# Process Diagnostics: a Method Based on Process Mining

Melike Bozkaya, Joost Gabriels, Jan Martijn van der Werf
LaQuSo, Laboratory for Quality Software,
an activity of Technische Universiteit Eindhoven and Radboud Universiteit Nijmegen,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Email: {m.bozkaya, j.gabriels, j.m.v.d.werf}@laquso.com

***Abstract*—As organizations change, their information systems can evolve from simple systems to complex systems, which are hard to understand, and therefore hard to maintain or extend. Process mining can help organizations in trying to understand the information systems by analyzing the system.**

**In this paper we propose a methodology to perform process diagnostics, based on process mining. Given an event log of an information system within an organization, process diagnostics gives a broad overview of the organization's process(es) within a short period of time. In the process diagnostics methodology, several perspectives of the process are highlighted. The outcome covers the control flow perspective, the performance perspective and the organizational perspective. We used the methodology on a case study for a Dutch governmental organization.**

***Index Terms*—process management, process mining, information system design, process analysis, methodology**

## I. Introduction

Today, information systems are indispensable within organizations. As organizations change, these systems evolve from simple systems that were easy to understand to complex systems that are hard to understand, and hence difficult to maintain. This raises the question whether the system is working the way the organization thinks it works. This question typically occurs in case the information system has some flaws, like performance problems. More important, for organizations to be in control, they need to ensure that the system is working according to their specification.

Information systems supporting business processes typically record all *events* in a log. Such a log, called an *event log* or *audit trail*, contains information about the events: for which activity and *process instance*, referred to as *case*, it is executed, by whom and at what time. An activity can have multiple events. All events for a case form a sequence, which is called the *trail* or the *run* of that case, and shows what events happened in what order and at what time. *Process mining* allows these event logs to be analyzed. Process mining looks "inside the process" to answer questions like: "how does the actual process look like?", "are the executed logs conform specification, i.e. follow all cases the specified process model?", "are there any bottlenecks in the process?", "who executes what tasks?", "who typically work together?", etc. The field of process mining covers many areas, like performance characteristics (e.g. throughput times) [1], process discovery (discovery of the control flow) [2], [3], [4], process conformance (is the event log conform specification) [5], [6], and social networks (e.g. cooperation or subcontracting) [7], [8]. Answers on these questions can serve as a handle for organizations to answer two of the main questions: "are we in control?", and "does the information system really reflect the state of affairs of the business process?".

One of the tools to support process mining is the process mining framework ProM [9]. It is plugin-based to support new areas and techniques. In the last decade, process mining evolved from control flow discovery to a broad area of research to get all kinds of information from a log, which resulted in more than 250 different plugins within ProM. This shows that many different techniques exist to apply process mining. Since there are so many, it is not clear anymore when to use which plugin. Although many case studies, see e.g. [10], [11], have been performed, the main problem with these case studies is that they were all done case-by-case on the insights and knowledge of the researcher performing the case study. Up-to-now, there is no methodology to tackle a new case study, i.e. it is hard to make process mining a repeatable service.

In this paper we propose a methodology to make a diagnostics of a process based on process mining. The methodology is designed to deliver within a short period of time a broad overview of the process(es) within the information system. Key in this methodology is the absence of prior and domain specific knowledge. The only information available is the event log. This also implies that the diagnostics only presents facts about the process. It is up to the organization to interpret the outcome of the diagnostics. We illustrate the methodology with a case study performed for a Dutch governmental organization.

This paper is organized as follows: first we explain the methodology in Section II. In Section III, we present a case study in which we applied the methodology. We conclude this paper in Section IV.

## II. The Process Diagnostics Method

We present a methodology for quick process diagnostics. The methodology aims at giving a broad overview of the process in the information system within a short period of time. It consists of five phases: (1) log preparation, in which the event log of the information system is extracted, (2) log inspection, to get a first glance of the process, (3) control flow analysis, (4) performance analysis and (5) role analysis, i.e. the

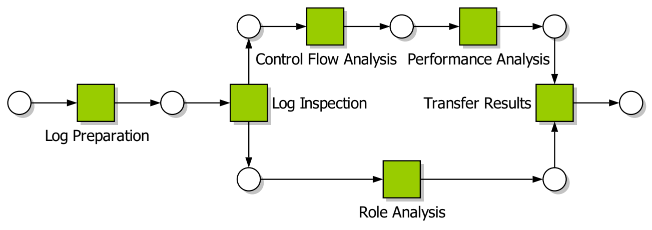

Fig. 1. The phases of the methodology

persons and resources that execute activities in the process. Finally, the results are reported to the client. Figure 1 shows the methodology as a process. The first two stages are input for both control flow analysis and role analysis, performance analysis also needs the input of control flow analysis.

