ProM 6 Exercises

J.C.A.M. (Joos) Buijs and J.J.C.L. (Jan) Vogelaar
{j.c.a.m.buijs,j.j.c.l.vogelaar}@tue.nl

August 2010

The exercises provided in this section are meant to become more familiar with ProM 6 and its process mining plug-ins. Plug-ins that will be covered include the Transition System Miner, Transition System Analyzer, the α-algorithm, the Heuristics Miner, the Genetic Miner, the Fuzzy Miner and the Dotted Chart Analysis. For each exercise detailed solutions are provided which you can use when you get stuck.

It is recommended to have completed the Getting Stared with ProM 6 and the ProM 6 Tutorial before attempting to solve the exercises.

Please note that the event logs provided in this section describe the approximate maximal behavior of the executing process.

1 Exercise 1

The event log we will start to analyse contains the following three traces:

```
1x Case1   A B C D
1x Case2   A C B D
1x Case3   A E D
```

These traces are recorded in exercise1.xes. Please answer the following questions for this event log:

1. Import the event log from the file exercise1.xes.
2. Inspect the contents of exercise1.xes using the log visualization option in ProM and answer the following questions:
   (a) In what timeframe do the events in exercise1.xes occur?
   (b) When did the event `B` occur for `case2.0` and who executed it?
   (c) What is the relative occurrence and what is the frequency of the event `E`?

3. Inspect the contents of exercise1.xes[1] using a text editor. Can you distinguish between the 'header' of the file and the actual traces and events? What does the header include?

4. Construct a transition system using the Transition System Miner in ProM 6. Be sure to set the `key classifier collection` type to `list` and the `collection size` to `no limit` in the second wizard screen. Accept all default settings in all the other wizard screens.

5. Try to construct a Petri net by hand which can replay the traces recorded in exercise1.xes. Use 5 transitions that represent the actions A through E. Try to allow as little extra behavior as possible (e.g. no 'flower'-nets).

6. Create a Petri net from the transition system using the `Transition System to Petrinet` plug-in in ProM. Hint: with the result of the Transition System Miner open, press the `play`-button on the top right.

2 Exercise 2

Repeat the questions of exercise 1 on the traces shown below which are stored in the file exercise2.xes.

```
1x Case1   A C D
1x Case2   B C E
```

It might be nice to play around with the settings of the Transition System Miner and see what the impact is on the resulting transition system.

3 Exercise 3 (extra)

You might want to repeat the questions of exercise 1 for exercise3.xes, of which the content is shown below. The result in a very nice transition system and petri net. No answers are provided for this exercise however.

```
1x Case1   A C E G
1x Case2   A E C G
1x Case3   B D F G
1x Case4   B F D G
```

[1] If you want more information about the XES standard, visit `http://www.xes-standard.org/`.

4 Exercise 4

For this exercise we will use the traces shown below which are stored in exercise4.xes.

```
1x Case1   a b c d f
1x Case2   a c b d f
1x Case3   a b d c f
1x Case4   a c d b f
1x Case5   a d e f
1x Case6   a e d f
```

1. Since it is always a good idea to inspect the event log at hand, inspect exercise4.xes to get an idea of its contents. What is remarkable about this event log?

2. Try to construct a Petri net by hand which can replay the traces recorded in exercise4.xes. Use 6 transitions that represent the actions A through F. Try to allow as little extra behavior as possible (e.g. no 'flower'-nets).

3. Now create three Petri nets using the following plug-ins:

   (a) As in exercises 1-3, first create a transition system from exercise4.xes and then create a petri net from the transition system;

   (b) Run the α-algorithm on exercise4.xes;

   (c) Run the ILP Miner on exercise4.xes.

4. Compare the three Petri nets created by the different algorithms. Are there any fundamental differences or are they bisimilar? Why do you think that there is a difference between (some of) the Petri nets?

5. Now run the Heuristics Miner on exercise4.xes. Is this a Petri net? What is different?

   (a) Create a new visualization of the HeuristicsNet that shows the join and split semantics. Is this a similar net as the Petri nets from the previous steps?

6. Now run the Genetic Miner on exercise4.xes. Why is the result with the default settings so poor? Try to improve the setting and get a result with a fitness of at least 0.8.

5 Exercise 5

For this exercise we will use exercise5.xes which is the result of a simulation.

1. Start with a log inspection on exercise5.xes.

2. Create some proces model(s) using two (or more) of the plug-ins from the previous exercises.

3. Run the dotted chart analysis plug-in on exercise5.xes and try to answer the following questions:

   (a) What do you see using the default settings? What does each row, column and dot represent?

   (b) What does the diagonal line from the top left to the middle of the last row represent?

   (c) Using the default settings, what do you think is the meaning of the row of dots that is located below and to the left of the diagonal line?

   (d) Can we see a clear division of tasks that are executed in the beginning and (repeated) near the end of the process?

   (e) Change the settings to `Time option: Relative(Time)` and `Sort By: Actual duration`. What do we see now?

   (f) Change the settings to `Component type: Originator`, `Time option: Actual` and `Sort By: None`. What do we see now?

   (g) Using these settings, can we distinguish groups of users doing similar tasks?

6 Exercise 6

For this exercise we will use exercise6.xes which is the result of a simulation.

1. Start with a log inspection on exercise6.xes. What is remarkable about this event log?

2. Optional: run the Dotted Chart Analysis on this event log to visualize it in another way.

3. Create some proces model(s) using two (or more) of the plug-ins from the previous exercises.

4. Run the Fuzzy Miner on the event log, using the default settings. Now try to answer the following questions:

   (a) What is remarkable about the resulting process model compared to the models discovered by the other plug-ins?

   (b) The Fuzzy Miner tries to aggregate low frequent activities into clusters and only show edges which are significant enough. Using this brief explanation, can you indicate why the resulting process model looks so different?

(c) Read the documentation on the Fuzzy Miner which can be found at `http://prom.win.tue.nl/research/wiki/online/fuzzyminer`. Please note that this describes the ProM 5 version of the plug-in so step 2 in the *preparation* section is not applicable.

(d) Play around with the settings of the Fuzzy Miner (the sliders you see on the right). What do you need to change to get closer to the models as discovered by the other plug-ins?

(e) Why do you think this model is still not exactly the same as the original model?

(f) On what kind of logs would the Fuzzy Miner be useful?

(g) Can you now think of some benefits but also some weak points of the Fuzzy Miner?

5. Run the Transition System Miner on the event log. Be sure to set the `backward keys` to both `Event Name` and `Lifecycle transition` and not `Resource`, as is shown in Figure 1.

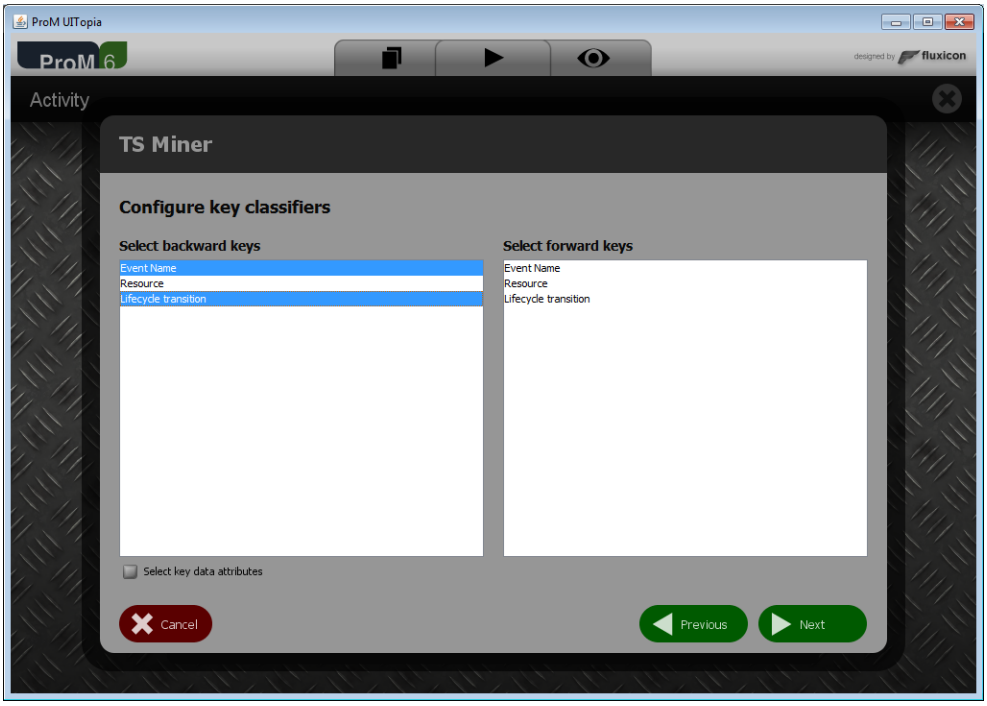

Figure 1: Correct Transition System Miner settings for exercise6.xes.

6. From the resulting transition system, press the `Play` button on the top right. Select the `Transition system analyzer` plug-in. Now click the `Log` object in the `input` column and select exercise6.xes. Now run the plug-in (press `Start`) and try to answer the following questions:

(a) How long does a case take on average? And what are the minimal and maximal durations recorded? On how many ways can you get this information from this plug-in?

(b) What do the red states indicate? And what do the red arrows indicate?

(c) On the top left, change the `Color By:` setting from `Sojourn` to `elapsed`. Why is part of the process colored red and why are the last two events yellow?

(d) If we change the setting to `remaining` we see the first events colored red and 1 event colored yellow. Can you explain this? And why do you think that the outgoing arrow of the yellow event is colored red?

7 Solutions to the Exercises

In this section we will provide detailed answers for the exercises.

7.1 Solutions for Exercise 1

The following screenshots provide a step-by-step indication on how to get to the answer of each question.

1. *Import the event log from the file exercise1.xes.*
   See the ProM tutorial for an explanation on how to import event logs into ProM.

2. *Inspect the contents of exercise1.xes using the log visualization option in ProM.*

   (a) *In what timeframe do the events in exercise1.xes occur?*
   On the right hand side of the log visualization dashboard, as shown in Figure 2, the start and end date of the event log is provided. The answer is from 9 December 2008 8:20:01 CET until 9 December 2008 8:23:01 CET, or in a timeframe of 3 minutes.

Figure 2: Viewing the log dialog of exercise1.xes in ProM

   (b) *When did the event `B` occur for `case2.0` and who executed it?*
   By inspecting `case2.0` we can look up the occurrence of event `B` and we can see that this event occurred on 9 December 2008 8:22:01.527 CET and that is was executed by `UNDEFINED`, as is shown in Figure 3.

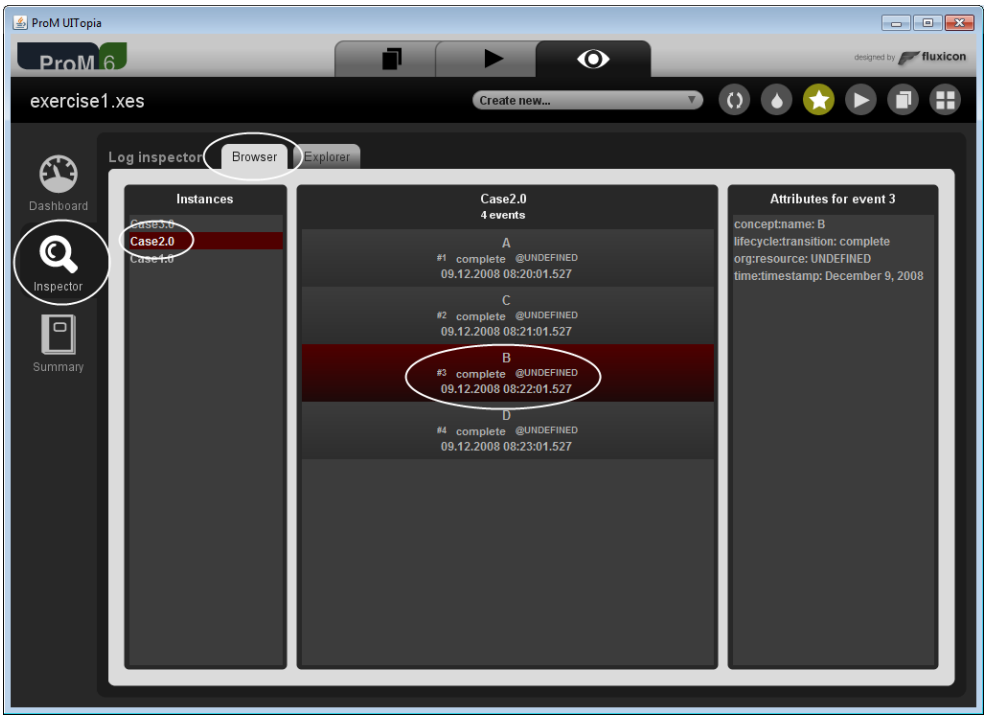

Figure 3: Inspecting exercise1.xes in ProM

(c) *What is the relative occurrence and what is the frequency of the event* `E`*?*
The relative occurrence can be found in the log summary and indicates of all events, how many are of `E`. In our case 9% of the events in the event log is `E`, as is shown in Figure 4. The frequency of event `E` can be found by going to the `explorer` mode in the `inspector`. As we can see in Figure 5 the frequency of `E` is 33.3% which means that event E occurs at least once in one third of the traces in the event log.

