YAWL generated process log analyzed with Disco

Project - Process Mining

Prof. Dr. Hense

Author	Date	Version
Lucas Prochnow, Frank Thielen, Marc Stammerjohann	03.07.2015	1.0

Contact:
lucasprochnow@gmx.net
thielen.frank91@gmail.com
marcstammerjohann@web.de

1. Introduction

In the context of our university project, “Process Mining” at Hochschule Bonn-Rhein-Sieg, we are executing processes with YAWL and using Disco to analyze the process logs.

In this article we are describing, how we analyzed a compact YAWL generated process log with Disco. First of all we like to explain the YAWL process and the analysis of the resulting process log with Disco. Furthermore, we will describe our self-written java tool for replacing YAWL generic resources.

2. YAWL Process

Before we are going into the program details we want to introduce our simplified restaurant process depicted in the Figure 2.1. It describes how a waitress takes an order, a cook prepares the selected meal and the waitress finally serves it. The guests are able to choose a burger, a pizza or pasta. As you can see the task „take an order“ has got a XOR-Split, this means just one of the options can be picked by the guest with each order.

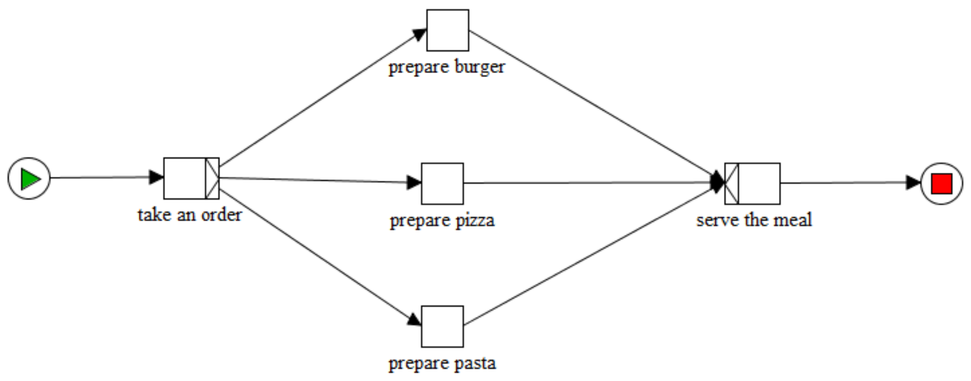


Fig. 2.1. YAWL restaurant process

2.1 Executing Process

Our goal is to have a really accurate log file. This is the reason why we execute the workflow ten times which leads to an exact percentage result. The following table shows how many times each task is going to be executed.

Task	**Number of executions**	**Percentage**	**Role**
take an order	10	100%	waitress (Tim)
prepare burger	3	30%	cook (Paul)
prepare pizza	6	60%	cook (Paul)
prepare pasta	1	10%	cook (Paul)
serve the meal	10	100%	waitress (Tim)

As mentioned before we got two different roles, the waitress Tim and the cook Paul, in the YAWL execution process. Each activity is linked to one of the roles. Due to the process model and the including XOR-Split the waitress owns 66% of the task in one workflow execution and the cook the other 33%.

2.2 Process Log

In Figure 2.1. each activity appears only once in the process. However, during the execution YAWL adds a start and a complete tag to the activities.
Looking into the log file, each activity appears twice in the process, but they can be distinguished by their added tag. For example the activity “take an order” can be found twice in Figure 2.2.