### A. Log Preparation

In an event log, each event refers to a case and to an activity, and has a time stamp indicating when it occurred. Typically, the information system in question has an internal log format, which contains all information, but this needs to be extracted, or even transformed, in order to be able to use it for process mining. The information system's internal format of logs can anything from plain text files to internal databases, like in an SAP system. Therefore, preprocessing of the log is needed. Preprocessing already raises many questions. The first step is to select the best notion of a case, since often there are several candidates. The next step is the identification of activities and their events. If the log has multiple time stamps, the semantics of the time stamps needs to be clear, e.g. is it a time stamp of the start of the event, or of the case. These kind of problems need to be tackled in order to get a proper event log to proceed to the next phase.

### B. Log Inspection

The next phase is to get familiar with the event log, and to get a first glance of the process represented by the event log. In this phase, statistics about the log are gathered. This includes information about the number of cases and roles, the total number of events, the different events present, the minimal, maximal and average number of events per case, the found start and end events and their occurrences, etc. These statistics give insights in the size of the process and event log, and helps to tune mining algorithms and to evaluate the results obtained in next phases. Based on these statistics, the event log is filtered to remove *incomplete cases*, i.e. cases that were started before the start of the event log, and cases that were still running at the end of the event log are removed. This filtered event log is input for the next phases.

### C. Control Flow Analysis

In this phase, the control flow aspect of the process is analyzed. It gives answers to the question "what does the actual process look like?". First, if the organization has a process description, a conformance check is executed to check whether the process is conform specification, i.e. that each case in the event log can be replayed in the process definition [5]. If either this is not the case, or a process description does not exist, the control flow needs to be discovered. There are many algorithms available to discover processes, cf. [2], [12], [13], [4]. Just running different discovery algorithms often gives some process model, but typically, this results in a spaghetti-like process model, since infrequent behavior, like exceptions, are also taken into account. Infrequent behavior means that there is a case whose sequence does not occur too often. Hence, to discover a process model that describes the run of the mill, it is necessary to rule out infrequent sequences first, and to consider sequences with high frequency only. Following the Pareto principle, the log needs to be filtered such that at least $80\%$ of the log is covered, but that it only contains cases with high occurrence frequency. This event log then can be used to discover the control flow. A good check for the process model is to run a conformance check on it. On this result, infrequent behavior can be marked.

### D. Performance Analysis

After discovery of the control flow, these process models can be used to analyze the performance of the process. This phase answers questions like "are there any bottlenecks in the process?". First, a dotted chart analysis [14] is used to compare cases and their throughput times. The vertical axis of a dotted chart represents the cases in the event log, the horizontal axis represents time. This already gives helpful insights in the performance of the system.

Next, by replaying the log on the process model, bottlenecks and throughput times of individual activities and the process itself are calculated [1]. As the process model only describes the run of the mill, this analysis is not enough. Therefore, after the replaying, the different sequences need to be inspected separately, giving valuable information about throughput times in cases with exceptions or other infrequent behavior.

### E. Role Analysis

If the event log is rich enough, or more specifically, if it contains information about whom executed an event, the roles in the process can be analyzed. A *role* is a person or other resource that is involved in the process, by executing activities of that process. This phase gives answers to questions like "who executes what activities?" and "who are working together?". First, a *role-activity matrix* is created. In this matrix, the rows represent the roles, the columns represent each event of the event log. Each cell contains the number of times that role executed that event. Next, for each role a profile is made from this matrix. If roles have similar profiles, they form a group. In this way, the roles in the event log can be partitioned into groups. For the organization it is now possible to check these discovered groups against the organizational model used. This gives the organization helpful information about the division of work within departments of the organization, whether people across departments execute