3. *Inspect the contents of exercise1.xes using a text editor. If you want more information about the XES standard visit* `http://www.xes-standard.org/`.
If you open exercise1.xes in a text editor with XML syntax highlighting you get something like Figure 6.

4. *Try to construct a Petri net by hand which can replay the traces recorded in exercise1.xes. Use 5 transitions that represent the actions A through E. Try to allow as little extra behavior as possible (e.g. no 'flower'-nets).*
See the result of the Transition System Miner in Figure 10

5. *Construct a transition system using the Transition System Miner in ProM 6. Be sure to set the* `key classifier collection` *type to* `list` *and the collection size to* `no limit`.
Figure 7 shows how to start the Transition System Miner, the correct settings are

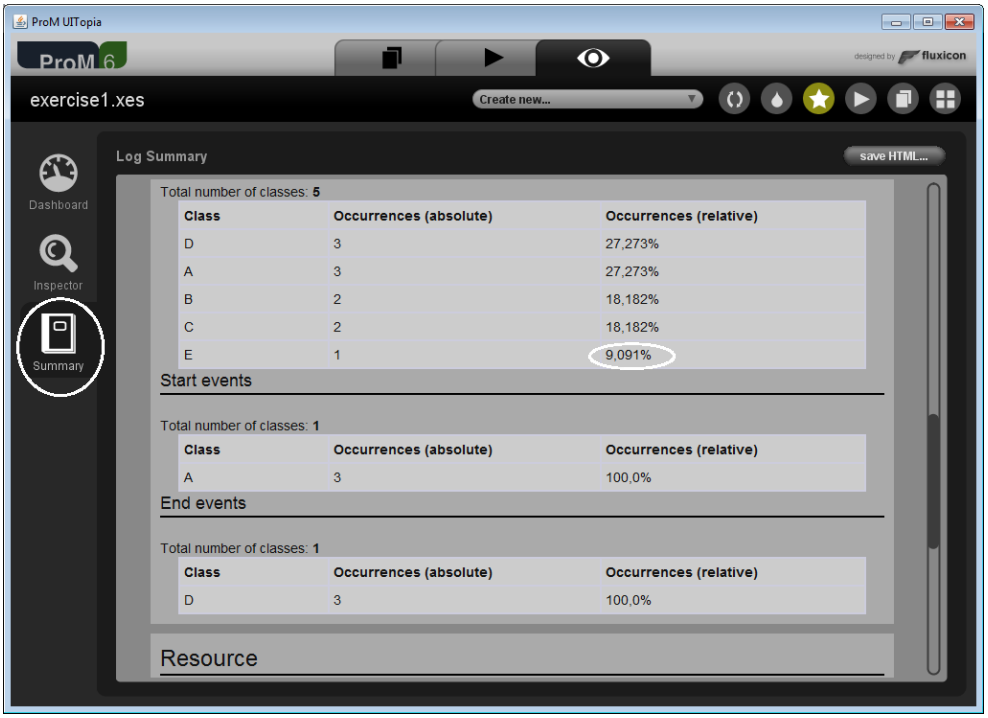

Figure 4: Viewing the log summary of exercise1.xes in ProM

Figure 5: Using the explorer to inspect the traces in exercise1.xes.

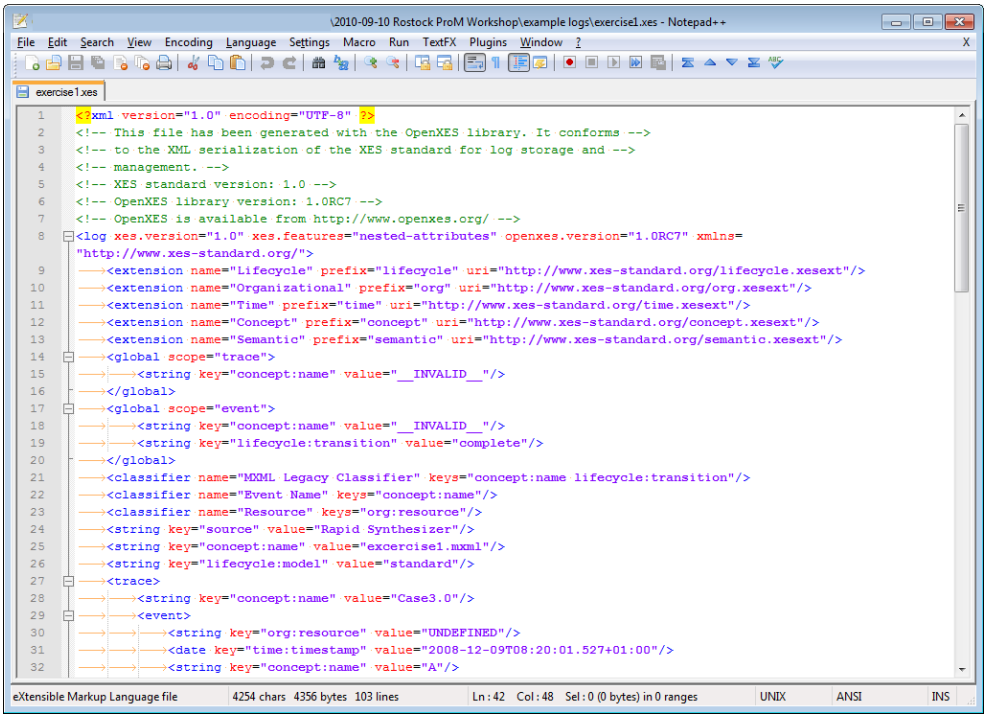

Figure 6: Viewing exercise1.xes in NotePad++

shown in Figure 8. The expected result is shown in Figure 9.

6. *Create a Petri net from the transition system using the* ***`Transition System to Petrinet`*** *plug-in in ProM. Hint: with the result of the Transition System Miner open, press the* ***`play`****-button on the top right.*
   The resulting Petri net is shown in Figure 10.

7.2 Solutions for Exercise 2

The following screenshots provide a step-by-step indication on how to get to the answer of each question.

1. *Inspect the contents of exercise2.xes using the log visualization option in ProM.*
   Your results should be similar to the screenshots shown in Figure 11, Figure 12, Figure 13 and Figure 14.

2. *Inspect the contents of exercise2.xes using a text editor. If you want more information about the XES standard visit* ***`http://www.xes-standard.org/`****.*
   If you open exercise2.xes in a text editor with XML syntax highlighting you get something like Figure 15

3. *Try to construct a Petri net by hand which can replay the traces recorded in exercise2.xes. Use 5 transitions that represent the actions A through E. Try to allow as*

Figure 7: Start the Transition System Miner on exercise1.xes.

Figure 8: The correct settings for the Transition System Miner (all the other default settings are alright for now).

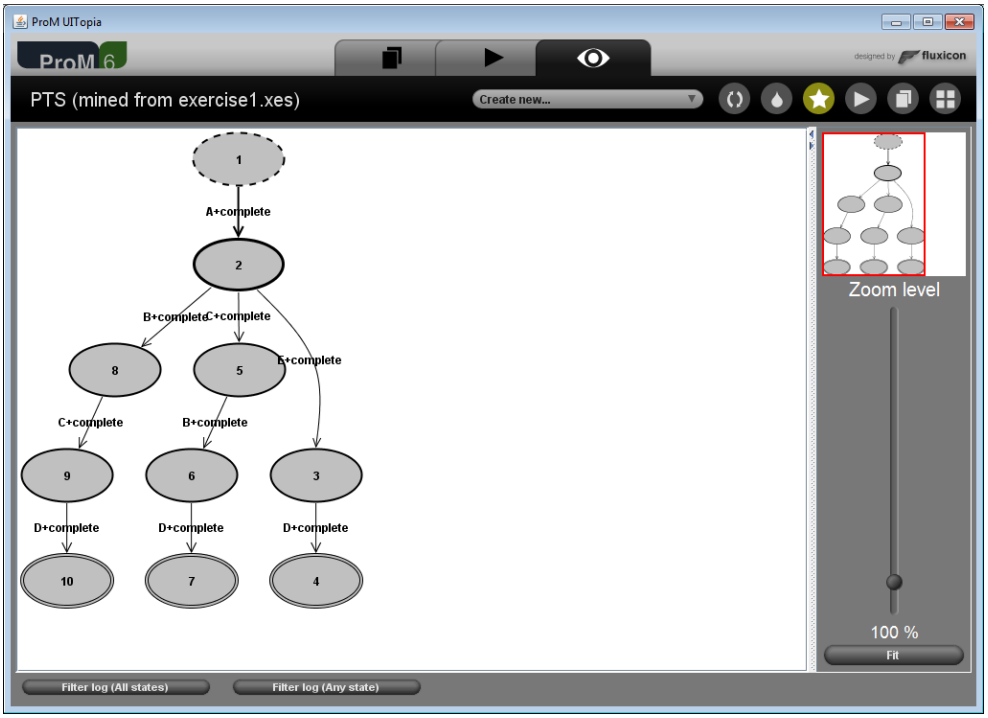

Figure 9: The expected resulting transition system.

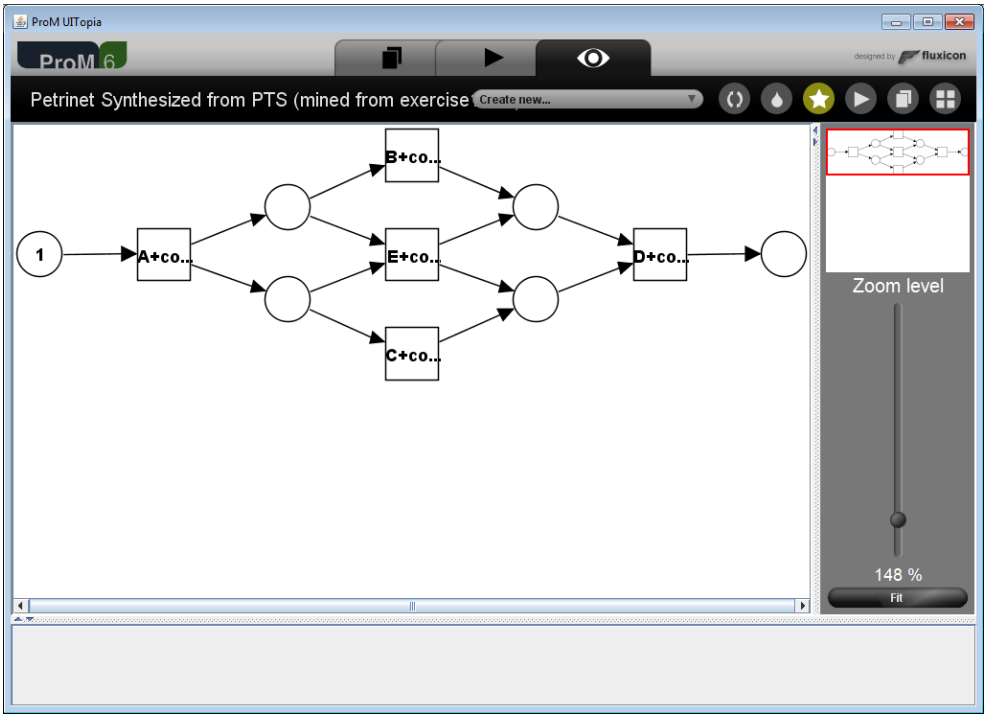

Figure 10: The Petri net created from the transition system for exercise1.xes.

