decreasing number of the latter.

A second issue was that the relatively high number of different actions. This led to a set of very dissimilar sequences, despite the fact that most of them shared a limited set of common actions. For example, most tasks involve some form of “request”, whereas the action “annotate” happened only once in the entire study. This suggests that the emphasis should be put on highly recurrent actions, which provide the basic structure for most sequences. The least recurrent actions (in the tail of figure 4) represent ad-hoc variations that provide no real insight into the typical behavior. The last preprocessing stage was therefore to decide on a threshold for the number of occurrences; only actions above that threshold were allowed to remain in the sequences.

The case and sequence tables can then be built from the results of these preprocessing stages, and provided to the MSC algorithm for processing. In order to present and discuss a complete result set, here we will restrict our analysis to only the first five actions in figure 4 (i.e. “request”, “inform”, “test”, “ask” and “answer”). As a consequence, the sequences will also be rather short; there were 64 sequences with an average number of four events per sequence. Figure 5 shows the results of applying MSC to the input sequences. The sequences have been grouped into five clusters.

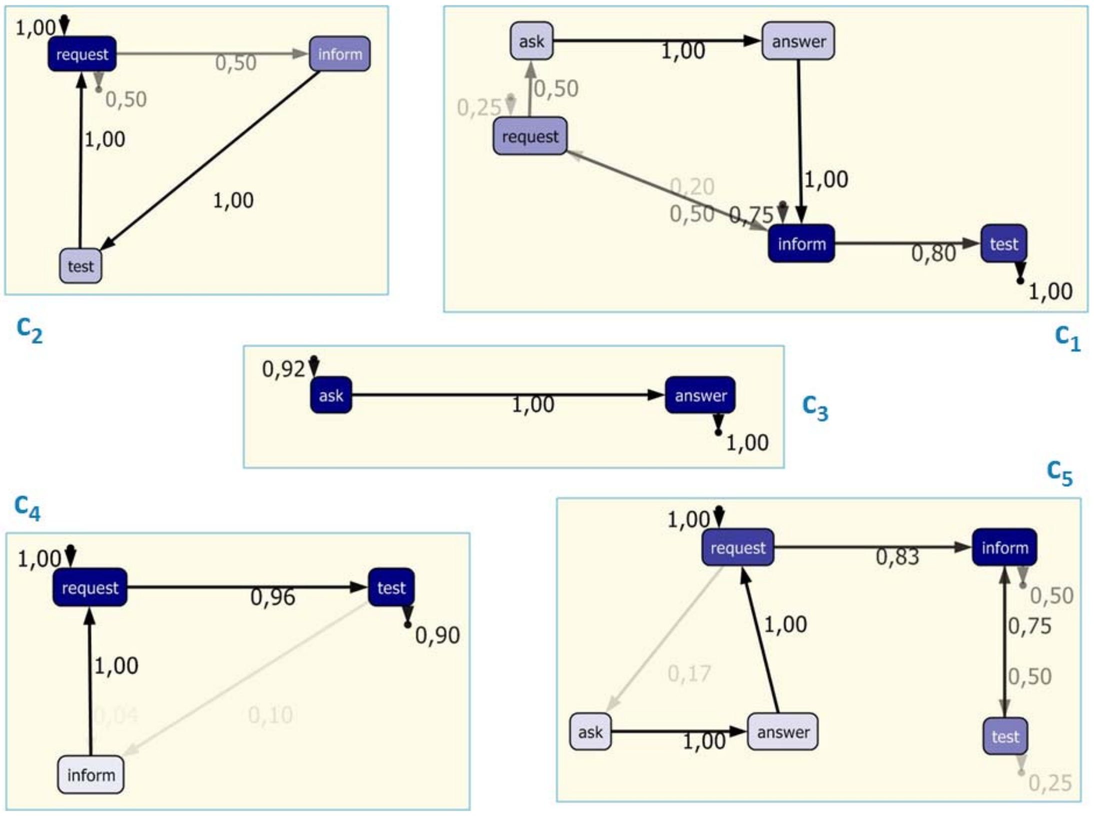


Figure 5. Clusters models for a subset of actions

Despite using a limited set of actions, it is still possible to interpret these results in terms of the activities performed by the software team. In fact, it was only when the team was shown such kind of results that further insight into their activities emerged. From these results it is possible to identify the following:

- Clusters 1 and 4 concern software integration tests. The tests can be performed either upon explicit request or depending on the result of previous tests. Clusters 4 and 1 capture these two scenarios, respectively. Cluster 4 represents tests performed upon request, where sequences are typically in the form “request-test”. Cluster 1 represents tests performed upon result of previous tests, where sequences have the form “inform-test”. In this case, the typical sequence should include a third state to become “analyze-inform-test”. However, the action “analyze” was not recorded since it is usually performed by an individual that was not observed in this study.
- Clusters 2 and 5 concern software publishing activities. These include both forms, either “request-inform-test-request” (cluster 2) or “request-inform-test-inform” (cluster 5). This behavior should include an additional state to become “request-publish-inform-test-request” or “request-publish-inform-test-inform”. However, the action “publish” is also performed by an unobserved member.
- Cluster 3 contains common behavior in the form “ask-answer”. This behavior occurs in several tasks and has to do mainly with team members helping each other. It also appears in particular contexts such as those of clusters 1 and 5.

In the case of integration tests and publishing activities, it is remarkable that it is possible to distinguish between these activities even though key actions such as “analyze” and “publish” are missing. On one hand, this suggests that sequences of actions that belong to different processes have a distinct signature, i.e. they have a basic sequential structure that can be distinguished even if some key states are missing; sequence clustering is a very useful technique to separate inherently different sequences which would otherwise seem similar or impossible to distinguish in the midst of a large event log. On the other hand, only a business analyst will be able to look at the clustering results and recognize the business activities being depicted in those results. This is especially true in practical applications where the users or systems being observed can provide only an incomplete view of the business processes within an organization. The kind of analysis that can be done via sequence clustering, or for that matter any other technique, is bound by the breadth and relevance of the input data that can be collected in the first place.