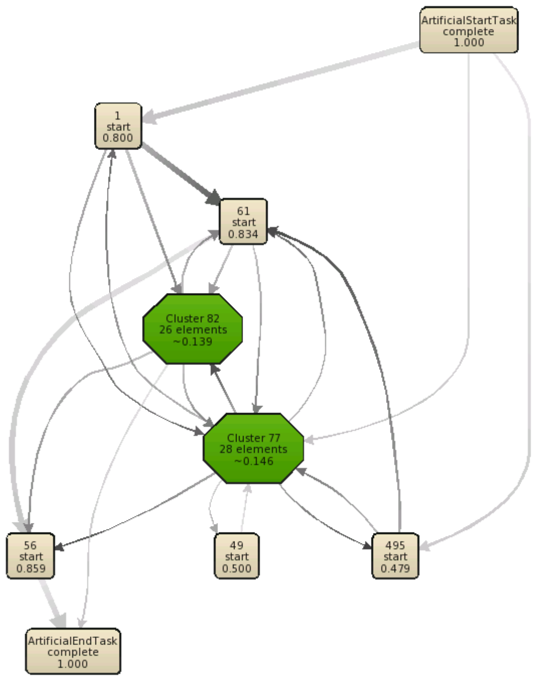

Fig. 2. Fuzzy Miner process model

similar activities, and how communication is between different departments [8].

Next, the roles are analyzed to discover *specialists*, i.e. roles that only execute a few activities, but very frequently, and *generalists*, i.e. roles that execute many different activities in the process. One way to discover these types of roles is to generate a *role hierarchy*, a hierarchy based on the different activities roles execute. A directed arrow between two roles indicate that the role at the base of the arrow can do at least the activities of the role at the arrow head.

Thirdly, a social network analysis is performed. A social network is a (directed) graph representing relations between roles, based on a property discovered in the event log. This property can be handover of work, i.e. roles that pass work of a single case to one and another, and subcontracting, i.e. a role passes work of a single case to another role, and this role returns the work back to the first role after finishing its activity. Important metrics within social network analysis are the *betweenness centrality*, i.e. the number of roles a role can reach by only traversing directed edges, the degree of incoming edges, and the degree of outgoing edges [7].

### F. Transferring the Results

The aim of this methodology is to gain insights in the processes of the organization and their support by information systems. The outcome of the diagnostics shows all behavior seen in the system. This behavior often deviates from the intended process, because of unwanted behavior, like sequences or user profiles that are not allowed due to e.g. legislation, but also because of smart shortcuts taken by the users of the system to make their work easier. The performer of the diagnostics is not able to make the distinction between wanted and unwanted behavior. Therefore, it is important to discuss the outcome of the diagnostics with the client, so that the client understands the outcome, and gets a better understanding of the information system. With this knowledge, and with the support of the performer of the diagnostics, the client can refactor its information system to make it more efficient or to speed it up, e.g. by deciding to disallow certain sequences or user profiles, or to support certain shortcuts taken by its users.

## III. Case Study

For a Dutch governmental organization, we performed a case study to give an overview of the process within their information system, implemented in Oracle $9i$. The organization takes care of a process to issue a document. As more and more people and organizations need this document, the process needs to be highly optimized. To be able to give insights in how the information system currently works, and to give recommendations to improve the system in order to speed up the process, we performed the process diagnostics method to give a broad overview of their system.

### A. Log Preparation

The received log was a text file with records containing 23 fields. Each record contains information about the case identifier, the activity performed, the executing resource, called the originator and several time stamps, indicating the start time of the case, the time on which the event occurred, and time stamps indicating the planned and actual end time of the case. Unfortunately, it was not possible to find a time stamp indicating the end of an activity, which means that it is only known when an activity occurred, but not the time spent on it.

### B. Log Inspection

In our case study, the event log consists of 60 different activities and 83611 cases with in total 276333 events. There were 60 originators involved in the process. In the log we found about 25 start events, of which activities 1 and 495 are the start events in 90% of all cases. The log consists of 30 end events, of which activity 56 is in 87% of all cases the end event.