Figure 11: Viewing the log dialog of exercise2.xes in ProM

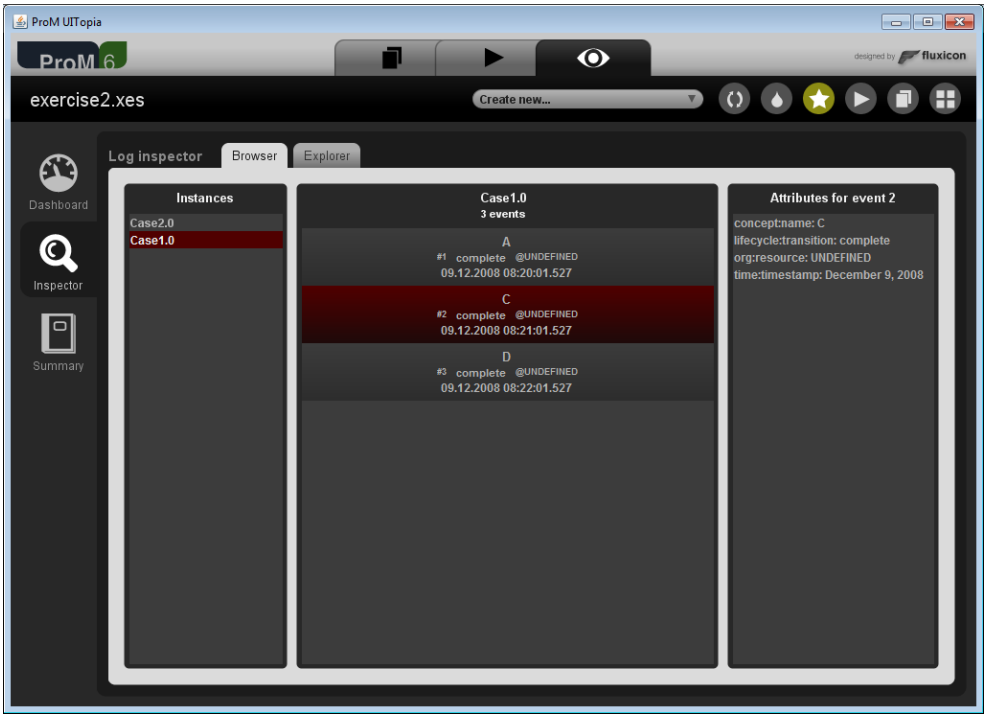

Figure 12: Inspecting exercise2.xes in ProM

Figure 13: Using the explorer to inspect the traces in exercise2.xes.

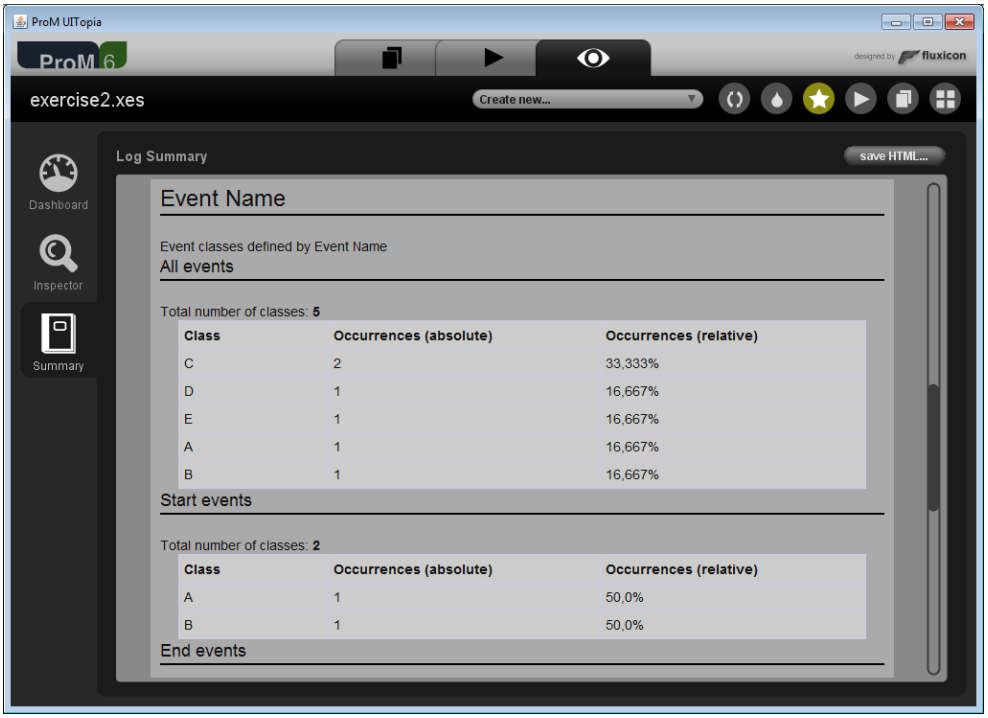

Figure 14: Viewing the log summary of exercise2.xes in ProM

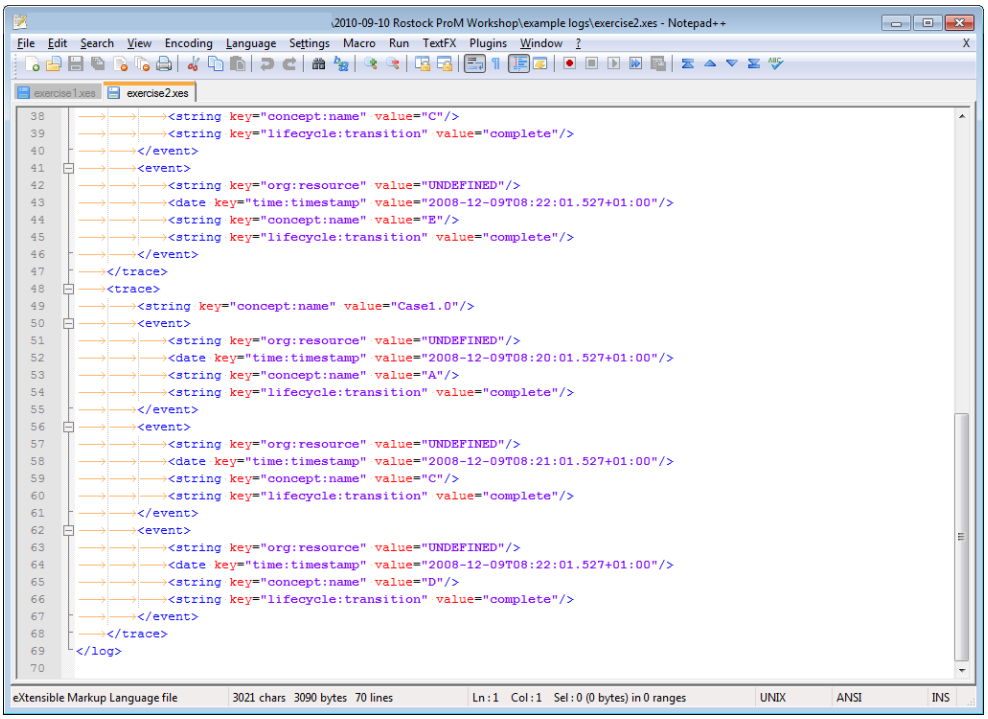

Figure 15: Viewing exercise2.xes in NotePad++

little extra behavior as possible (e.g. no 'flower'-nets).
See the result of the Transition System Miner in Figure 17

4. *Construct a transition system using the Transition System Miner in ProM 6. Be sure to set the* ***key classifier collection*** *type to* ***list*** *and the collection size to* ***no limit****.*
   The expected result is shown in Figure 16.

5. *Create a Petri net from the transition system.*
   The expected result is shown in Figure 17.

7.3 Solutions for Exercise 3 (Extra)

Not provided since repeating the solution approaches of exercises 1 and 2 would provide you with the correct answers.

7.4 Solutions for Exercise 4

1. *Since it is always a good idea to inspect the event log at hand, inspect exercise4.xes to get an idea of its contents.*
   One of the most remarkable things is that all events have the same date and time.

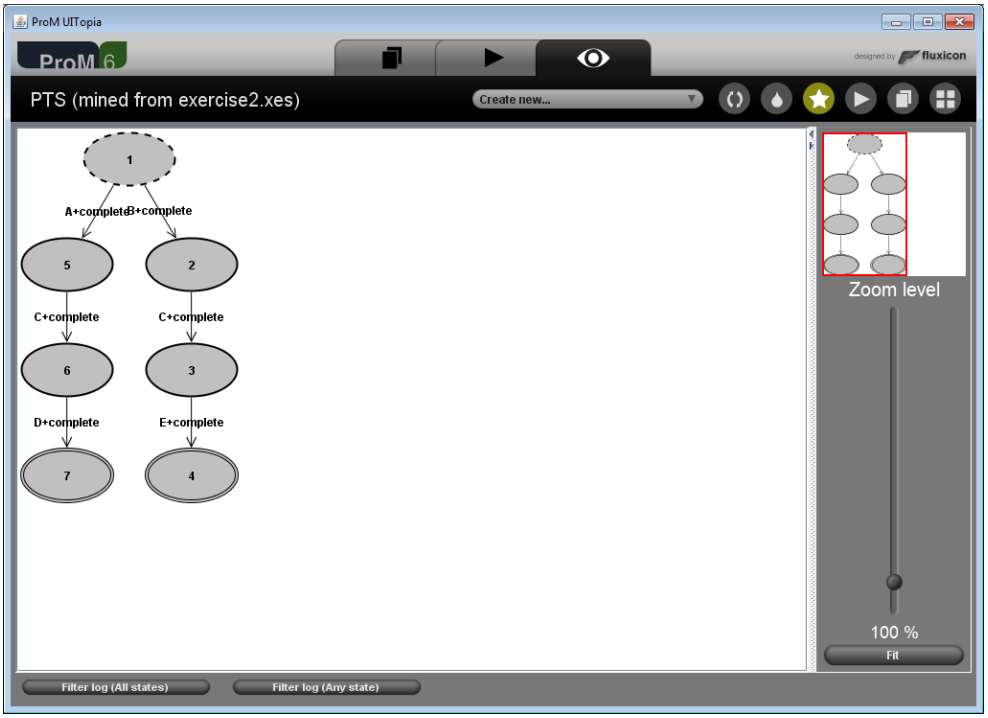

Figure 16: The expected resulting transition system.

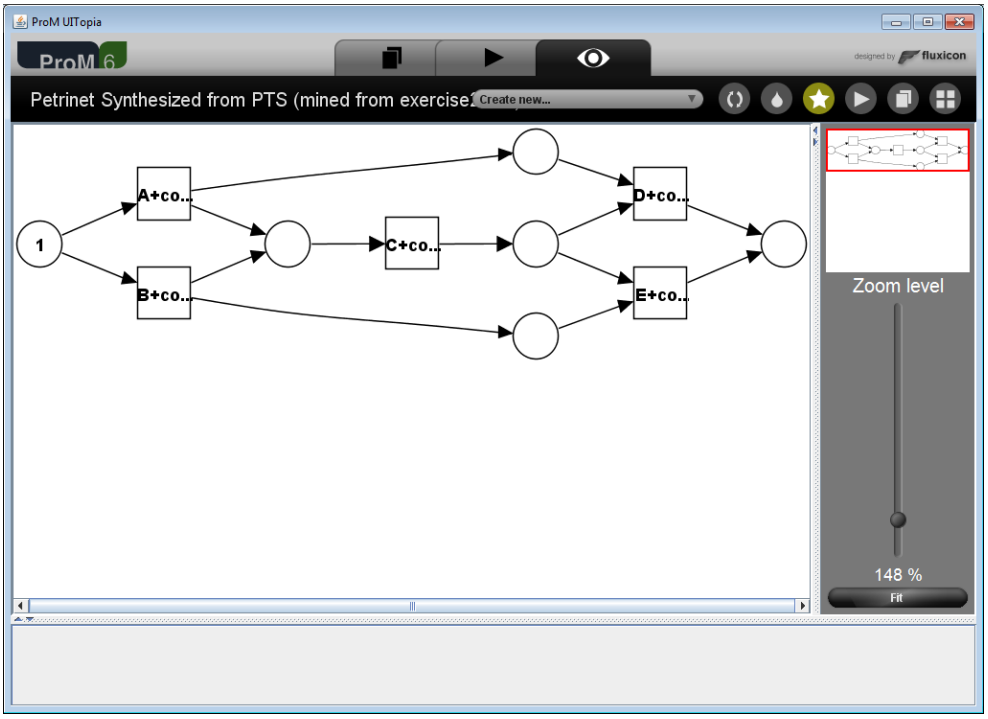

Figure 17: The Petri net created from the transition system for exercise2.xes.