5. CASE STUDY: CONFORMANCE OF AN ISSUE HANDLING PROCESS

This case study involves a medium-sized IT company whose main product is an advanced software platform designed to accelerate the development of custom business applications. Using this platform, even complex applications can be developed in a graphical way without programming, and then deployed to a Web-based run-time environment. The platform is being improved continuously by successive release versions that add new functionality, improve existing features, and correct bugs. Besides extensive manual and automated in-house testing, end users also have an active role in pointing out desired improvements and problems to be solved.

Figure 6 illustrates the case study scenario, where customers report issues to the technical support team. To keep track of all reported issues and to handle them appropriately, the company developed a custom solution using its own software platform. The system is called Issue Manager and it is basically a database with a Web-based interface. The database stores information about each issue such as date, description, submitter, status, priority, risk, severity, etc., along with the product version where the issue was detected, as well as possible relationships to other issues. Most of these data can be filled with whatever the support team finds appropriate, except for the status field which is allowed to have one of a limited set of possible states.

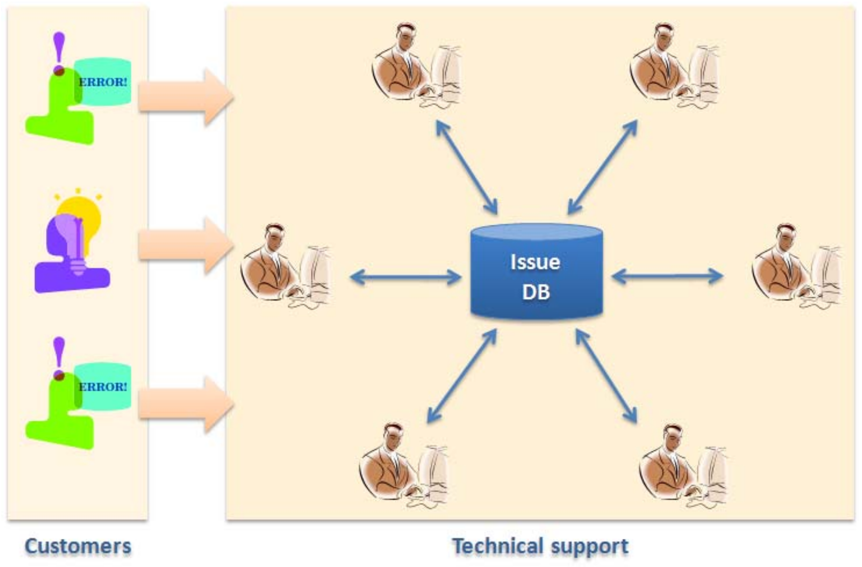


Figure 6. Case study scenario

Figure 7 illustrates the typical set of states that an incident goes through. Regardless of the channel an issue comes from, it will be recorded in the system as “New”. Then someone will look at it and check whether it is a duplicate issue, whether it is relevant, what priority level it should be assigned, whether there is enough information for the issue to be handled, whether there are other issues that could be related to this one, etc. In some cases, the issue may end up being “Discarded” if it is a non issue; for example, it could be just a user mistake. In other cases it may be labeled as “Duplicated” if a similar issue is known and has been already recorded in the database.

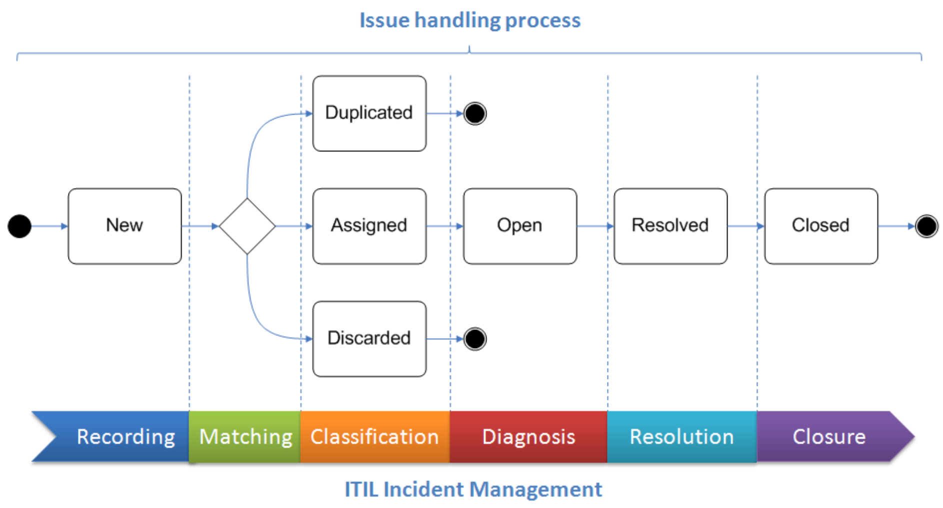


Figure 7. The issue handling process within the framework of ITIL

In most cases, issues will follow the regular handling process. The issue may be “Assigned” either to a specific person, or collectively to the support team. The state will be changed to “Open” when someone is actively working on the issue. It will then be a matter of time and effort until a solution or at least a workaround is found; at this point the issue becomes “Resolved”. A few issues may end up in a “Not Resolved” state but this result is, in general, not to be expected. Issues are automatically “Closed” when a new product version that includes the solution is released.

The process just described clearly resembles ITIL Incident Management. The Information Technology Infrastructure Library (ITIL) (van Bon et al, 2005) defines a set of standard best practices for IT service management, ranging several processes, and Incident Management is the ITIL process that focuses on the handling of events that disturb normal service operation. ITIL Incident Management defines the following steps to handle those events:

1. Recording: upon reception, the incident must be recorded.
2. Classification: the incident is characterized in terms of type, impact and urgency, leading to a certain priority class.
3. Matching: a solution may already exist if the incident matches a known problem or error condition.
4. Diagnosis: all available information about the incident is gathered in order to investigate and determine a solution or workaround.
5. Resolution: the solution is applied in order to restore normal service or system operation.
6. Closure: the incident is closed once the service has been restored.

During incident diagnosis, successive levels of service support may be invoked until a solution or workaround is found. This behavior is known as *escalation* – if the current support level is unable to find a solution, then the incident escalates to the next (higher) support level.