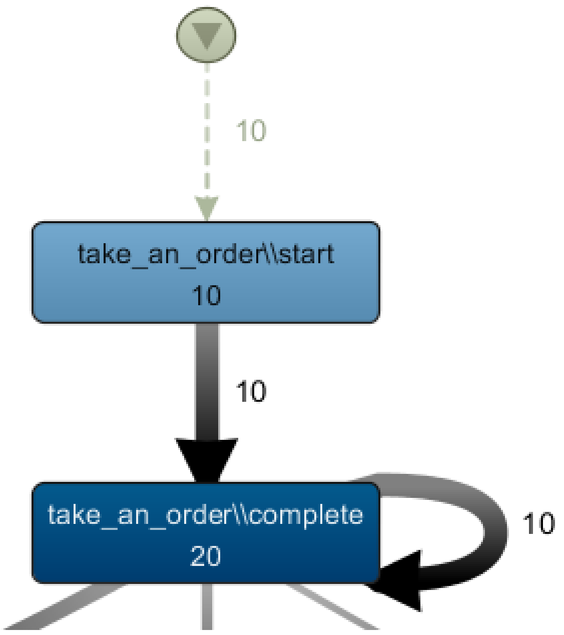
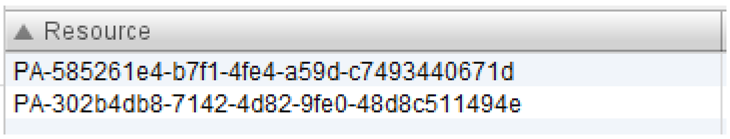


Fig. 2.2. Disco process map - start and complete activity

2.3 Replace Resource ID

In the YAWL workflow editor we specified which activity can be executed by which role. When our process is started in the engine every person, who owns the assigned role, is able to start this task.
Every interaction with a task leads to an entry in the log file. The specific resource (in our scenario would be Tim the waitress or Paul the cook) is linked by a generic resource string without any readable information about the role name or either the resource name.

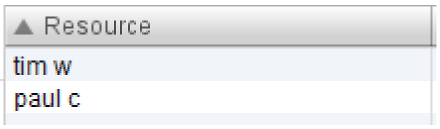


Resource
PA-585261e4-b7f1-4fe4-a59d-c7493440671d
PA-302b4db8-7142-4d82-9fe0-48d8c511494e

Fig. 2.3. The generic resource string from YAWL.

If we would not replace the generic strings in Figure 2.3. with the resource names, we won’t see which resources are handing the activity to another.

To deal with this problem we are going to use a little tool, which takes the log file as an input, connects to the YAWL resource service and replaces every generic resource string with the resource full name. See capital 2.4 for detailed information about the java tool.



▲ Resource
tim w
paul c

Fig. 2.4. The generic resource string is replaced by the resource name.

2.4 Java Tool for replacing YAWL Generic Resources

The problem we were facing: Every event logged by YAWL has a resource attribute. At this place we actually expected the YAWL user who executed the activity. Nevertheless the .xes-log we exported contained only a generic created Participant-ID.

To solve this problem we wrote a short java application, which search through the log for generic Participant-IDs and replace them with the real usernames from the YAWL-Server. For this workaround the program needs a connection to the server including the IP-address, port and a valid user with admin rights. Naturally you also need the path to the .xes-file, which should be edited.

Before using this tool in a productive environment you should check data protection regulations.

3. Preparation of the Process Log with Disco

3.1 Process Map

After the execution is done, YAWL provides a log file, which can be imported by Disco. In comparison to Figure 2.1. the process map in Disco (shown in Figure 3.1) contains twice as much activities and self-calls for each complete task.