2. *Try to construct a Petri net by hand which can replay the traces recorded in exercise4.xes. Use 6 transitions that represent the actions A through F. Try to allow as little extra behavior as possible (e.g. no 'flower'-nets).*
   See the results of the Petri net discovery plug-ins performed for the next question in Figure 20 and Figure 21.

3. *Now try to create a Petri net using the following plug-ins:*

   (a) *As in exercises 1-3, first create a transition system from exercise4.xes and then create a petri net from that;*
   The expected results are shown in Figure 18 and Figure 19.

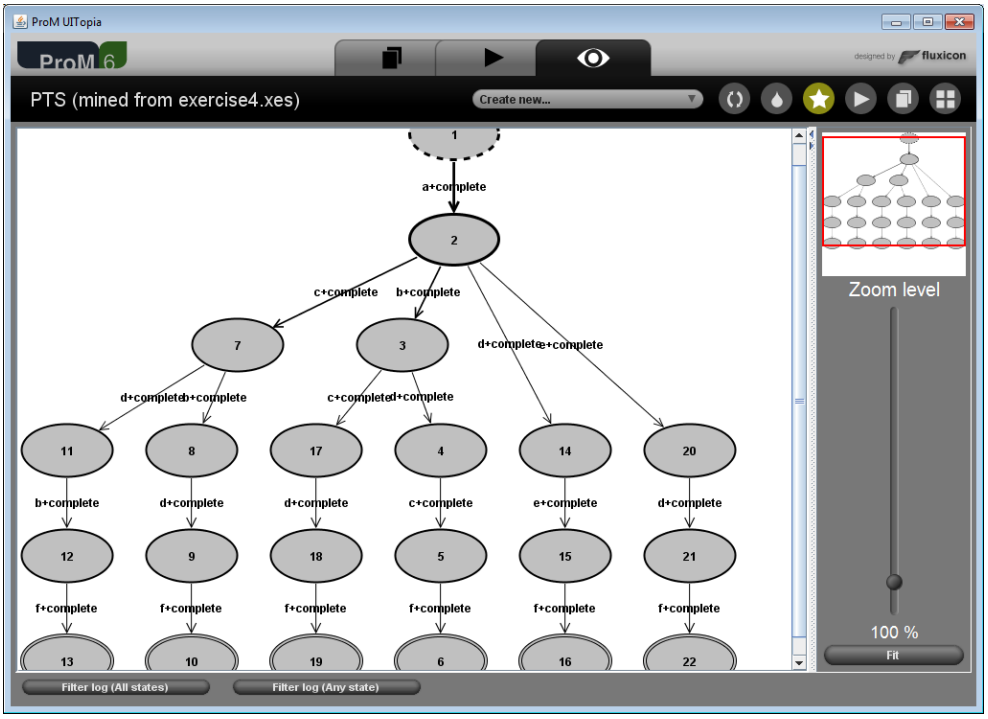

Figure 18: The result of the Transition System Miner on exercise4.xes with the same options as for Exercise 1.

   (b) *Run the α-algorithm on exercise4.xes;*
   The expected result is shown in Figure 20.

   (c) *Run the ILP Miner on exercise4.xes.*
   The expected result is shown in Figure 21.

4. *Compare the three petri nets created by the different algorithms. Are there any fundamental differences or are they essentially the same? Why do you think that there is a difference between (some of) the Petri nets?*
   The resulting Petri nets of the α-algorithm and the ILP Miner are the same. The one

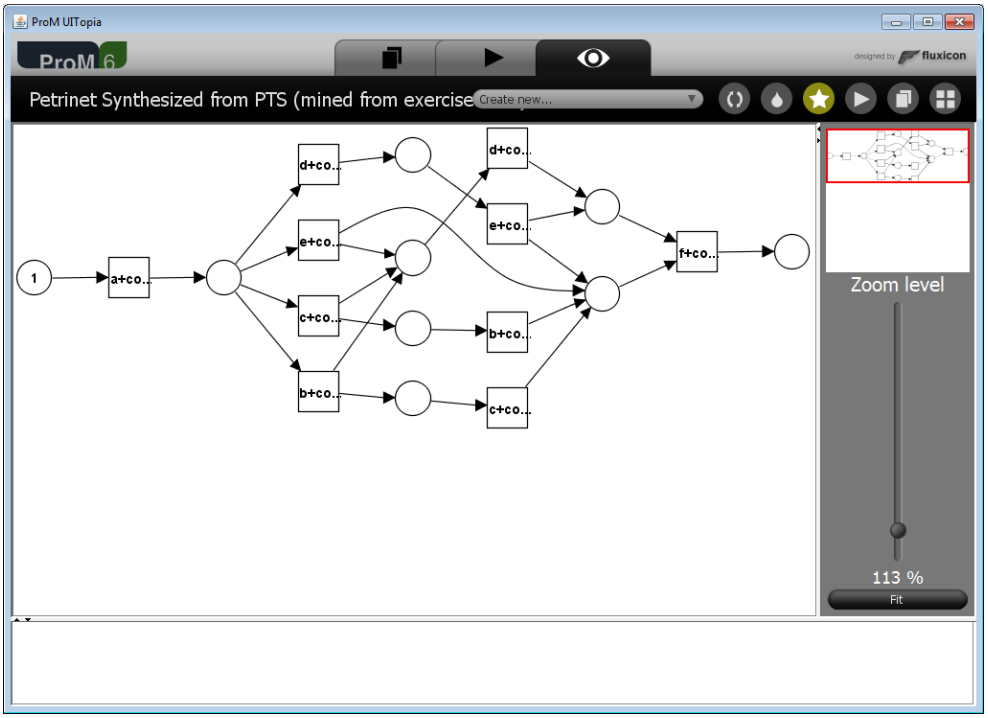

Figure 19: The result of the Transition System Miner to Petri net conversion.

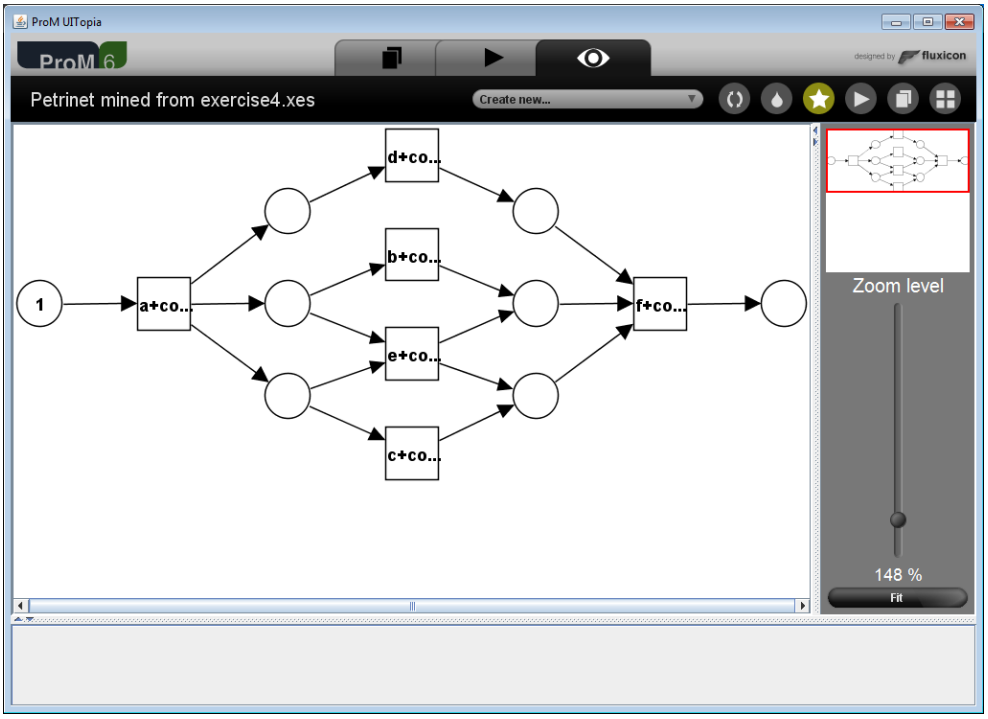

Figure 20: Result of the α-algorithm on exercise4.xes.

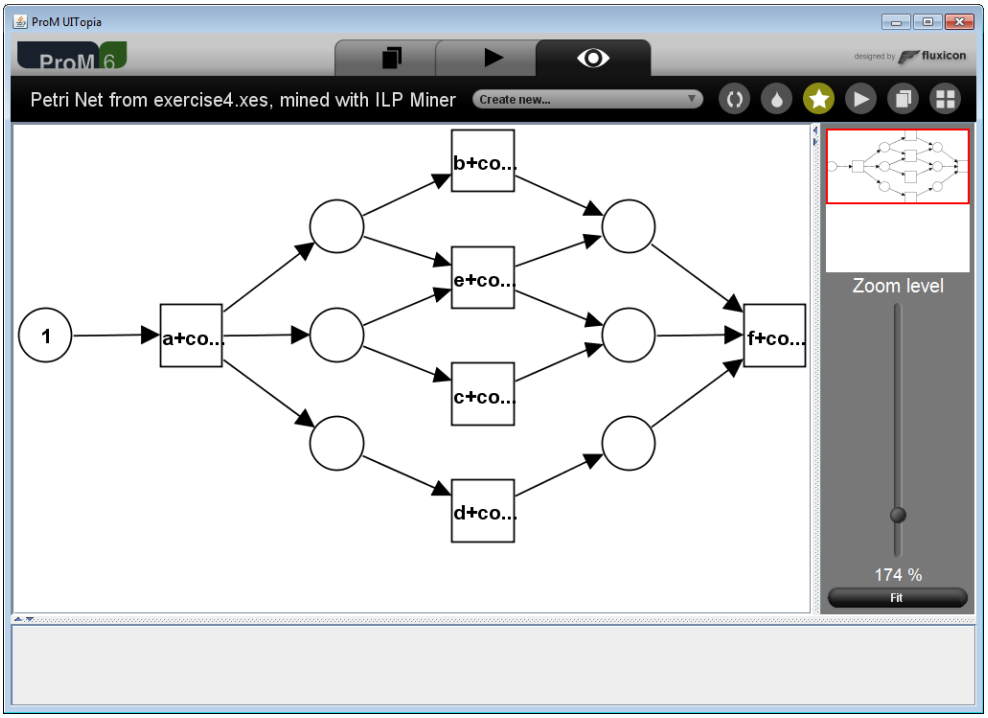

Figure 21: Result of the ILP Miner (with default settings) on exercise4.xes.

converted from the Transition System Miner result is different however. The reason for this is that the α-algorithm and the ILP Miner only allow for each event to be present only once. In the transition system events are present multiple times. During the conversion of the transition system to a Petri net not all events are merged, therefore the resulting Petri net looks different but describes exactly the same behavior.

5. *Now run the Heuristics Miner on exercise4.xes. Is this a Petri net? What is different?*
   The result of the Heuristics Miner, as shown in Figure 22, is not a Petri net. The net only indicates direct relationships without specifying the split and join semantics.

   (a) *Create a new visualization of the HeuristicsNet that shows the join and split semantics. Is this a similar net as the Petri nets from the previous steps?*
   To visualize the Heuristics net from the previous action with the split and join semantics we need to select the visualization option as shown in Figure 23. The resulting net then shows the split and join semantics, as shown in Figure 24. The resulting net is essentially the same as the Petri nets that resulted from the α-algorithm and the ILP Miner.