As suggested in figure 7, the issue handling process can be mapped directly to the structure of ITIL Incident Management: recording, classification, matching, diagnosis, resolution and closure are all present. Classification is being done between "New" and "Assigned"; diagnosis takes place when the issue is "Open"; resolution and closure are represented by appropriate states as well. Issue handling is, in itself, a process with all the characteristics of Incident Management.

Despite the fact that the issue handling process is clearly defined, the model depicted in figure 7 provides only an indication of the sequence of states that an issue should go through. There is, in practice, no restriction being placed on the particular state of an issue, nor on the transition to other states. Members of the technical support team are free to use these or other states, and to change the issue state in any way as they see fit. The question now is to determine how far the behavior recorded in the system database actually complies with the original process depicted in figure 7. This is a problem of conformance checking (Rozinat & Aalst, 2008).

In the database it was found that an issue can actually be in one of 15 possible states, in alphabetical order: "Approved", "Assigned", "Closed", "Discarded", "Duplicated", "Needs Approval", "Needs Specification", "Needs Verification", "New", "Not Approved", "Not Resolved", "Open", "Postponed", "Resolved", and "Waiting". The semantics of some of these states are not entirely clear, although their names provide some indication of their meaning. It was also found that the database contains many interrelated tables that aim at supporting a wide range of functionalities. An analysis of both the database schema and content revealed that there were two tables of interest to collect the state sequences:

- Table *issue* – contains general information about an issue such as a unique identifier, name, description, product version and date of submission, but also about the priority, severity, present state and who is currently assigned to handle the issue. There were 14 982 issues in the database.
- Table *history* – keeps track of all events where an event is a change of some sort, including change of assignment, change of priority, and change in state. There were 143 220 events in the database, roughly ten times as much as the number of issues.

With the data contained in the history table it was possible to build a useful dataset for analysis. Basically, each sequence corresponds to the time-ordered list of state changes recorded for a given issue. Figure 8 shows that sequence length varies widely, from issues with a single recorded event to issues with over 50 events. In fact, the longest sequence had 75 events, most of which were just a repetition of the "Waiting" state. Figure 8 also shows that most issues had sequence lengths between 1 and 15.

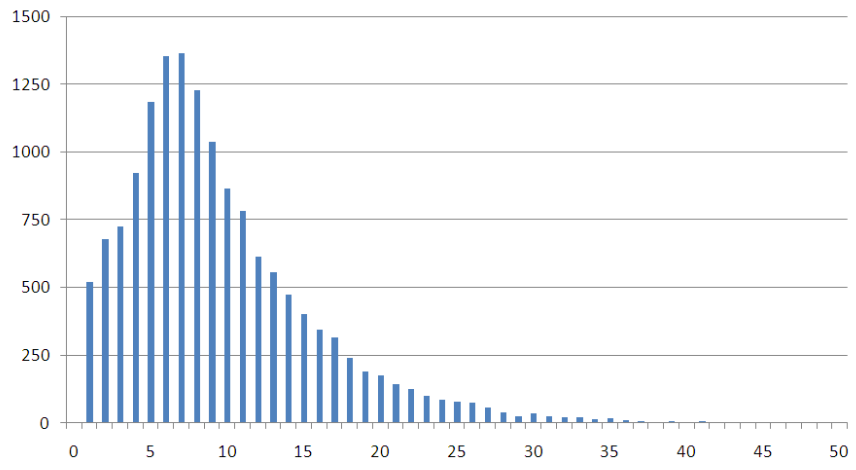


Figure 8. Number of issues vs. sequence length found in the history table

The fact that the system allowed any kind of change to be freely made to an issue means that the sequences displayed an arbitrary repetition of states when the changes were being made to fields other than state. For this reason, the sequence length was often longer than it would have been obtained if only the change in the state would be considered. These and other preprocessing steps had to be done before applying MSC to the dataset. The following preprocessing steps were used:

1. *Dropping events with low support* – Figure 9 shows the number of occurrences of each state in the history table. The states in the bottom of figure 9 have low support since they occur only very rarely. However, in this case study all states have been kept.
2. *Dropping consecutively repeated events* – since many consecutive events were created by changes to fields other than state, they could be considered as a single event for our purposes. Around 63% of all events were eliminated in this step. The average sequence length also decreased dramatically, and there was a significant increase in the number of sequences with length between 1 and 5.
3. *Dropping sequences with either insufficient or excessive length* – Figure 1 shows that many sequences are actually non-sequences as they comprise a single event, so these sequences were removed. About 1000 sequences were eliminated in this step.
4. *Dropping sequences with repeated events* – a sequence that contains a (non-consecutive) repetition in a state represents a case where the handling of an issue had to recede to a previous state. Sequences with such repetitions display a mixture of behavior, which makes them difficult to assign correctly to a single cluster. About 2500 sequences were eliminated in this step.
5. *Dropping unique, one-of-a-kind sequences* – sequences that are unrepeatable are not interesting for the purpose of identifying typical behavior. About 300 unique sequences were removed from the dataset.