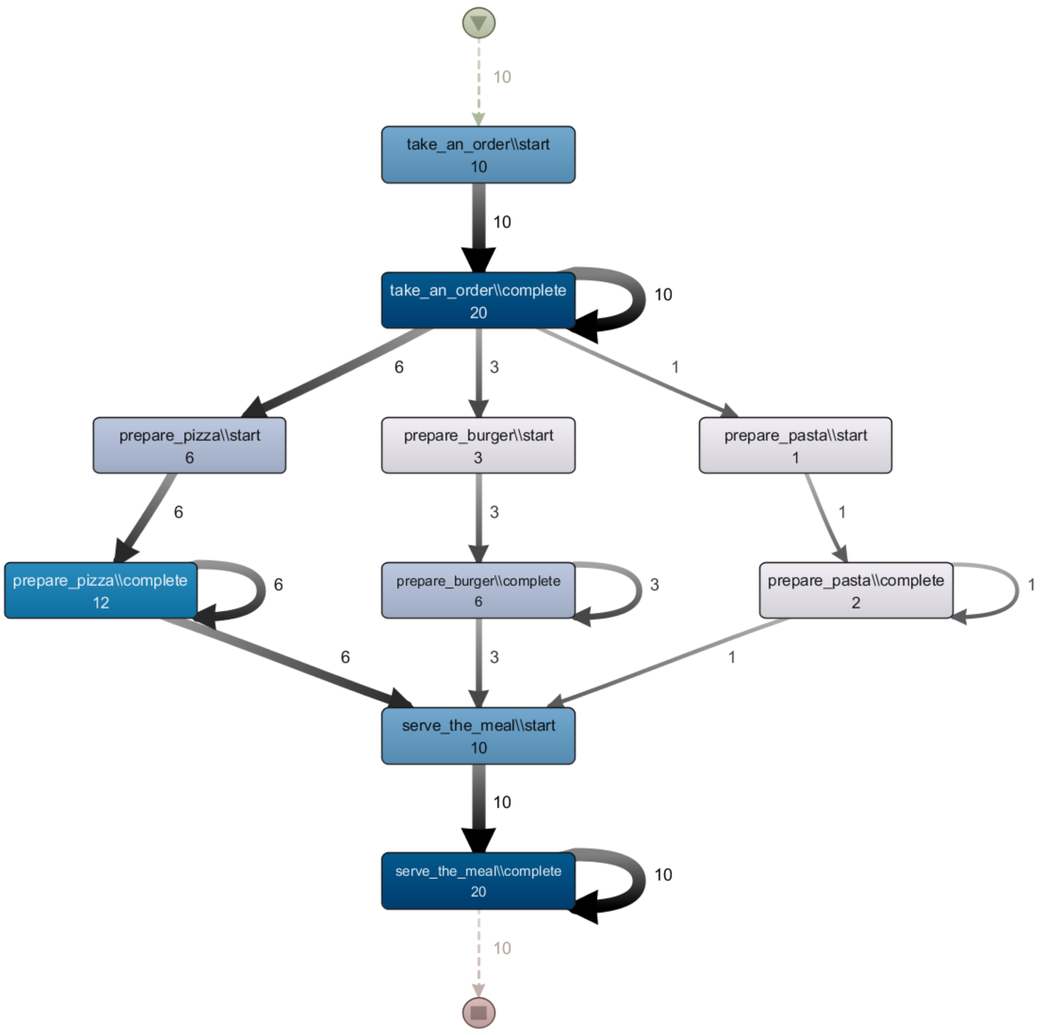
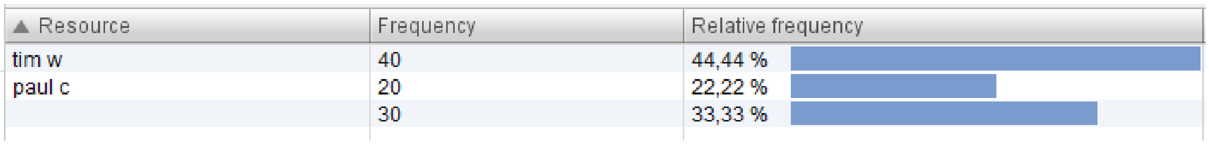


Fig. 3.1. The whole process map in Disco without any filters.

3.2 Filter Process

The following descriptions are based on the process log with the replaced resource names.