To get a first glance, the *Fuzzy Miner* [3], a plugin in the tool ProM, is used on the log, with an artificial start and end event added to each case. The Fuzzy Miner clusters coherent activities with log significance but high correlation. Figure 2 shows the resulting process model. The thickness of an arc indicates the frequency. Hence, it appears that the sequence $1 - 61 - 56$ is a frequent path. This model shows that both activities 1 and 495 are the most frequent start events, and that the process terminates mostly with activity 56. This result is in accordance with the statistics found in the previous phase. Hence, it is a good idea to filter the log to get only cases that start with activity 1 or 495 and end with activity 56. This

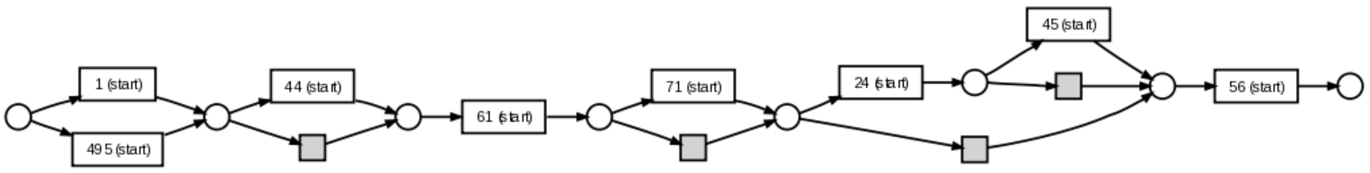

Fig. 3. Process model constructed from most occurring sequences

TABLE I
TOP 15 OF MOST OCCURRING SEQUENCES

| **Pattern** | **Occurrences** (#, %) | | **Average** (h) | **Min** (h) | **Max** (h) | **St. dev** (h) |
|---|---|---|---|---|---|---|
| 1 – 61 – 56 | 41422 | 49.5 | 255.14 | 8.52 | 1725.63 | 104.82 |
| 495 – 61 – 56 | 10250 | 12.3 | 22.82 | 6.24 | 96.69 | 12.64 |
| 1 – 44 – 61 – 56 | 4546 | 5.4 | 253.29 | 10.00 | 879.77 | 114.91 |
| 1 – 61 – 24 – 56 | 3495 | 4.2 | 346.05 | 49.81 | 948.34 | 123.58 |
| 1 – 61 – 24 – 45 – 56 | 3101 | 3.7 | 381.79 | 90.97 | 1132.73 | 125.67 |
| 1 – 61 – 71 – 56 | 1086 | 1.3 | 397.86 | 73.00 | 701.25 | 146.02 |
| 495 – 61 – 24 – 56 | 701 | 0.8 | 119.65 | 18.92 | 233.29 | 46.92 |
| 495 – 44 – 61 – 56 | 429 | 0.5 | 197.49 | 12.12 | 368.49 | 60.17 |
| 1 – 44 – 61 – 24 – 45 – 56 | 421 | 0.5 | 377.24 | 126.25 | 926.75 | 127.56 |
| 1 – 44 – 61 – 24 – 56 | 343 | 0.4 | 350.92 | 87.36 | 911.38 | 127.91 |
| 495 – 61 – 71 – 56 | 264 | 0.3 | 80.78 | 72.20 | 123.06 | 8.83 |
| 1 – 61 – 71 – 24 – 56 | 154 | 0.2 | 432.58 | 117.87 | 646.09 | 132.53 |
| 495 – 61 – 24 – 45 – 56 | 138 | 0.2 | 159.37 | 31.07 | 300.36 | 53.44 |
| 1 – 61 – 71 – 24 – 45 – 56 | 121 | 0.1 | 450.98 | 122.12 | 677.39 | 122.97 |
| 1 – 44 – 61 – 71 – 56 | 111 | 0.1 | 427.23 | 75.75 | 578.13 | 114.14 |
| **Total** | 66582 | 79.6 | | | | |

results in a filtered event log with 67411 cases, which is 81% of the original event log.

### C. Control Flow Analysis

To find the most frequent paths in the event log, we have used the *Performance Sequence Analysis* plugin of ProM, which shows the different patterns, together with some performance metrics of each pattern, like average throughput time. The results of this analysis show that the observation of our frequent path is correct. The sequence $1 - 61 - 56$ is executed 41422 times, i.e. 49.5% of all cases. In total there are 210 different sequences a case can follow. Table I shows the top 15 of most frequent process patterns. This table shows that only 8 activities out of a total of 60 activities, and 15 sequences describe 66582 of the 67411 cases, i.e., most process sequences follow a very simple process model, while only a very small percentage will follow a more difficult path. We filtered the log such that it only contains cases that follow one of these 15 sequences. On this event log, we have discovered the process model shown in Figure 3. The conformance check on this model shows a correct termination of 99.4%, i.e. 99.4% of all cases that start with activity 1 or 495 and end with activity 56 follow exactly this model.