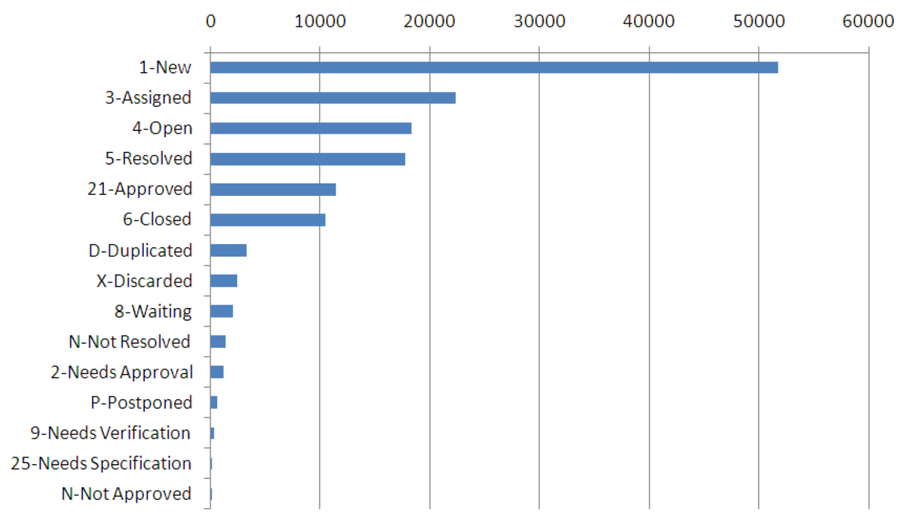


Figure 9. Number of occurrences of each state in the history table

After these preprocessing steps 11 085 sequences remained, with a total of 35 778 events. Figure 10 shows the most frequent sequences after preprocessing of the input dataset. Although some of the expected behavior is immediately recognizable at the top of the figure, it should be noted that the most frequent sequence accounts for only about 10% of the total number of sequences. The total of 11 085 sequences includes 264 different sequences, of which only the top twenty are shown in figure 10. We now turn to the application of sequence clustering to this dataset.

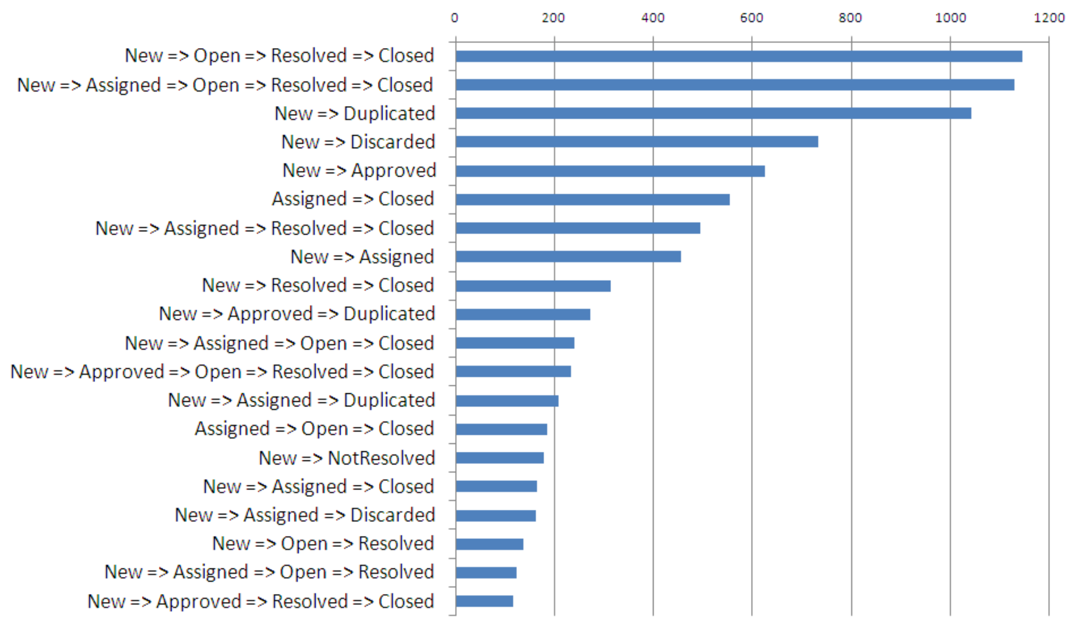


Figure 10. The most frequent sequences after preprocessing (top 20)

Judging by the kind of sequences found in the input dataset, a number of about 12 clusters seemed to be a good initial guess. After setting the parameters and running MSC on the dataset, 14 clusters were created. Figure 11 shows the top three most frequent sequences in each cluster. For cluster 14 only one sequence is shown, since this cluster has only one kind of sequence.

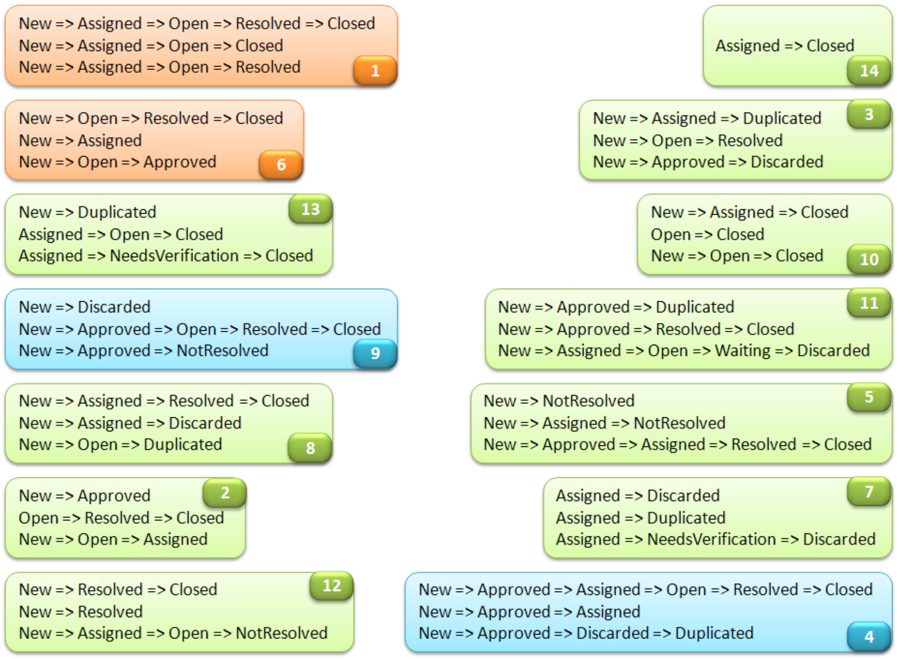
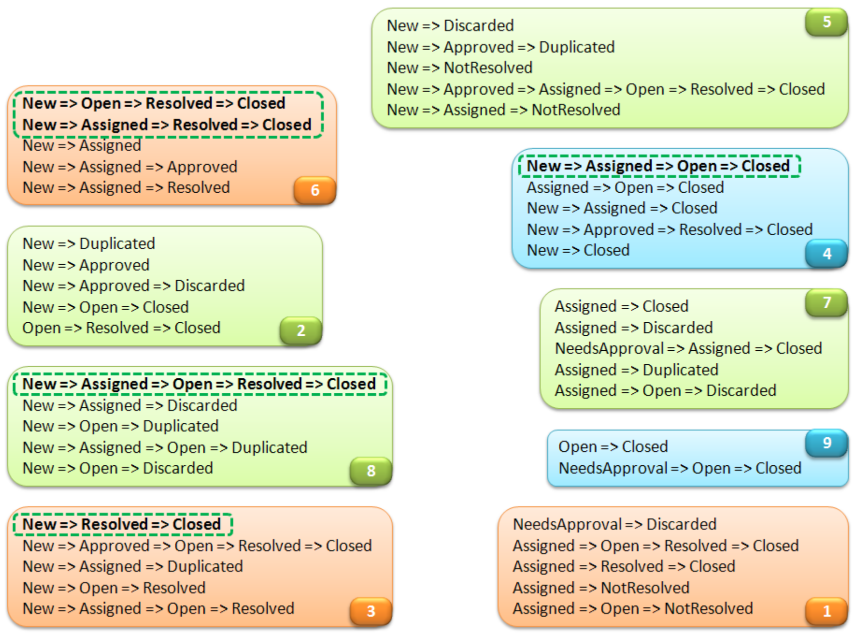