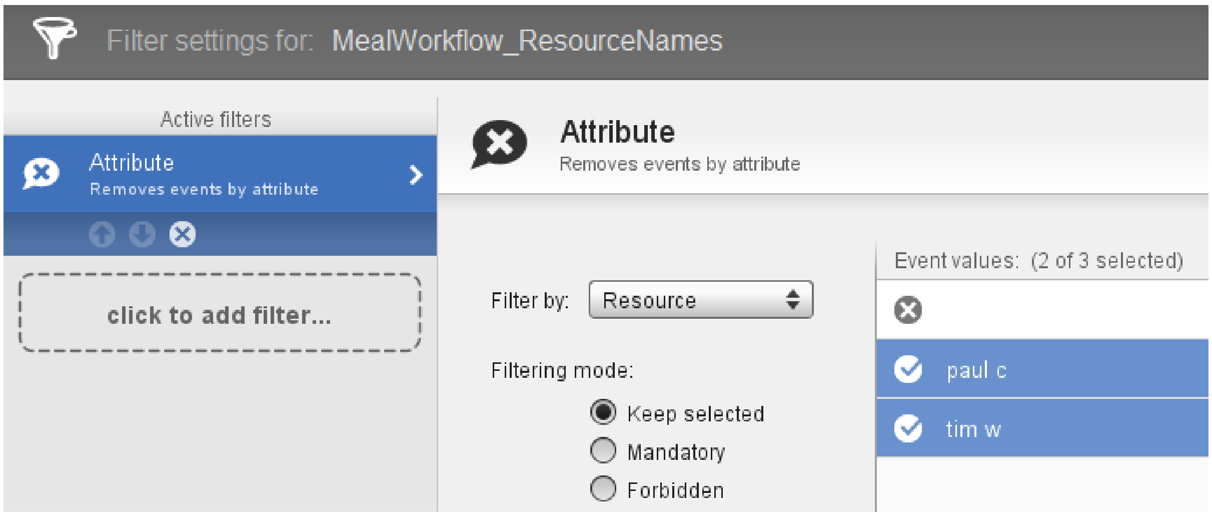


▲ Resource	Frequency	Relative frequency
tim w	40	44,44 %
paul c	20	22,22 %
	30	33,33 %

Fig. 3.2. Unknown resource.

The relative frequencies are not as we expected. Figure 3.2. shows an additional resource with a frequency of 30, which has no resource id either a name.

Disco provides different filter functions, to exclude the unknown resource represented in Figure 3.3.


Filter settings for: MealWorkflow_ResourceNames

Active filters

Attribute
Removes events by attribute

click to add filter...