6. *Now run the Genetic Miner on exercise4.xes. Why is the result with the default settings so poor? Try to improve the setting and get a result with a fitness of at least 0.8.*

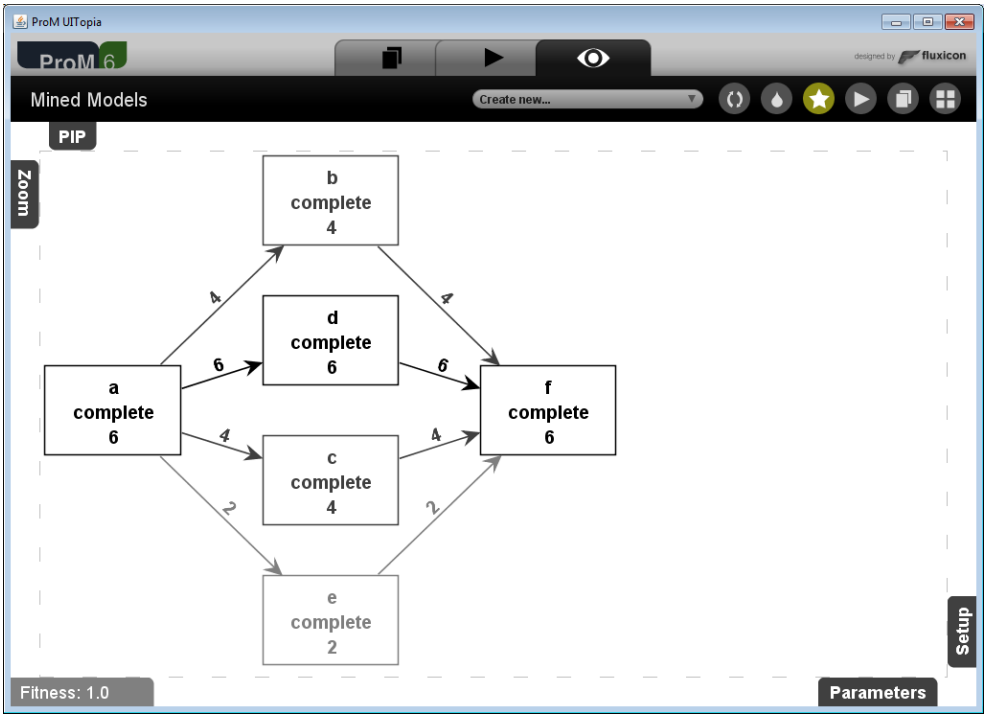

Figure 22: Initial result of the Heuristics Miner

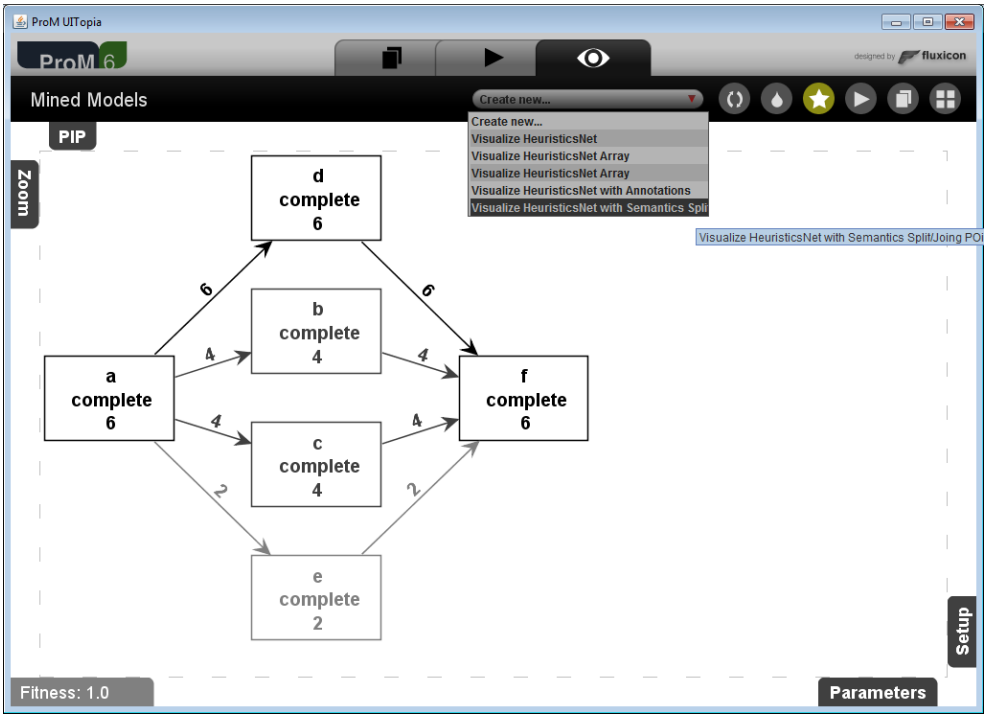

Figure 23: Option to visualize the Heuristics net with the split/join semantics.

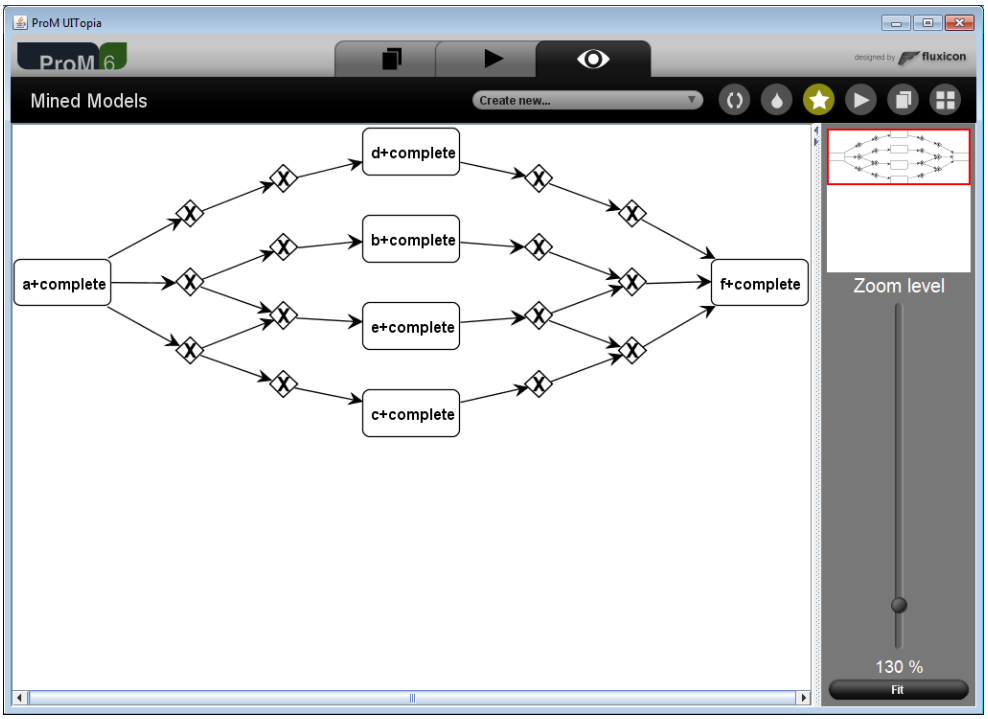

Figure 24: The same Heuristics net but now in EPC form with split and join semantics shown.

In Figure 25 the result with the default settings for the Genetic Miner is shown. We increased the number of iterations from 2 to 200 to get a model with a fitness of 0.875. In Figure 26 the resulting model is shown.

7.5 Solutions for Exercise 5

1. *Start with a log inspection on exercise5.xes.*
   There is not really anything special to discover during log inspection in this case.

2. *Create some proces model(s) using two (or more) of the plug-ins from the previous exercises.*
   The result of the α-algorithm is shown in Figure 27 and the result of the Fuzzy Miner in Figure 28. Each plug-in should come up with similar process models.

3. *Run the dotted chart analysis plug-in on exercise5.xes and try to answer the following questions:*

   (a) *What do you see using the default settings? What does each row, column and dot represent?*
   When you run the Dotted Chart on exercise5.xes you get a result similar to Figure 29, although your colors can be different. Each horizontal line is one

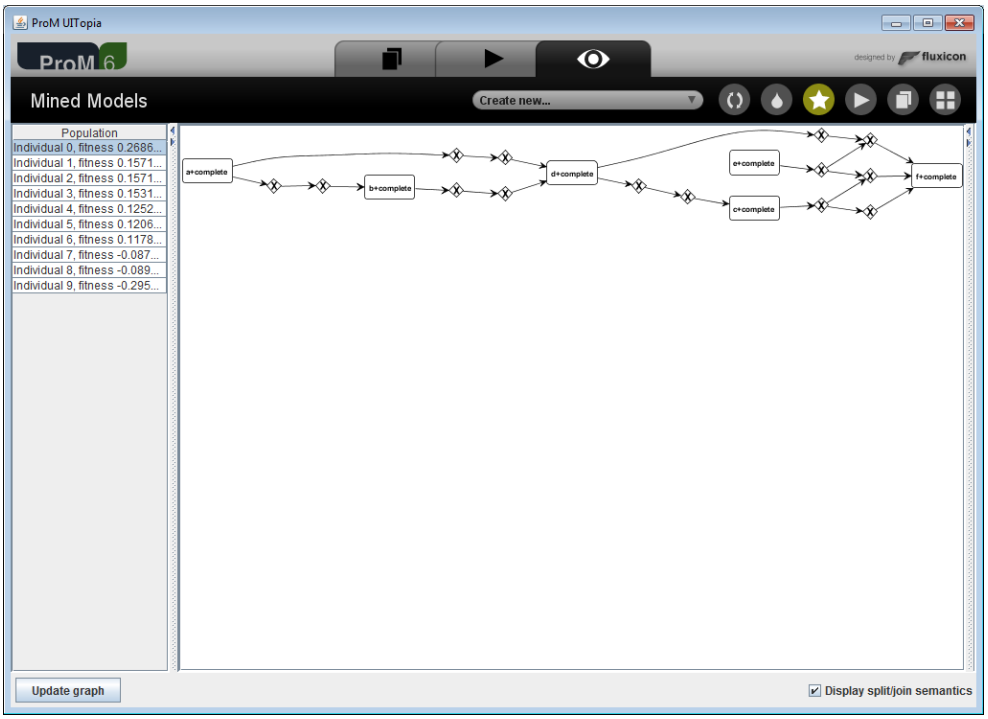

Figure 25: The result of the Genetic Miner on exercise4.xes with the default settings and split/join semantics shown. This model has a fitness of 0.26.

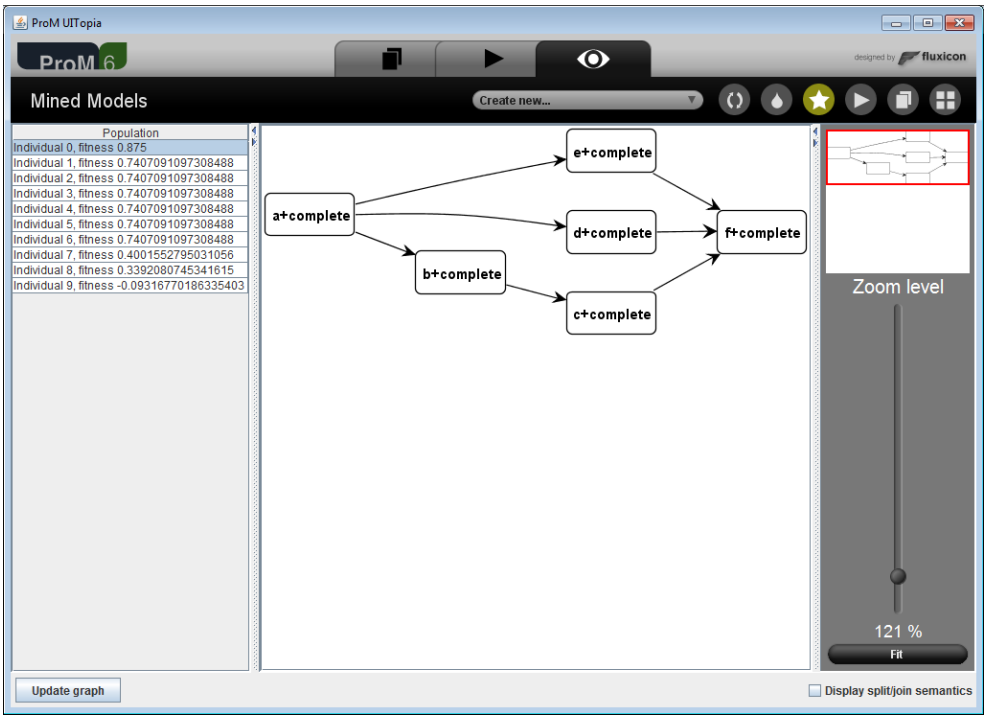

Figure 26: The result of the Genetic Miner on exercise4.xes with 200 iterations.