### D. Performance Analysis

On the obtained event log, we have performed a dotted chart analysis, of which the result is shown in Figure 4. From this diagram, we can conclude that between the first two activities, activity 1 and 61, the waiting time deviates a lot. As we do not have any information about the duration of activities, we can only conclude that either activity 1 has a high deviation, or activity 61 is an activity which is executed irregularly. Performance analysis has shown that there are two bottlenecks in the standard process, i.e. the places between activity 1 and 44 and the place between activities 44 and 61. From this, we can conclude that activity 1 has an irregular duration. Table I also indicates that cases starting with activity 495 have a low average throughput time, while starting a case with activity 1 gives a high average throughput time.

### E. Role Analysis

The event log is rich enough to perform a role analysis. The ProM plugin *Organizational Miner* [8] is used to calculate the profile of each of the roles present in the event log. We have discovered 4 different groups. In the first group, 14 roles reside, in the second group 37. Interesting is that the last two groups both consists of a single role. The first two groups executed respectively 15 and 23 different activities. The last two groups executed respectively 2 and 4 activities. Inspecting the role-activity matrix reveals that the activities performed by the last two groups occur very infrequent, while the activities performed by the other roles are very frequent. Most likely, the last two groups are two administrators executing activities if the process got stuck.

Next, specialists and generalists are inspected. A role hierarchy as shown in Figure 6 is discovered. From the generated tree, we can conclude that there are 24 specialists, each executing at most 3 activities, and 5 real generalists that execute each

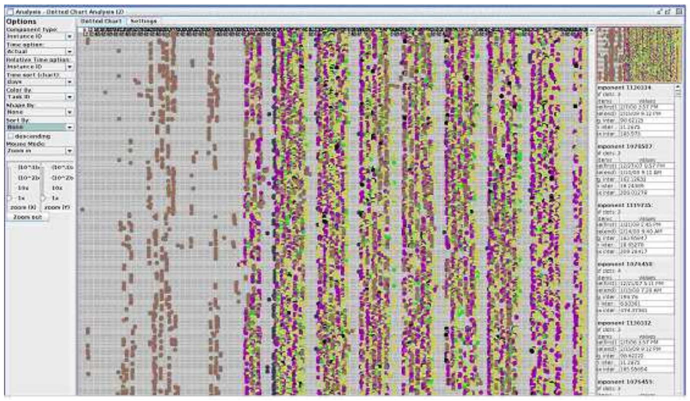

Fig. 4. Dotted Chart Analysis

13 or more different activities. Most users execute between 4 and 9 different activities. Interesting is that the web service is one of the specialists, only executing two different activities. Remarkable is that the admin role only executed 9 activities, which is in this process too few to be a generalist. This indicates that there are only a few cases in which the admin had an important intervention.

A social network analysis depicting the handover of work is shown in Figure 5. In this figure, the wideness of nodes depicts the number of outgoing arcs, while the height of the node depicts the number of incoming arcs. The graph shows that most roles have a very low degree of incoming and outgoing arcs, while only one role has a very large degree of outgoing arcs, indicating that this role handovers activities to many different roles. There are some originators that have a high degree of incoming arcs, which means that they have a kind of collecting position, since they get work handed over from many different roles, while they do not pass it to many other roles. To measure central originators in the process, we calculate the betweenness centrality metric. From this metric we can see that there is one role that is most central. The role-activity matrix reveals that this role only executes activities 513 (37 times), 56 (1658 times) and 24 (3186 times).

### F. Transferring the Results

At the end of the process diagnostics, we had a session with the client in which we explained the obtained results. During the discussion with the client, the client recognized the process model and agreed that this indeed represented the core flow of their processes, although they found the throughput time of cases very high. In the discussion, it appeared that activity 45 indeed was only executed if activity 24 was executed. Activity 45 is an activity in which the case is checked whether it is handled correctly, and should only be executed once every 100 cases. However, it turned out that this activity was executed in almost half of all the traces, implicating that this check was executed too often. Inspecting the period within the organization revealed that this was due to a considerable arrears and the high number of newcomers at that time. The high throughput time could also be explained by the same arguments.