Attribute
Removes events by attribute

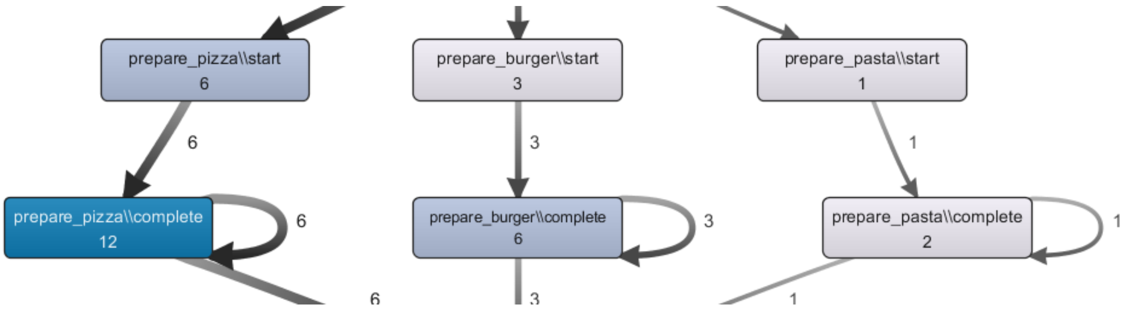
Filter by: Resource

Filtering mode:
- [✓ Keep selected]
- Mandatory
- Forbidden

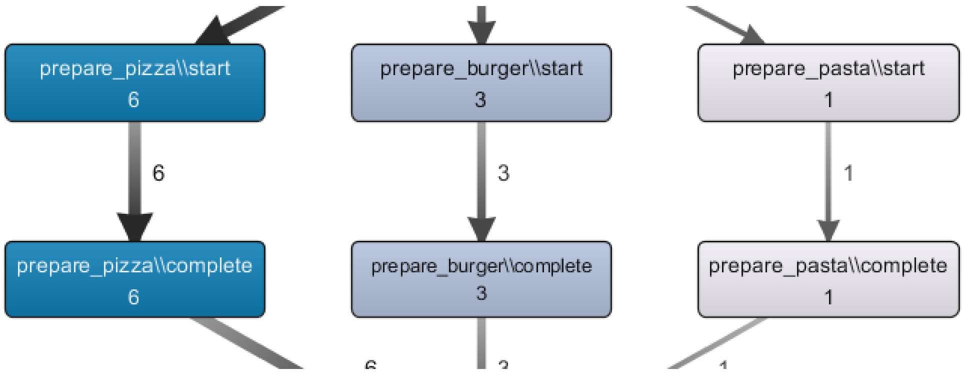
Event values: (2 of 3 selected)
- ✓ paul c
- ✓ tim w

Fig. 3.3. Adding a new filter to exclude the unknown resource.

The result of the filter is reflected in the following process maps.



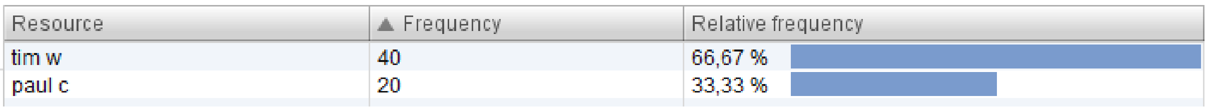
(a) Before filter was added.



(b) After filter was added.

Fig. 3.4. Comparison of the normal and filtered process map.

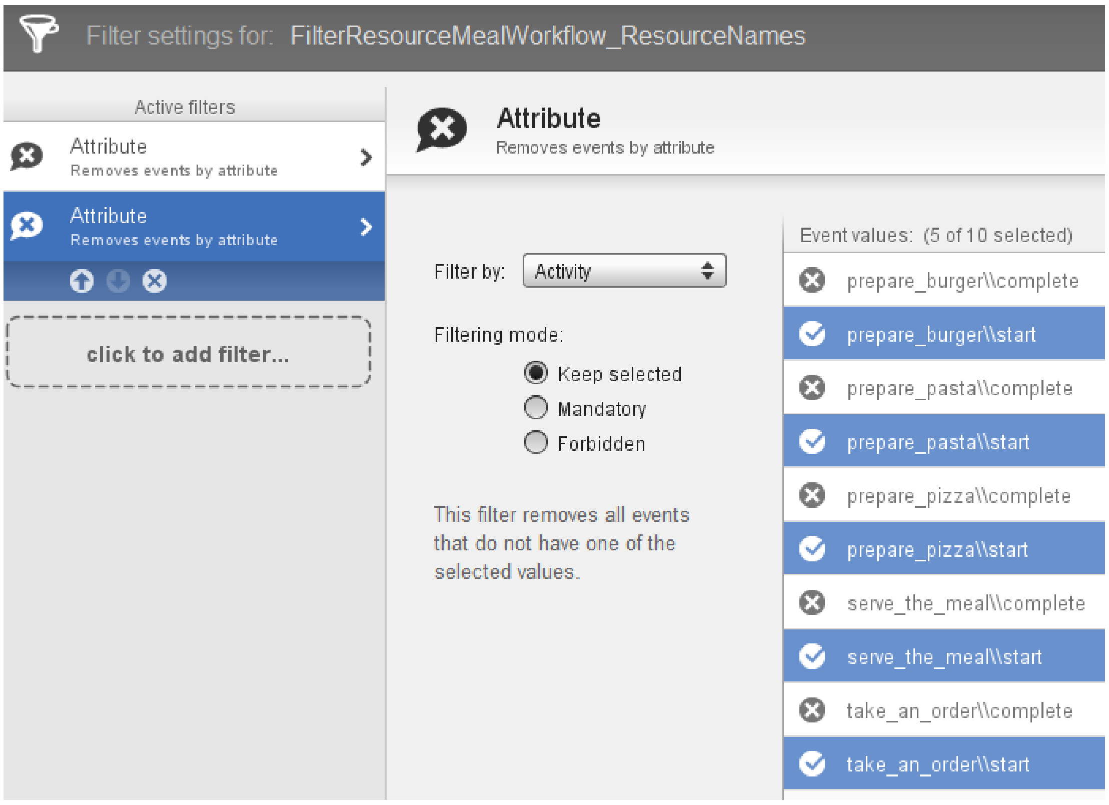
The empty resource represents the self-call of the complete activities. After adding the filter in Disco, the self-calls are removed. Furthermore, the relative frequency in Figure 3.5. matches the above described distribution in chapter 2.1.