New => Assigned => Open => Resolved => Closed
New => Assigned => Open => Closed
New => Assigned => Open => Resolved
1
Assigned => Closed
14
New => Open => Resolved => Closed
New => Assigned
New => Open => Approved
6
3
New => Assigned => Duplicated
New => Open => Resolved
New => Approved => Discarded
13
New => Duplicated
Assigned => Open => Closed
Assigned => NeedsVerification => Closed
New => Assigned => Closed
Open => Closed
New => Open => Closed
10
New => Discarded
New => Approved => Open => Resolved => Closed
New => Approved => NotResolved
9
11
New => Approved => Duplicated
New => Approved => Resolved => Closed
New => Assigned => Open => Waiting => Discarded
New => Assigned => Resolved => Closed
New => Assigned => Discarded
New => Open => Duplicated
8
5
New => NotResolved
New => Assigned => NotResolved
New => Approved => Assigned => Resolved => Closed
2
New => Approved
Open => Resolved => Closed
New => Open => Assigned
7
Assigned => Discarded
Assigned => Duplicated
Assigned => NeedsVerification => Discarded
12
New => Resolved => Closed
New => Resolved
New => Assigned => Open => NotResolved
New => Approved => Assigned => Open => Resolved => Closed
New => Approved => Assigned
New => Approved => Discarded => Duplicated
4

Figure 11. Top sequences for 14 clusters

Some of these clusters display very similar behavior. For example, clusters 1 and 6 have sequences that could have been probably included in the same cluster. The same happens with clusters 4 and 9, and other clusters as well. The presence of similar sequences in different clusters suggests that the number of clusters should be decreased.

Running MSC again with different parameter settings, nine clusters were obtained. Figure 12 shows the most frequent sequences in each of these nine clusters. Again, cluster 9 shows fewer sequences because it has only two types of sequences. The top sequences in clusters 3, 4, 6 and 8 are clearly related, and other sequences within different clusters were also found to be similar. These results suggested that the number of clusters should be decreased even further.



New => Discarded
New => Approved => Duplicated
New => NotResolved
New => Approved => Assigned => Open => Resolved => Closed
New => Assigned => NotResolved
5

New => Open => Resolved => Closed
New => Assigned => Resolved => Closed
New => Assigned
New => Assigned => Approved
New => Assigned => Resolved
6

New => Assigned => Open => Closed
Assigned => Open => Closed
New => Assigned => Closed
New => Approved => Resolved => Closed
New => Closed
4

New => Duplicated
New => Approved
New => Approved => Discarded
New => Open => Closed
Open => Resolved => Closed
2

Assigned => Closed
Assigned => Discarded
NeedsApproval => Assigned => Closed
Assigned => Duplicated
Assigned => Open => Discarded
7

New => Assigned => Open => Resolved => Closed
New => Assigned => Discarded
New => Open => Duplicated
New => Assigned => Open => Duplicated
New => Open => Discarded
8

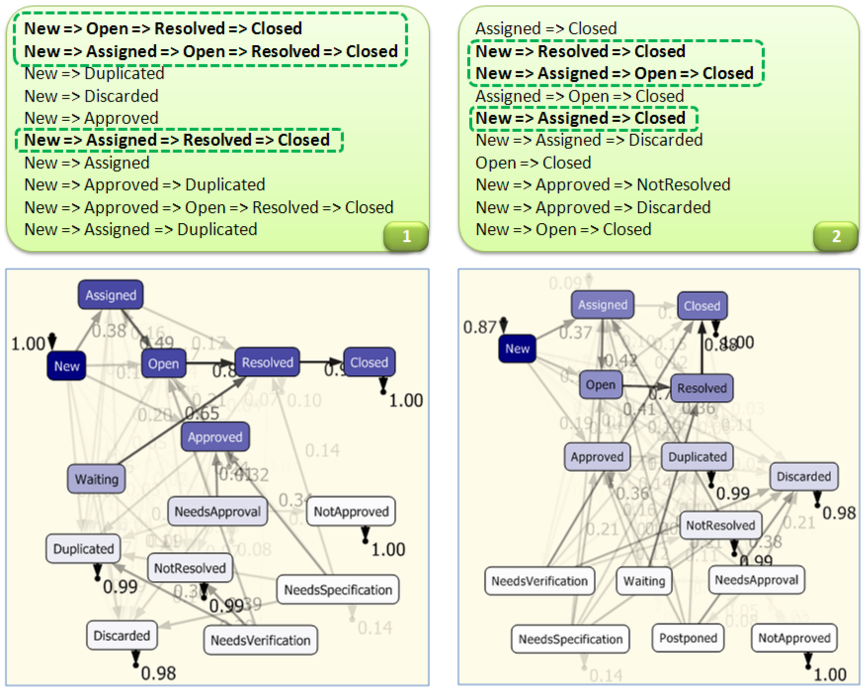
Open => Closed
NeedsApproval => Open => Closed
9