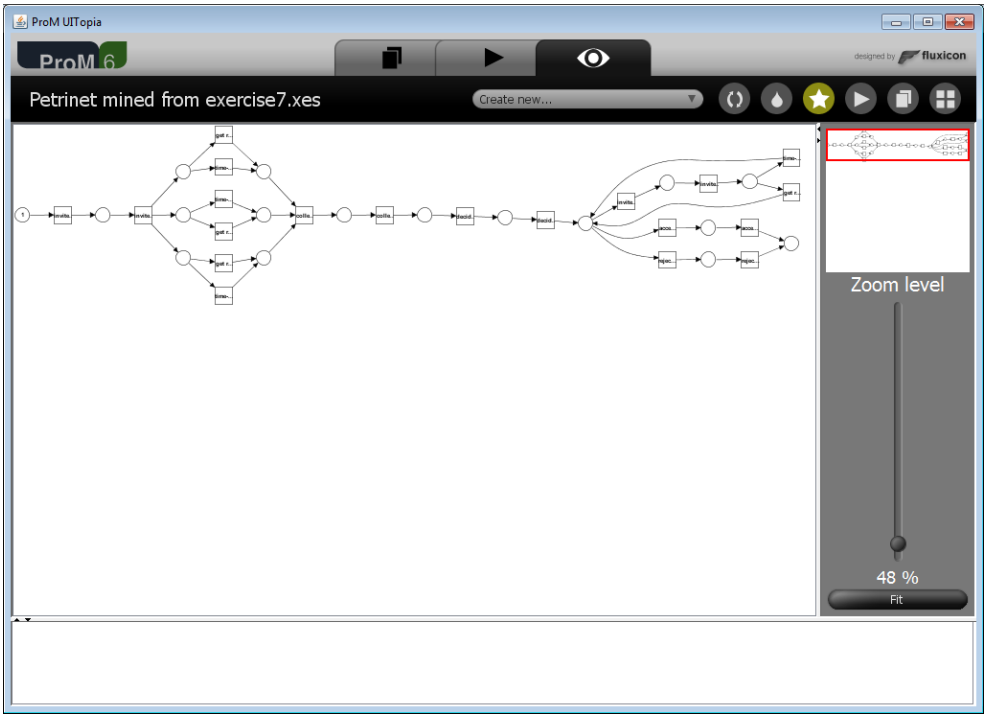

Figure 27: Result of the α-algorithm on exercise5.xes.

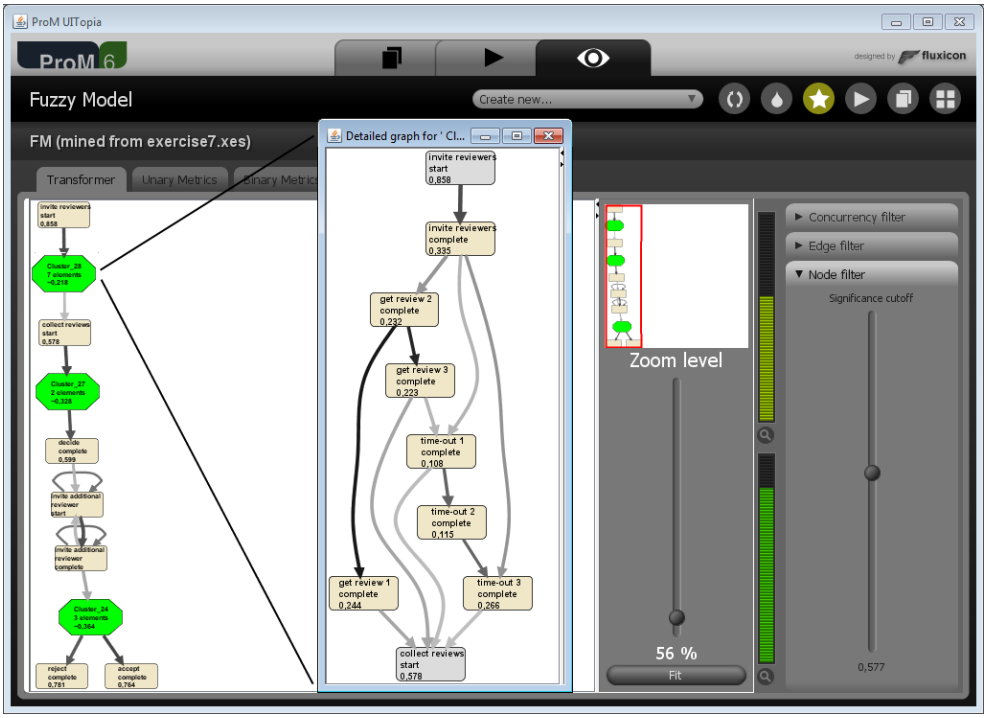

Figure 28: Result of the Fuzzy Miner on exercise5.xes.

case, as the `Component Type` is set to `Instance ID`. On the x-axis on top you see dates so from left to right you progress through time. Each dot is an event as recorded in the event log. Each color refers to a particular activity, as the `Color By` option is set to `Task ID`.

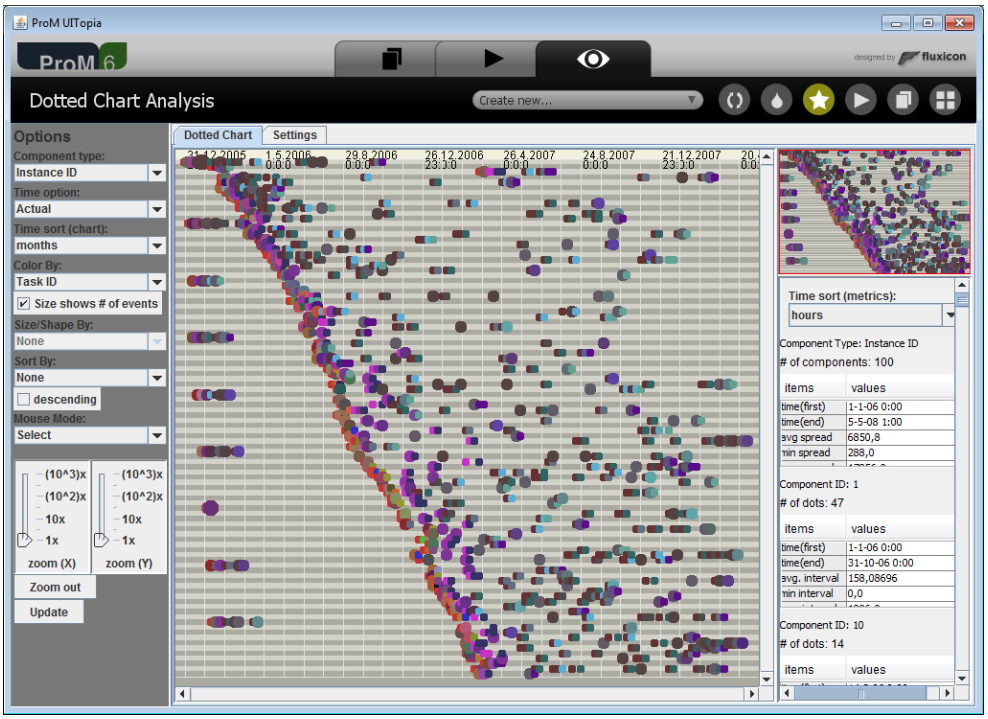

Figure 29: Result of the Dotted Chart with default settings on exercise5.xes.

(b) *What does the diagonal line from the top left to the middle of the last row represent?*
The diagonal line highlighted in Figure 30 indicates the start of new cases. Since it is logical that cases with a higher identification number start later in time, the line of start events will not be vertical but actually progress diagonal.

(c) *Using the default settings, what do you think is the meaning of the row of dots that is located below and to the left of the diagonal line?*
In Figure 30 these sections of dots are circled. If you select dots in ProM you will see that the identification number of the case is between 2 and 9. The sorting algorithm sorts the numbers on their first digit so the order in the graph is 19-2-20-21. Can you spot case number 100?

(d) *Can we see a clear division of tasks that are executed in the beginning and (repeated) near the end of the process?*
Yes, if we look again at Figure 29 we see that at the beginning of cases 'red' events are executed which do not appear later in time for a case.

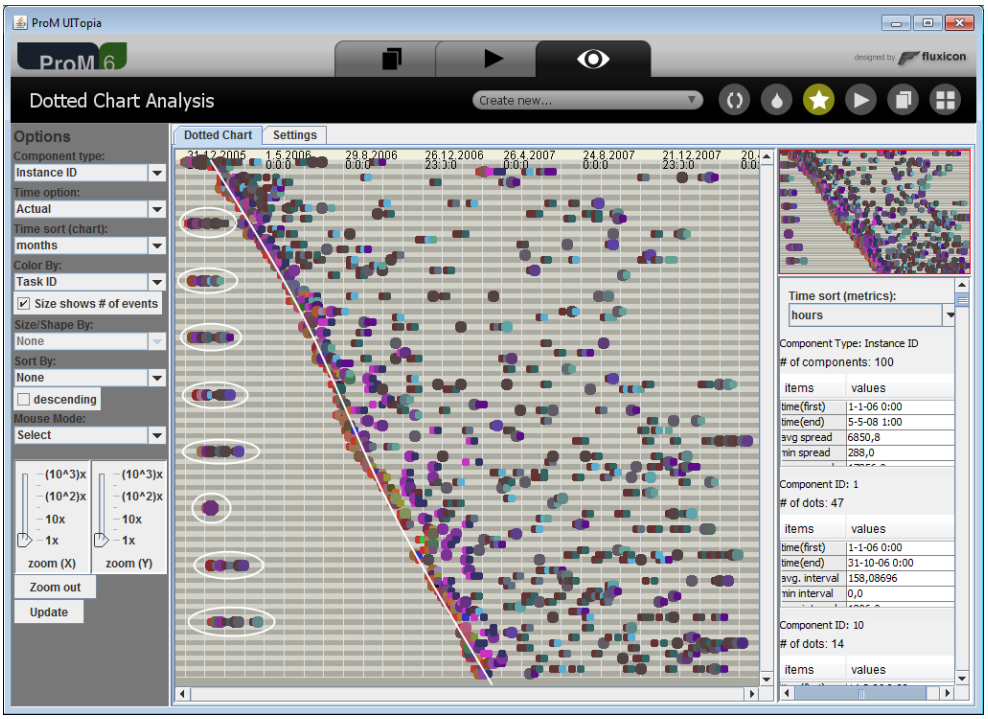

Figure 30: Dotted chart with the start event diagonal indicated and 'incorrectly sorted' cases.

(e) *Change the settings to* `Time option: Relative(Time)` *and* `Sort By: Actual duration`*. What do we see now?*
The resulting dotted chart is shown in Figure 31. Time is made relative from the first event for each case and cases are sorted with the shortest on top. This gives a good indication of the time distribution amongst cases. We can for instance see that some cases are processed within 30 or 60 days. The longest case however lasts for 720 days after its first registered event.

(f) *Change the settings to* `Component type: Originator`, `Time option: Actual` *and* `Sort By: None`*. What do we see now?*
The result is shown in Figure 32. What we see now is which user executes which activity at what time.

(g) *Using these settings, can we distinguish groups of users doing similar tasks?*
It appears that there are 4 groups of users doing similar tasks. Anne and Mike do similar tasks as does the group of Carol, John, Mary, Pam, Pete, Sam and Sara. Then there is Wil which only performs one type of tasks and he is the only one that performs this task. Then there is the '_INVALID' 'user' which 'executes' all kinds of activities.

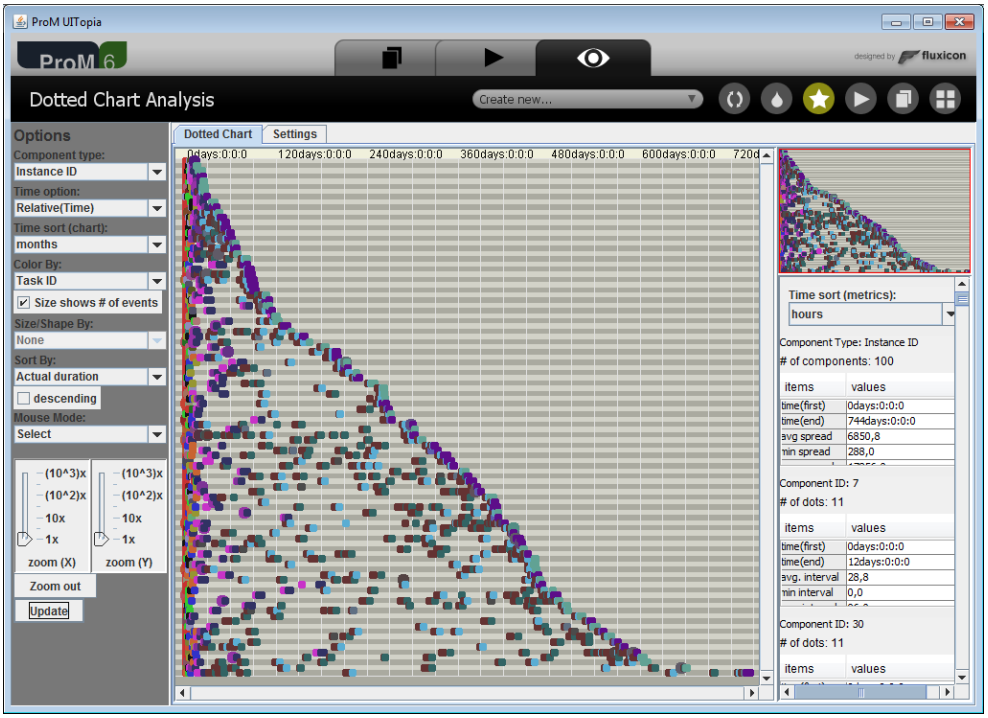

Figure 31: Dotted chart with relative time, sorted by case duration.