Resource	▲ Frequency	Relative frequency
tim w	40	66,67 %
paul c	20	33,33 %

Fig. 3.5. Expected relative frequency.

The process map still contains more activities, as designed in the YAWL process in Figure 2.1. In addition, we added a filter to ignore the complete activities.


Filter settings for: FilterResourceMealWorkflow_ResourceNames

Active filters

Attribute
Removes events by attribute

Attribute
Removes events by attribute

click to add filter...

Attribute
Removes events by attribute

Filter by: Activity

Filtering mode:
[✓ Keep selected]
Mandatory
Forbidden

This filter removes all events that do not have one of the selected values.

Event values: (5 of 10 selected)

	prepare_burger\\complete
✓	prepare_burger\\start
	prepare_pasta\\complete
✓	prepare_pasta\\start
	prepare_pizza\\complete
✓	prepare_pizza\\start
	serve_the_meal\\complete
✓	serve_the_meal\\start
	take_an_order\\complete
✓	take_an_order\\start

Fig. 3.6. Adding a new filter to ignore the complete activities.

The process map generated with Disco matches with the number of activities and the chosen paths of the designed YAWL process.

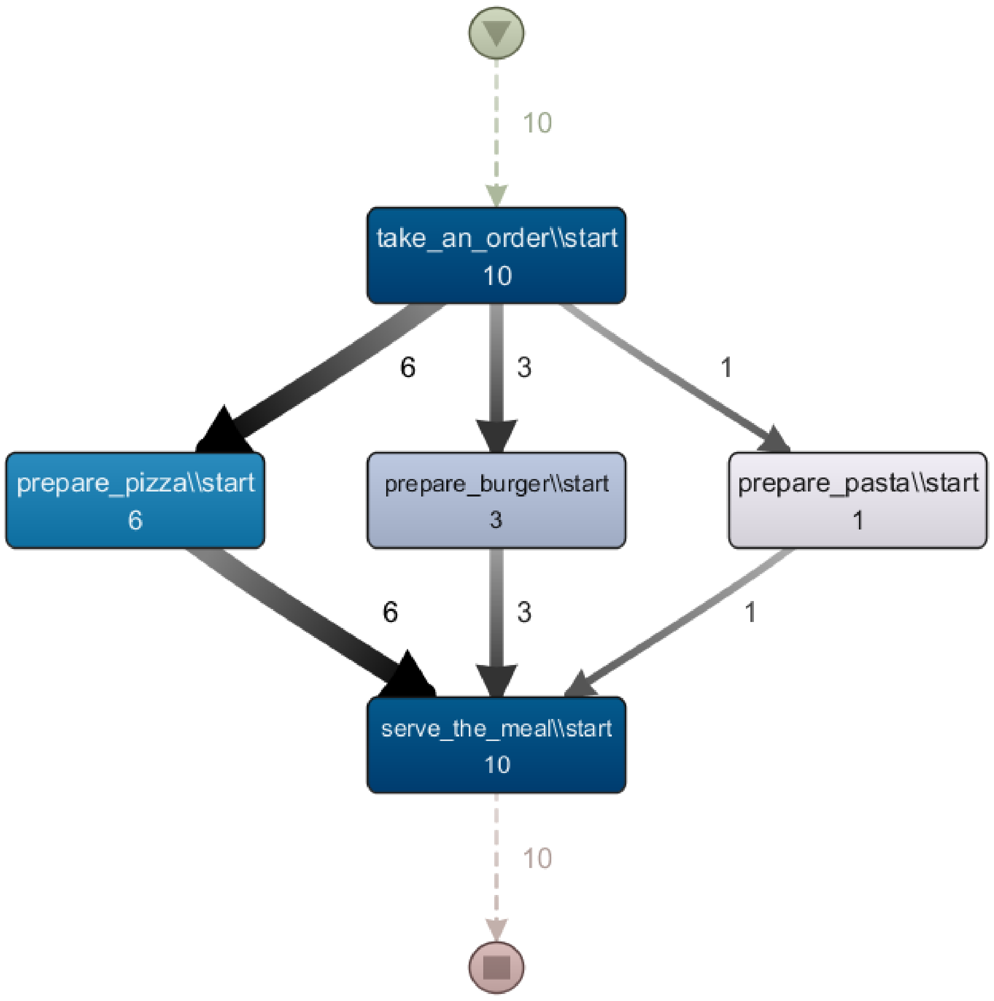
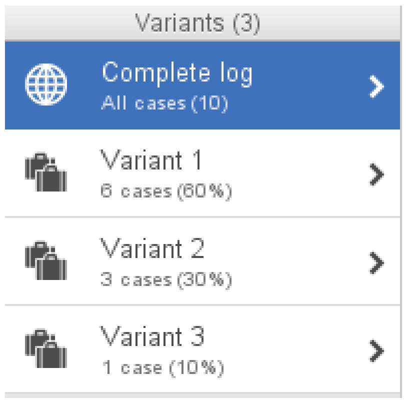