New => Resolved => Closed
New => Approved => Open => Resolved => Closed
New => Assigned => Duplicated
New => Open => Resolved
New => Assigned => Open => Resolved
3

NeedsApproval => Discarded
Assigned => Open => Resolved => Closed
Assigned => Resolved => Closed
Assigned => NotResolved
Assigned => Open => NotResolved
1

Figure 12. Top sequences for 9 clusters

By setting the number of clusters to automatic, a surprising result emerged as the MSC algorithm produced only two clusters. Figure 13 shows the most frequent sequences for each cluster together with the corresponding cluster model. However, none of these models display clearly distinct behaviors. There are still similar sequences across the two clusters, and the cluster models do not help in establishing any meaningful difference. And yet, the dataset does contain very different sequences, as can be seen by manual inspection. These results suggest that the observed behavior, despite being quite heterogeneous, is evenly distributed in such a way that it is difficult to identify clearly distinct patterns.



New => Open => Resolved => Closed
New => Assigned => Open => Resolved => Closed
New => Duplicated
New => Discarded
New => Approved
New => Assigned => Resolved => Closed
New => Assigned
New => Approved => Duplicated
New => Approved => Open => Resolved => Closed
New => Assigned => Duplicated
1

Assigned => Closed
New => Resolved => Closed
New => Assigned => Open => Closed
Assigned => Open => Closed
New => Assigned => Closed
New => Assigned => Discarded
Open => Closed
New => Approved => NotResolved
New => Approved => Discarded
New => Open => Closed
2

Figure 13. Top sequences and cluster models for 2 clusters

We therefore turn to the study of the input dataset as a whole. If behavior cannot be separated into different clusters, then a single global model should suffice to identify typical behavior. Figure 14 depicts such model. Rather than transition probabilities, the actual state and transition counts are shown, providing an indication of the most frequent states as well as the most occurring transitions in absolute terms. Also, the node shading and line weight were made proportional to the state and transition counts, respectively. It is easy to see, for example, that “New” is the most recurring state, and that in most sequences the following state is “Assigned”. However, some care must be taken when drawing conclusions about the most common sequences, as subsequent transitions may refer to different cases.

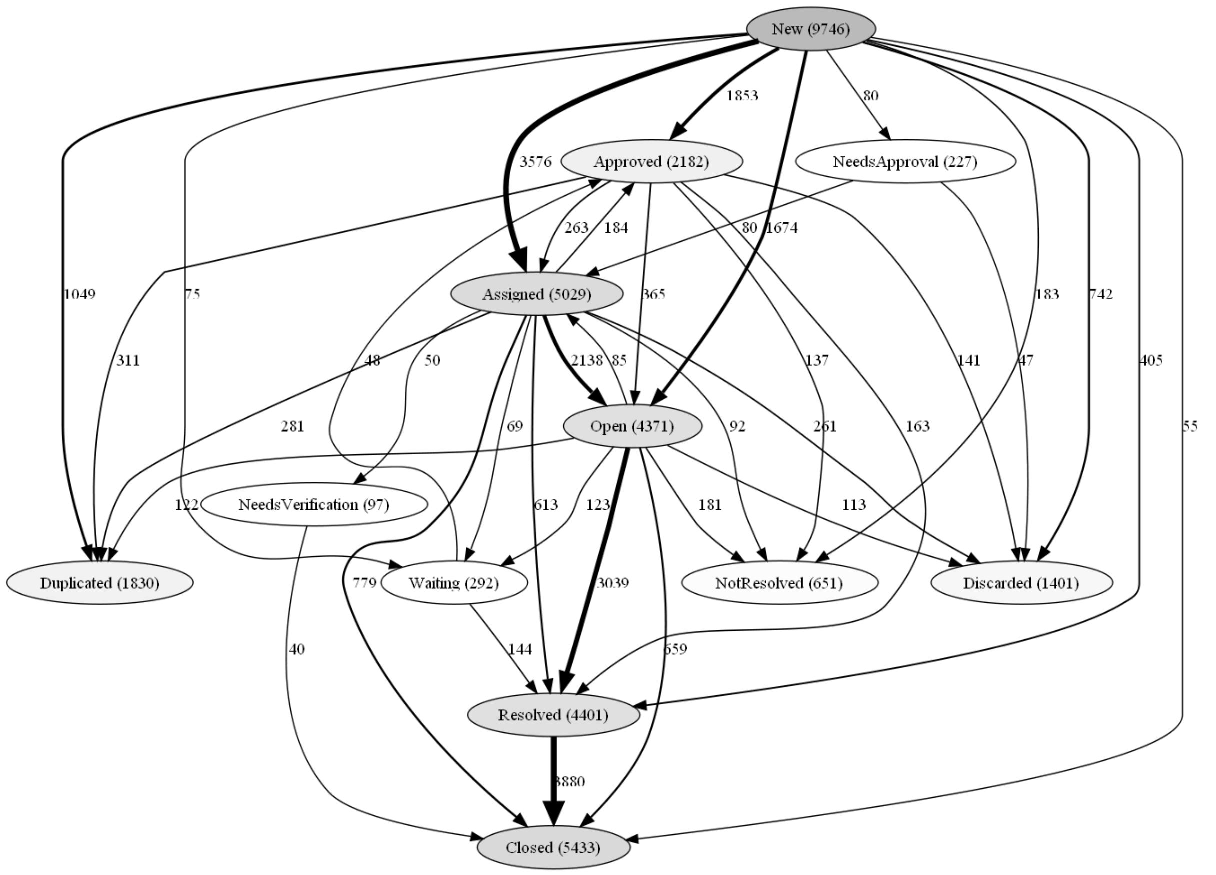


Figure 14. Global behavioral model for the preprocessed dataset (only states with total count above 95 and transitions with total count above 35 are shown)

Looking up “New”, “Assigned”, “Open”, “Resolved”, “Closed”, “Duplicated” and “Discarded” in figure 14 reveals that the overall behavior in the input dataset actually resembles the original process depicted in figure 7. The main trend “New→Assigned→Open→Resolved→Closed” is clearly distinguishable in figure 14, and the alternative branches are also apparent: “New→Duplicated” and “New→Discarded”.

