Proposal, suggestions and comments

1. Shall we use the wf4ever namespace, or could we create a new one? My concern using the wf4ever namespace is that the effort will look more like a product from the project than a product of Pinar's and Daniel's thesis.

It will be used in the project, yes. But we could create a purl like "scientific workflos motifs". This is just a suggestion, if you two think that we should use the wf4ever ns then I won't oppose.

1. Examples must be given for each motif. If possible with pointers to the real workflows.

We have them in the spreadsheets and presentation.

1. We could add a superclass of both DataMotif and WorkflowMotif called Motif, with the definition we gave in the presentation/paper.
2. No semantic restrictions are defined among the motifs. I think we could add some disjoints. For example, can a data motif be a workflow motif? Filter and Merge are disjoint, input augmentation and output splitting as well, etc. However others like filtering and Filter and data cleaning could overlap. We need to analyze this.
3. I have been thinking about how to model the relationships hasDataMotif and hasWorkflowMotif. This is what I found out, and what I suggest: (of course, it is open to discussion if you don't agree)
   1. We need this properties, that is for sure. My first concern is that a workflow step does not HAVE a workflow motif. It CORRESPONDS to a workflow motif, right? Thus I would change the properties to correspondsToWorkflowMotif and correspondToDataMotif.
   2. We should add a superproperty called "correspondsToMotif". Why? Because I think it makes easier the annotation. Imagine the user knows the Motif but can't access the ontology. It would save him/her to go and look it out.
   3. The properties need domain and range. As I stated in our discussion on email, it is a good practice for designing ontologies (and it is something that the people of my group put a lot of enphasis on). It also preserves the semantics (it restricts the different kinds of assertions) and prevents having something like:   
      :khalid motifs:correspondsToMotif :a1.

The next figure summarizes all my suggestions:



Rationale:

1. Why wfdesc:Process as domain for corresponds to Motif? Wfdesc:Process is a quite general concept. It includes workflows, steps, scripts, etc. If we want to assert that they have a motif but we are not sure whether they should be considered a workflow or a step, the just use this relationship. It would be like the “easy” way to annotate resources.
2. Why motif:Step is necessary? Wfdesc:Process is too general. If we pick this relationship as domain for both hasWorkflowMotif and hasDataMotif then all steps could have asserted motifs that are for workflows. Some steps can be workflows, but not all steps are workflows. That is why I propose to create this concept (or reuse from somewhere else). It is a subclass of wfdesc:Process, and it is not disjoint with wfdesc:Workflow.
3. What is the relation between wfdesc:Workflow and motif:Step. It is out of the scope. We just want to declare the difference between steps and workflows, that is all. They are not disjoint
4. What is the relation between motif:DataMotif and motif:WorkflowMotif? They are not disjoint

Example of annotation: Given a sample workflow we can annotate either Steps or Workflows. We can say that various steps correspond to the same motif instance or that a step may have one or more motifs. If you don’t want to be that precise about the domain, just use de top level relationship: correspondsToMotif.