Fig. 3.7. Filtered process map in Disco.

3.3 Variants

The process has three different paths to be executed. Each path was executed exactly as described. Disco provides all variants in the workflow within the Cases View. A variant is a collection of cases, which has the same activity execution order.



Variants (3)

Complete log
All cases (10)

Variant 1
6 cases (60%)

Variant 2
3 cases (30%)

Variant 3
1 case (10%)

Fig. 3.8. Variants overview.

Variants	**XOR-Tasks**	**Planned**	**Actual**
Variant 1	Prepare Pizza	60%	60%
Variant 2	Prepare Burger	30%	30%
Variant 3	Prepare Pasta	10%	10%

The different path variants are correctly displayed in Disco. The variants are very important for adjusting the detail level of the process map.

4. Scaling

Disco provides an adjusting of the detail level of the process map. Two sliders are available as depicted in Figure 4.1.

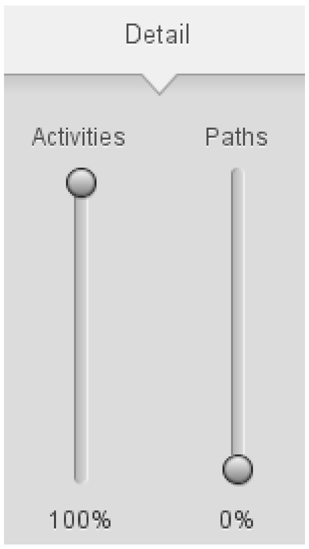


Fig. 4.1. Process map detail scaling

The activity slider affects the shown activities. A higher activity detail level is displaying more variants in the process map. At a level of 100 percentage all variants will be displayed. Only the variant with the highest ratio is shown at 0 percentage.

The path regulation is depending on the currently displayed activities. Selecting 0 percentage will ignore the connections with a low frequency. Same as for the activity slider, setting the path slider to 100 percentage will allow all connections to appear in the process map.

4.1 Adjusting the Level of Detail in a Process Map

The path slider has no effect on our process example, due to our strict process design.