On the other hand, figure 14 displays a lot of extra behavior that figure 7 is unable to account for. When presenting the results to the company, some of this extra behavior was explained by business analysts as follows:

- About the transition: “New→Approved” – Some states are no longer being used. For example, in the past it was common to make new issues go through an approval process, and some of that behavior is still present in the database, as can be seen in Figure 14 in the transition from “New” to “Approved”. Nowadays, that approval is implicit when the issue changes from “New” to “Assigned”.
- About the transitions: “New→Open”, “New→Resolved”, “New→Closed” – The support team members usually skip steps when the solution to the issue is obvious. For example, the team member who registers a new issue may immediately recognize the problem and solve it, jumping directly to the “Resolved” state.
- About the transition: “Open→Assigned” – The state transitions may appear to be reversed as the result of arbitrary loops. For example, an issue may be assigned to and opened by a team member, just to find that it should have been assigned to someone else; in this case, a transition from “Open” to “Assigned” will be recorded. This kind of backtrack is also allowed by ITIL when there is escalation to a higher support level.
- About the transitions: “Assigned→Duplicated”, “Assigned→Discarded”, “Open→Duplicated”, and “Open→Discarded” – The classification of an issue as a duplicate or the decision to discard it may come later in the process when more data about the issue has been collected or provided by the customer.

These special but relatively frequent cases explain most of the extra behavior shown in Figure 14. Overall, the event history recorded in the system database suggests that the issue handling process is being carried out in a way that is close to the originally intended process. The results of this study can now be used by business analysts to derive performance metrics, to investigate the causes of unusual behavior, or to devise new practices for the technical support team, to cite only a few of the potential benefits.

6. CONCLUSION

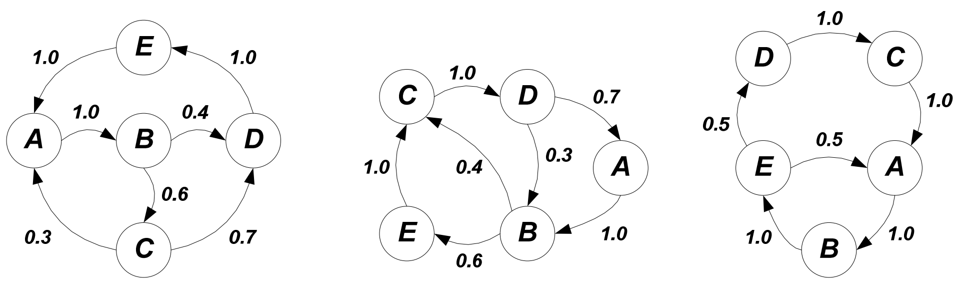
In both case studies, sequence clustering provided a useful result. Whereas in the first case study it showed that the work of the software team is structured according to a set of patterns, in the second case study the absence of distinct clusters suggests that the technical support team is indeed following the expected behavior for the issue handling process. This means that sequence clustering can become a useful technique both in the context of process discovery and in the context of process conformance.

What makes sequence clustering particularly attractive is its robustness to noise and its ability to deal with very large amounts of data. However, in practice it is often the case that some preprocessing must be applied before running the algorithm on the input dataset. This preprocessing is intended to build the input dataset in a way that the algorithm will produce results that can be easily interpreted by a business analyst. Allowing the presence of behavior that is known to be inadequate for processing will only make it more difficult to identify typical behavioral patterns.

In conclusion, sequence clustering with appropriate preprocessing is a powerful technique to acquire insight into the underlying structure of business processes. It can be used as a first approach to process mining, when the event log is large and must be partitioned into different behaviors. It is also useful when the presence of ad-hoc behavior makes it impossible for automated processing by deterministic algorithms. In these and other scenarios, sequence clustering becomes a valuable tool in the repertoire of available process mining techniques.

EXERCISES

1. Consider the following three cluster models, where states are represented by a single letter and arcs are labeled with the transition probabilities between states. Assume that a sequence may begin or end in any state.



Cluster 1 **Cluster 2** **Cluster 3**

For each of the following sequences, which cluster should it be assigned to? Why?
(a) ABCD (b) CABE (c) EAB (d) BCDE (e) ABE (f) BCDB

2. Suppose that a cluster has been assigned the following four sequences: XWY, ZYXW, XWZY, and ZYXZ, where each letter stands for a state.
(a) Draw the cluster model and determine the transition probabilities.
(b) List three sequences of length 4 that are allowed by the cluster model but are not present in the original dataset.

3. Consider the following input dataset: {GBB, AAH, AEHEBD, AEBBD, CCFCB, CFGFCBB, BBCD, BCDDD, DEEAC, AABCCD}
With respect to the four preprocessing steps described in section 3:
(a) Apply step 1 assuming that any state that does not occur at least 3 times can be removed.
(b) Apply step 2 on the previous results.
(c) Apply step 3 on the previous results.
(d) Apply step 4 on the previous results.
List the sequences in the preprocessed dataset.

4. Typically, the first step in preprocessing is to discard events with low support.
(a) In the first case study, only the first five actions in figure 4 have been used to produce the results. Explain how the results would be affected if only the first three actions would have been used.
(b) In the second case study, all the states in figure 9 have been used. Which states should be kept and which states could be discarded? Why? What would be the disadvantage of discarding those states?

SUGGESTED ADDITIONAL READING

For an introduction to the topic of model-based clustering and other data mining techniques, the book of (Han & Kamber, 2006) may serve as a general reference. A more formal account of Expectation-Maximization and related techniques can be found in (McLachlan & Krishnan, 1996). For sequence clustering in particular, the paper by (Cadez et al, 2003) describes the algorithm in detail together with an application to the discovery of navigation patterns on a Web site. The book of (Tang & MacLennan, 2005) as well as the online tutorials (Microsoft, 2007) explain how to use the MSC algorithm available in Microsoft SQL Server 2005® Analysis Services. The application of this algorithm in the context of process mining was first described in (Ferreira et al, 2007). For more information on the case studies presented here, we refer the reader to (Zacarias et al, 2005) and to (Ferreira & Mira da Silva, 2008). Regarding the problem of conformance checking of business processes, the work of (Rozinat & Aalst, 2008) is especially interesting and recommended.