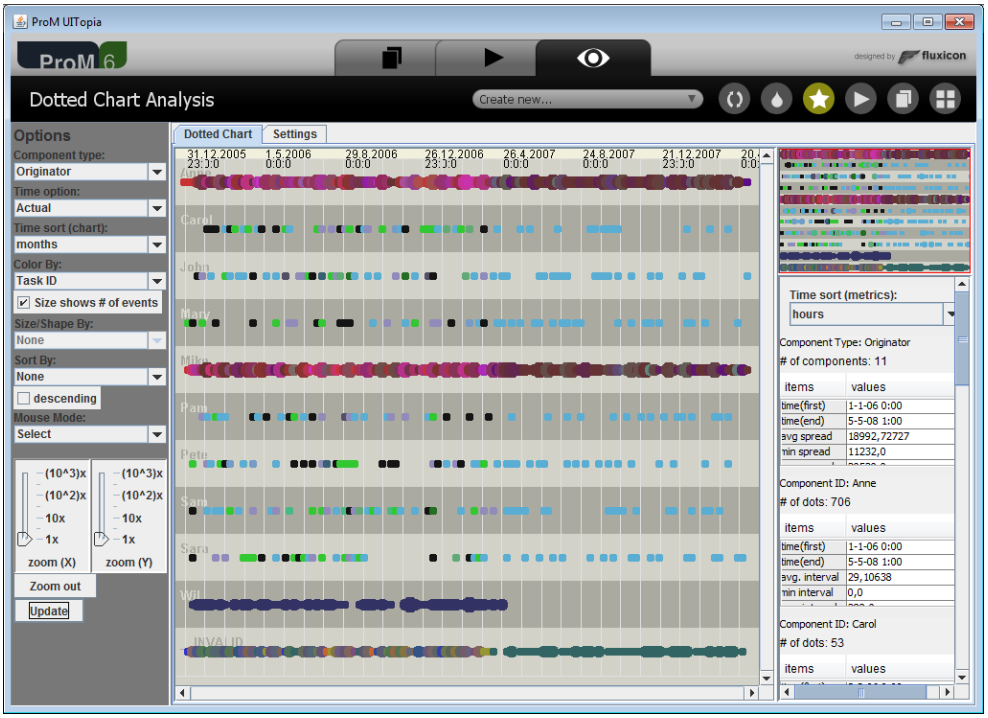

Figure 32: Dotted chart set to show users and their activities over time.

7.6 Solutions for Exercise 6

1. *Start with a log inspection on exercise6.xes. What is remarkable about this event log?*
   The most remarkable thing is that the event log includes events that are executed between 2006 and 2018. It also contains events of 2 types: `start` and `complete`. This allows for analysis of waiting and execution times.

2. *Optional: run the Dotted Chart Analysis on this event log to visualize it in another way.*
   You should know how to do this but we won't discuss things that you might discover.

3. *Create some proces model(s) using two (or more) of the plug-ins from the previous exercises.*
   The Heuristics Miner and α-algorithm come up with correct models. If the Transition System Miner is run with the settings as explained later on, the model is also correct. When run with the default settings you end up with 2 end events which in essence are the same activities.
   The ILP Miner comes up with the result shown in Figure 33 which is slightly different. On the one hand it is simpler to read, on the other hand it allows for more behavior than recorded in the event log.

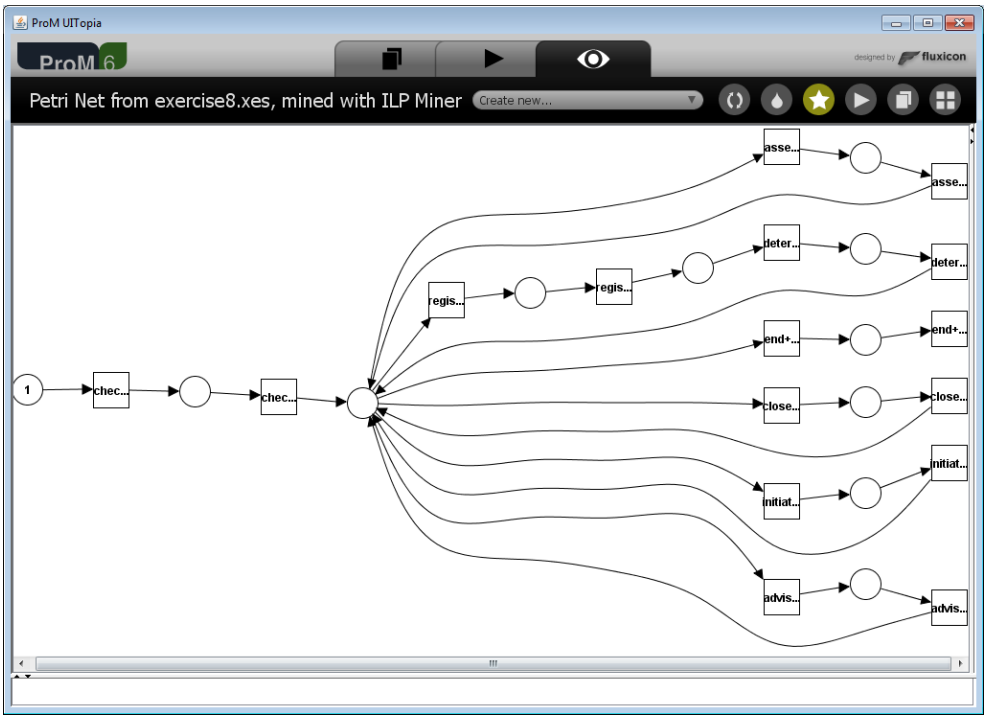

Figure 33: ILP Miner result on exercise6.xes.

4. *Run the Fuzzy Miner on the event log, using the default settings. Now try to answer the following questions:*

Running the Fuzzy Miner with the default settings will result in the model as shown in Figure 34.

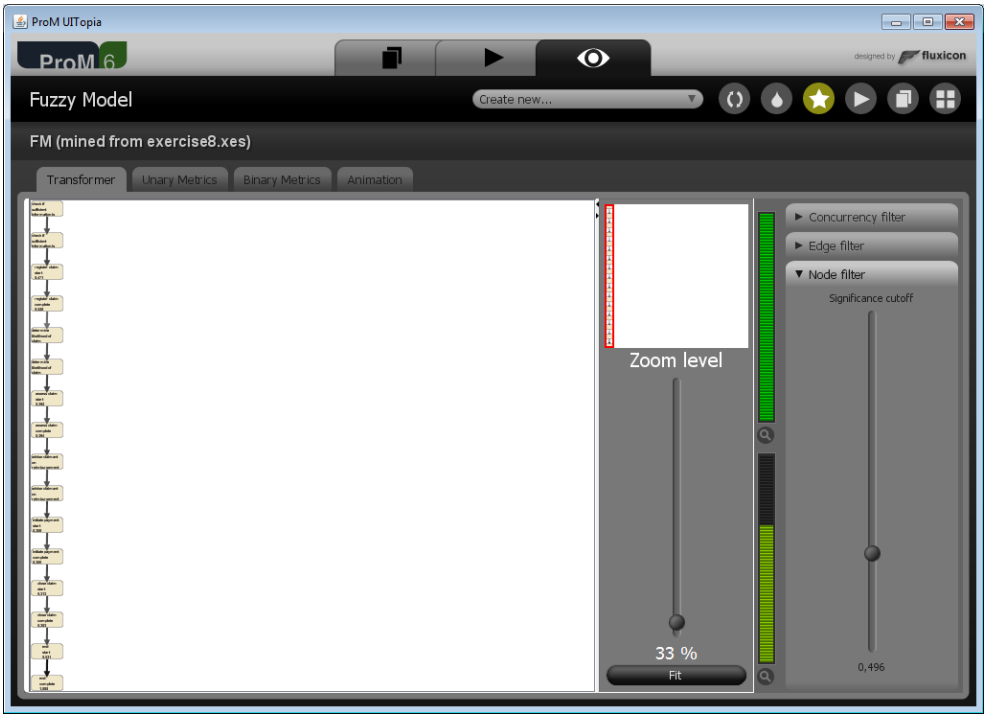

Figure 34: Fuzzy Miner result on exercise6.xes.

(a) *What is remarkable about the resulting process model compared to the models discovered by the other plug-ins?*
The resulting process model is entirely sequential while the other process models discovered from the same log are not.

(b) *The Fuzzy Miner tries to aggregate low frequent activities into clusters and only show edges which are significant enough. Using this brief explanation, can you indicate why the resulting process model looks so different?*
The Fuzzy Miner removes edges which are considered to be not relevant enough to show.

(c) *Read the documentation on the Fuzzy Miner which can be found at* `http://prom.win.tue.nl/research/wiki/online/fuzzyminer`. *Please note that this describes the ProM 5 version of the plug-in so step 2 in the* preparation *section is not applicable.*

(d) *Play around with the settings of the Fuzzy Miner (the sliders you see on the right). What do you need to change to get closer to the models as discovered by the other plug-ins?*
The most important setting you need to change is the `Edge cutoff` as shown in Figure 35. Changing other settings might also improve the result.

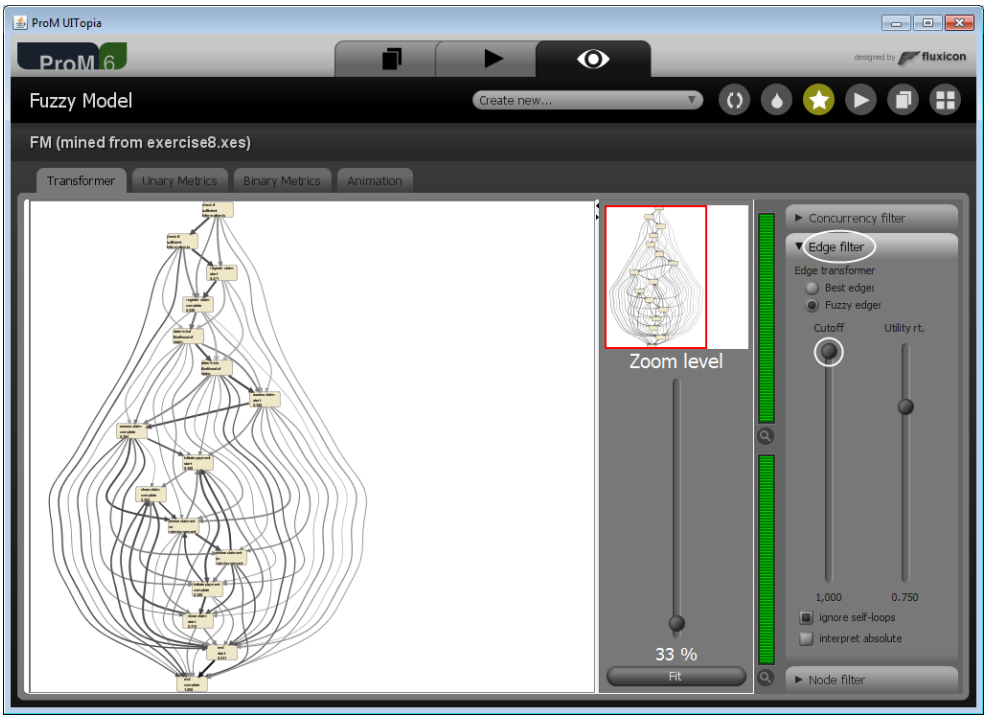

Figure 35: Fuzzy Miner result on exercise6.xes with `Edge cutoff` set to 1,000.