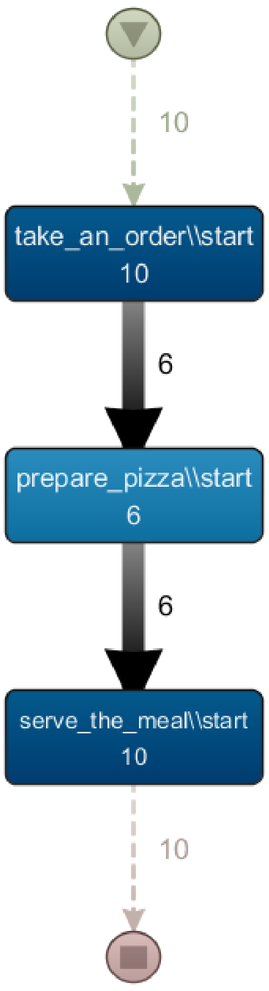


Fig. 4.2. Level of Detail - Activities 0% and Paths 0%. Only variant 1 (prepare pizza) is visible.

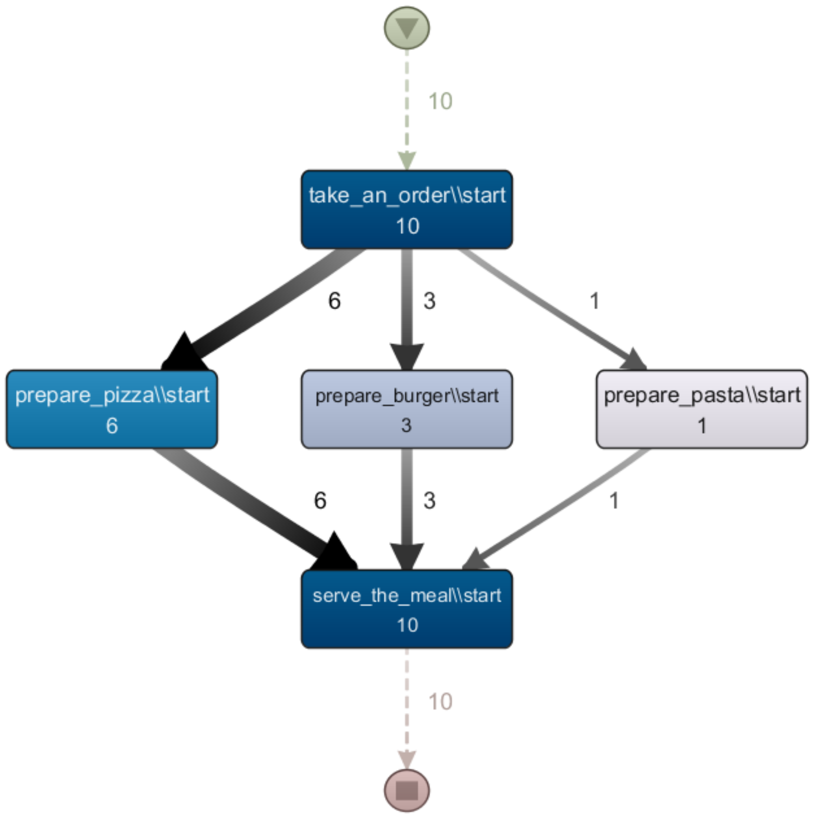


Fig. 4.3. Level of Detail - Activities 33% and Paths 0%. All variants are visible.

Unfortunately our compact process has a fluent passage from one variant to all three variants. In other words the process map will only change once at 33%, though the process includes three different variants.

5. Conclusion

The following advices are very useful for analyzing a YAWL generated process log with Disco.

1. Replace generic resource ids with the resource names
2. Disco filter options
    a. Exclude the unknown resource (self-calls from YAWL)
    b. Ignore the complete activities (tag from YAWL)
3. Scaling the process map
    a. first adjust the activity slider, until all necessary activities are shown
    b. then adapt the path slider (be careful process map might become overwhelming)