REFERENCES

W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, A.J.M.M. Weijters, "Workflow Mining: A Survey of Issues and Approaches", Data and Knowledge Engineering, 47(2):237-267, 2003

W.M.P. van der Aalst, H.A. Reijers, M. Song, "Discovering Social Networks from Event Logs", Computer Supported Cooperative work, 14(6):549-593, 2005

W.M.P. van der Aalst, A.J.M.M. Weijters, "Process Mining: A Research Agenda", Computers in Industry, 53(3):231-244, 2004

W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs", IEEE Transactions on Knowledge and Data Engineering, 16(9):1128-1142, 2004

R. Agrawal, D. Gunopulos, F. Leymann, "Mining Process Models from Workflow Logs", Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology, LNCS 1377, pp.469-483, Springer, 1998

A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst, "Genetic Process Mining: An Experimental Evaluation", Data Mining and Knowledge Discovery, 14(2):245-304, 2007

A.K. Alves de Medeiros, A. Guzzo, G. Greco, W.M.P. van der Aalst, A.J.M.M. Weijters, B. van Dongen, D. Sacca, "Process Mining Based on Clustering: A Quest for Precision", Proceedings of the BPM 2007 International Workshops, LNCS 4928, Springer, 2008

J. van Bon, M. Pieper, A. van der Veen, "Foundations of IT Service Management Based on ITIL", Van Haren Publishing, 2005

I. Cadez, D. Heckerman, C. Meek, P. Smyth, S. White, "Model-Based Clustering and Visualization of Navigation Patterns on a Web Site", Data Mining and Knowledge Discovery, 7(4): 399-424, October 2003

A. Ceglowski, L. Churilov, J. Wassertheil, “Knowledge Discovery through Mining Emergency Department Data”, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS '05), 2005

Y. Chen, K. Reilly, A. Sprague, Z. Guan, “SEQOPTICS: a protein sequence clustering system”, BMC Bioinformatics, 7(Suppl 4):S10, 2006

J. Cook, A. Wolf, “Automating process discovery through event-data analysis”, Proceedings of the 17th International Conference on Software Engineering, pp.73-82, ACM Press, 1995

A. Dempster, N. Laird, D. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm”, Journal of the Royal Statistical Society, Series B, 39(1):1-38, 1977

B.F. van Dongen and W.M.P. van der Aalst, “Multi-Phase Process Mining: Building Instance Graphs”, in Proceedings of the International Conference on Conceptual Modeling (ER 2004), volume 3288 of Lecture Notes in Computer Science, pages 362-376. Springer-Verlag, Berlin, 2004

A. Enright, S. van Dongen, C. Ouzounis, “An efficient algorithm for large-scale detection of protein families”, Nucleic Acids Research, 30(7):1575-1584, 2002

D. Ferreira, M. Zacarias, M. Malheiros, P. Ferreira, “Approaching Process Mining with Sequence Clustering: Experiments and Findings”, Proceedings of the 5th International Conference on Business Process Management (BPM 2007), LNCS 4714, pp. 360-374, Springer, 2007

D. Ferreira, M. Mira da Silva, “Using Process Mining for ITIL Assessment: a Case Study with Incident Management”, Proceedings of the 13th Annual UKAIS Conference, Bournemouth University, April 2008

G. Greco, A. Guzzo , L. Pontieri, D. Saccà, “Mining Expressive Process Models by Clustering Workflow Traces”, Procedings of the 8th Pacific-Asia Conference (PAKDD 2004), LNCS 3056, Springer, 2004

G. Greco, A. Guzzo, L. Pontieri, “Mining Hierarchies of Models: From Abstract Views to Concrete Specifications”, Proceedings of the 3rd International Conference on Business Process Management (BPM 2005), LNCS 3649, Springer, 2005

J. Han, M. Kamber, “Data Mining: Concepts and Techniques”, 2nd edition, Morgan Kaufmann, 2006

J. Herbst, D. Karagiannis, “Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models”, Proceedings of the 9th International Workshop on Database and Expert Systems Applications, pp.745-752, 1998

J.-Y. Jung, J. Bae, L. Liu, “Hierarchical Business Process Clustering”, Proceedings of the IEEE International Conference on Services Computing (SCC ’08), pp.613-616, 2008

W. Li, L. Jaroszewski, A. Godzik, “Sequence clustering strategies improve remote homology recognitions while reducing search times”, Protein Engineering, 15(8): 643-649, August 2002

G. McLachlan, T. Krishnan, “The EM Algorithm and Extensions”, John Wiley & Sons, 1996

A. Rozinat, R.S. Mans, W.M.P. van der Aalst, “Mining CPN Models: Discovering Process Models with Data from Event Logs”, Proceedings of the Seventh Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2006), volume 579 of DAIMI, pages 57-76, Aarhus, Denmark, October 2006

A. Rozinat, W.M.P. van der Aalst, “Conformance Checking of Processes Based on Monitoring Real Behavior”, Information Systems, vol.33, no.1, pp.64-95, 2008

M. Song, C. Günther, W.M.P. van der Aalst, “Trace Clustering in Process Mining”, Proceedings of the 4th Workshop on Business Process Intelligence (BPI 08), Milan, Italy, September, 2008

Z. Tang, J. MacLennan, “Data Mining with SQL Server 2005”, John Wiley & Sons, 2005

M. Zacarias, A. Marques, H. Pinto, J. Tribolet, “Enhancing Collaboration Services with Business Context Models”, International Workshop on Cooperative Systems and Context, 5th International and Interdisciplinary Conference on Modeling and Using Context, July 2005

M. Zacarias, H. Pinto, J. Tribolet, “A Context-based Approach to Discover Multitasking Behavior at Work”, Proceeding of the 5th International Workshop on Task Models and Diagrams for User Interface Design, October 2006