Explaining the role analysis results, in particular the organizational model, the client could name the different groups. The second group, group 1, was a specialized group of people executing the more elaborated activities. The first group consisted mainly of the administration department, together with some people of the second group, who were transferred at that time to help the administration department. The third group was a person who supported the different departments. The last group was the system administrator. Also the high centrality of role "role8" in the social network was directly recognized by the client, who was somehow surprised that we could find this, as this person indeed had a central role since its work was mainly to delegate work to different teams in the departments.

## IV. Conclusions

For organizations to answer the main question: "are we in control?" process mining techniques can be used. However, the diversity of current research in this area makes it hard to see how to apply process mining within organizations. In this paper, we propose a process diagnostics methodology, that gives a broad overview of the process supported by the information system. The methodology only uses some of the available analysis methods, and can be performed in a short period of time.

In the process diagnostics methodology, several perspectives of the process are highlighted. The outcome covers the control flow perspective, i.e. "how the process model actually looks like", the performance perspective, i.e. "how well does the system perform" and the organizational perspective, i.e. "who is involved in the process and how". The outcome of the methodology can be used for further analysis on specific topics.

We used the methodology on a case study for a Dutch governmental organization. This case study took about 50 man hours in total and was carried out without detailed knowledge of the company's process. The results from applying the

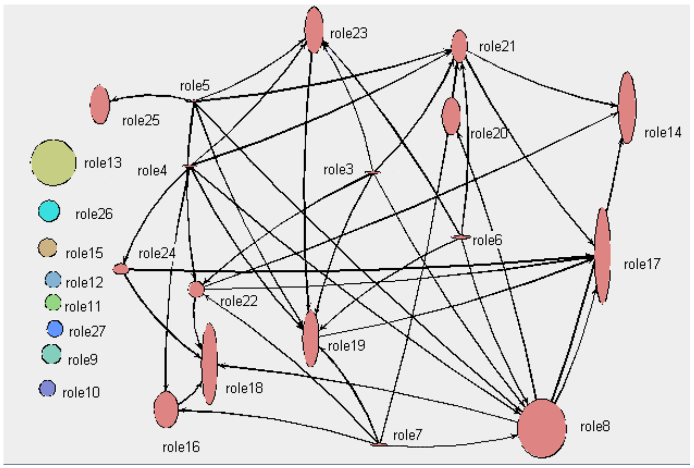

Fig. 5. Social network

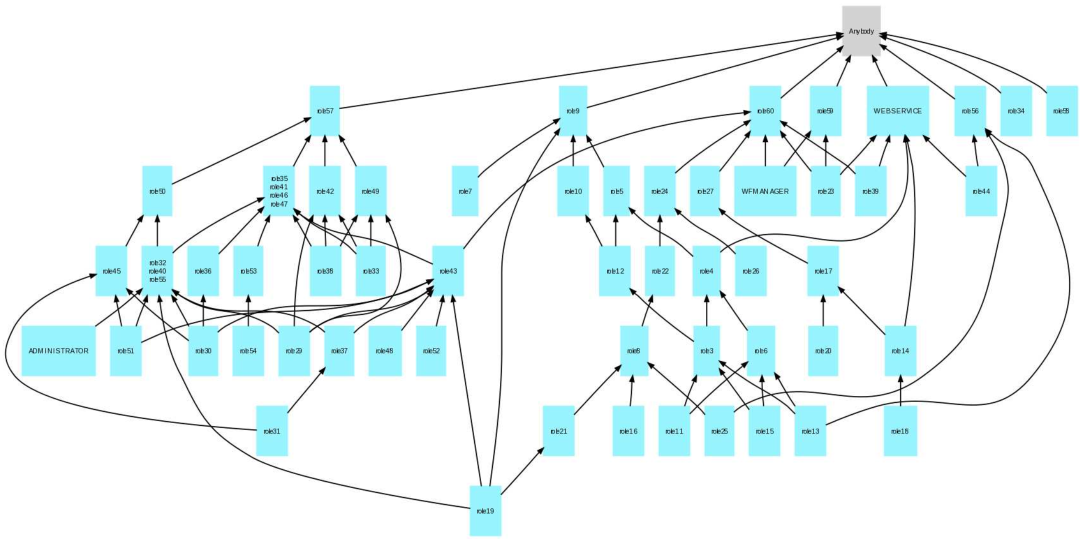
Fig. 6. Role hierarchy based on the activities executed by roles