(e) *Why do you think this model is still not exactly the same as the original model?*
The Fuzzy Miner takes many metrics into account when deriving nodes and edges, most of these are not used by other plug-ins. All metrics used by the Fuzzy Miner can be seen and changed by selecting the `Metrics` tab when starting the Fuzzy Miner

(f) *On what kind of logs would the Fuzzy Miner be useful?*
The Fuzzy Miner is particularly useful on large, unstructured and/or noisy logs.

(g) *Can you now think of some benefits but also some weak points of the Fuzzy Miner?*
The main benefit is that it can cope with unstructured event logs. The main weak point is that the model discovered by the plug-in does not really contain any semantics and only indicates 'relationships' between nodes/events.

5. *Run the Transition System Miner on the event log. Be sure to set the* ***`backward keys`*** *to both* ***`Event Name`*** *and* ***`Lifecycle transition`*** *and not* ***`Resource`****, as is shown in Figure 1.*

6. *From the resulting transition system, press the* ***`Play`*** *button on the top right. Select the* ***`Transition system analyzer`*** *plug-in. Now click the* ***`Log`*** *object in the* ***`input`*** *column and select exercise6.xes. Now run the plug-in (press* ***`Start`****) and try to answer the following questions:*

(a) *How long does a case take on average? And what are the minimal and maximal durations recorded? On how many ways can you get this information from this plug-in?*
You can answer this question if you select the first state and then inspect the metrics, as shown in Figure 36. The remaining time after the first state is 1118 days and 4 hours on average. The minimal remaining time is 8 days while the longest remaining time is 4462 days. Another option is selecting the last state of the transition system and looking at the `elapsed` row in the metrics table.

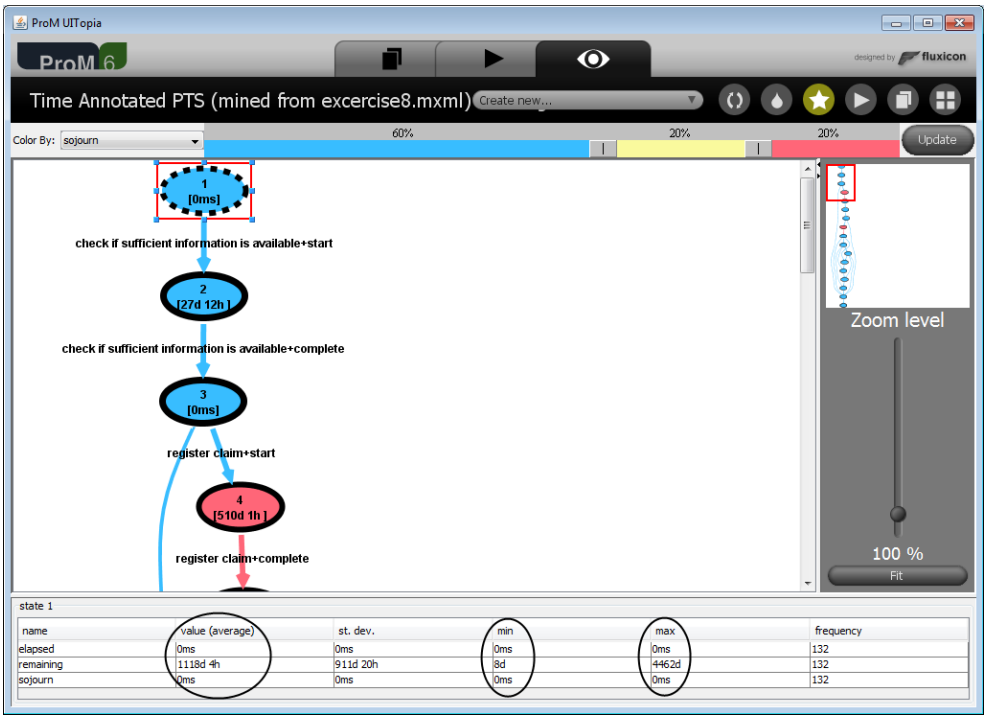

Figure 36: Transition System Analysis result with the first state selected.

(b) *What do the red states indicate? And what do the red arrows indicate?*
States and arcs are colored blue, yellow or red depending on the relative time spend there. In Figure 37 state 4 is red and selected. On the bottom of the screen some metrics are shown. Here we can see that the sojourn time is very high, hence the red color of the state. The red arrow is also colored red for the same reason. Apparently the time between starting and completing the registration of a claim takes 510 days on average. State 8, assessment of a claim, also takes a long time, 599 days on average.

(c) *On the top left, change the* `Color By:` *setting from* `Sojourn` *to* `elapsed`*. Why is part of the process colored red and why are the last two events yellow?*
In Figure 38 this option is chosen and the second-last state is selected. The last two events are colored yellow because there are different ways to get there. A

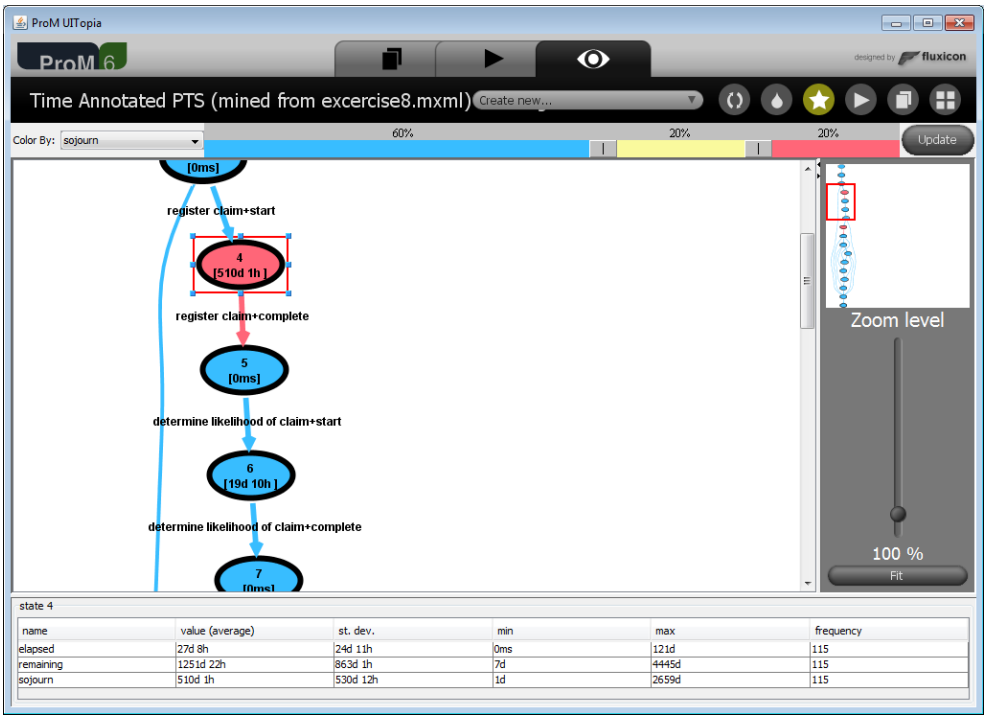

Figure 37: Transition System Analysis result with state 4 selected.

case is for instance already ended when insufficient information is available or when the likelihood of the claim is deemed to low/high (we don't know that). The part that is colored red is the path that a handled claim takes which takes longer than to be rejected.

(d) *If we change the setting to* ***`remaining`*** *we see the first events colored red and 1 event colored yellow. Can you explain this? And why do you think that the outgoing arrow of the yellow event is colored red?*
Figure 39 shows the result of this option with the one yellow state, state 8, selected. Of course, now the first part of the process is red because it can take a long time before the process is completed. State 8 is colored yellow because activity `Assess Claim` can take quite a while but does not qualify for the color red yet.

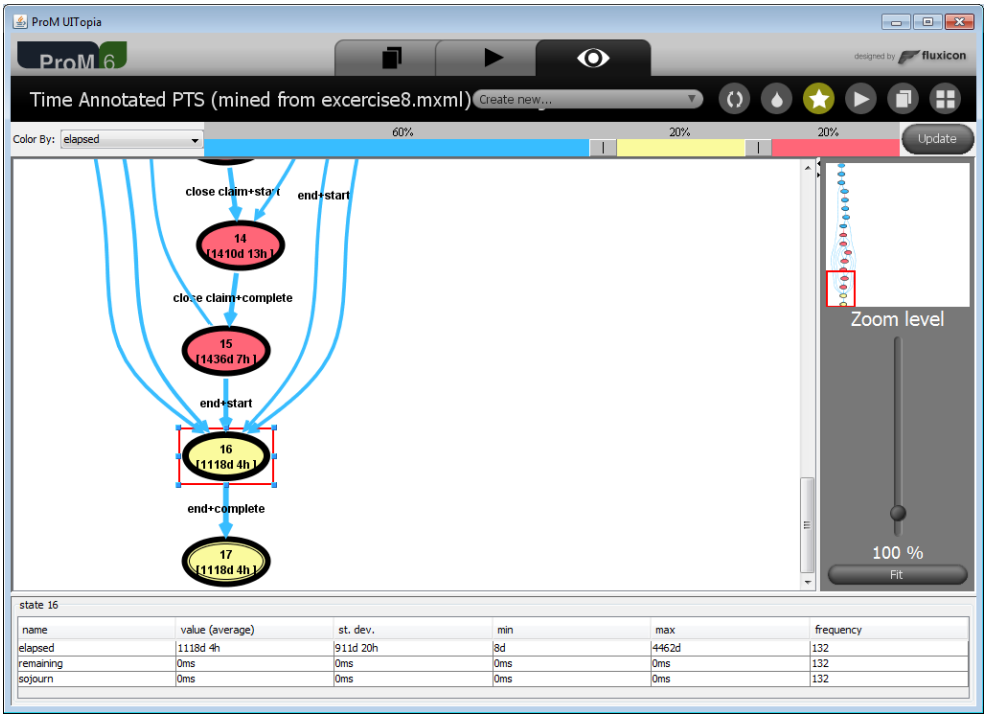

Figure 38: Transition System Miner result with colors by elapsed time with the second-last state selected.

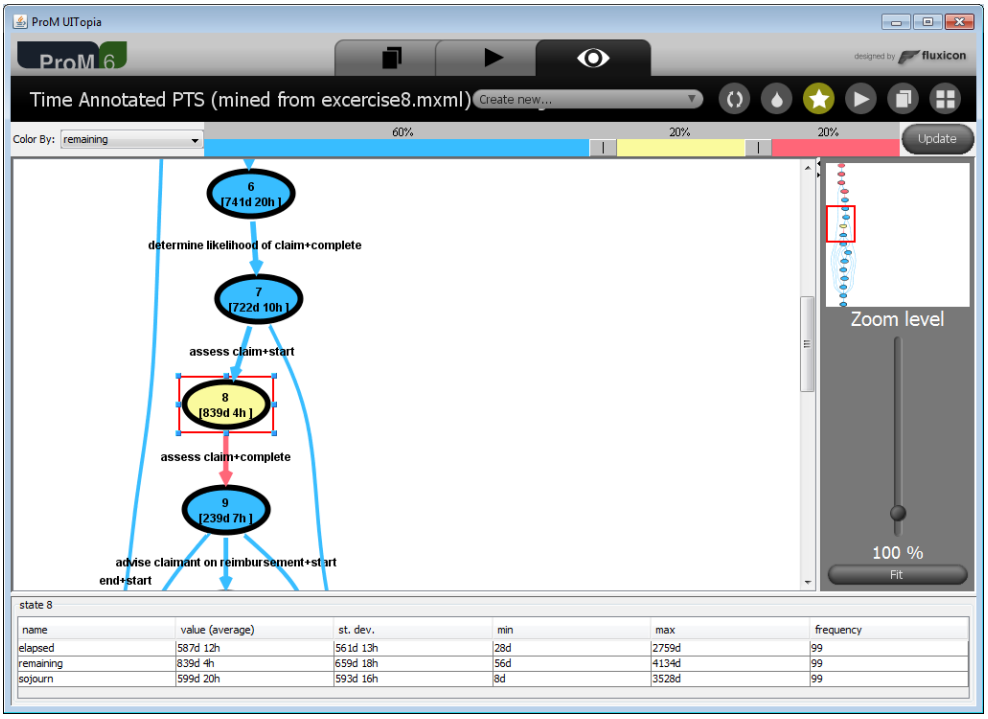

Figure 39: Transition System Miner result with colors by remaining time and with state 8 selected.