methodology show that roughly 80% of the cases can be described with a simple process model that has a performance bottleneck. Also the organizational analysis shows that there is one role within the organization that is very central. Such a central point often indicates a weak point within the organization, but such conclusions can only be drawn by the client. In general, process diagnostics only presents facts. To avoid misinterpretation, the result should never be given without detailed explanation from the diagnostics performer. Discussion between the client and the performers helps the client with the interpretation of the outcome.

The case study shows that the process diagnostics methodology indeed can be applied within a short period of time, giving a broad overview of the process within the information systems used. More case studies are needed to fine tune the methodology.

Most of the analysis techniques in the methodology use tools. It can be of value to automate parts of the process mining activities to simplify the usage of process diagnostics, and hence to make it easier.

Presenting the results in a written document, is typically not sufficient, since it only shows results in a fixed level of abstraction and time. The results are often better interpretable if the organization is able to interact with the found models, view exceptions in the process, and, in general, get a better feeling of the process. By giving a snapshot, in some way, we already limit the information. In order to capture all aspects of the process, we need to find better ways to present and visualize the results.

## References

[1] P. Hornix, “Performance analysis of business processes through process mining,” Master’s thesis, Technische Universiteit Eindhoven, 2007.

[2] W. van der Aalst, A. Medeiros, and A. Weijters, “Genetic process mining,” in *ICATPN 2005*, ser. LNCS, vol. 3536, 2005, pp. 48 – 69.

[3] C. Günther and W. van der Aalst, “Fuzzy mining: Adaptive process simplification based on multi-perspective metrics,” in *BPM 2007*, ser. LNCS, vol. 4714, 2007, pp. 328 – 343.

[4] J. M. van der Werf, B. van Dongen, C. Hurkens, and A. Serebrenik, “Process discovery using integer linear programming,” in *ATPN*, ser. LNCS, vol. 5062, 2008, pp. 368 – 387.

[5] A. Rozinat and W. van der Aalst, “Conformance testing: Measuring the fit and appropriateness of event logs and process models,” in *BPM 2005 Workshops*, ser. LNCS, vol. 3812, 2005, pp. 163 – 176.

[6] A. Rozinat and W. van der Aalst, “Conformance testing: Measuring the fit and appropriateness of event logs and process models,” in *BPM 2005 Workshops*, ser. LNCS, vol. 3812, 2005, pp. 163 – 176.

[7] W. van der Aalst, H. Reijers, and M. Song, “Discovering social networks from event logs,” *Computer Supported Cooperative Work*, vol. 14, no. 6, pp. 549 – 593, 2005.

[8] M. Song and W. van der Aalst, “Towards comprehensive support for organizational mining,” Technische Universiteit Eindhoven, Tech. Rep., 2007.

[9] W. van der Aalst, B. van Dongen, C. Günther *et al.*, “Prom 4.0: Comprehensive support for real process analysis,” in *ICATPN*, ser. LNCS, vol. 4546, 2007, pp. 484–494.

[10] W. van der Aalst, H. Reijers, A. Weijters, B. van Dongen, A. Alves de Medeiros, M. Song, and H. Verbeek, “Business process mining: an industrial application,” *Information Systems*, vol. 32, no. 1, pp. 713 – 732, 2007.

[11] A. Rozinat, I. de Jong, C. Günther, and W. van der Aalst, “Process mining of test processes: a case study,” Technische Universiteit Eindhoven, Tech. Rep. WP 220, 2007.

[12] W. van der Aalst, A. Weijters, and L. Maruster, “Workflow mining: Discovering process models from event logs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128 – 1142, September 2004.

[13] A. Weijters, W. van der Aalst, and A. Alves de Medeiros, “Process mining with the HeuristicsMiner algorithm,” Technische Universiteit Eindhoven, Tech. Rep. WP 166, 2006.

[14] M. Song and W. van der Aalst, “Supporting process mining by showing events at a glance,” in *WITS 2007*, 2007.