Evaluating the Cognitive Dimensions of FlowiXML

Josefina Guerrero-Garcia[1,2] Juan Gonzalez-Calleros[1,2]
[1]Facultad de Ciencias Computacionales
Benemérita Universidad Autónoma de Puebla
Puebla, Mexico,
jguerrero,juan.gonzalez@cs.buap.mx
[2]Research and Development Unit, Estrategia 360
Puebla, Mexico

Jean Vanderdonckt[3], Jaime Muñoz-Arteaga[4]
[3]Louvain School of Management
Université catholique de Louvain (UCL)
Louvain-la-Neuve, Belgium
[4]Sistemas de Información
Universidad Autónoma de Aguascalientes (UAA)
Aguascalientes, Mexico

Abstract—**supporting business processes through the help of workflow systems is a necessary prerequisite for many companies to stay competitive. An important task is the specification of workflow, i.e. the parts of a business process that can be supported by a computer-based system. We investigated how to close the gap between the organization requirements and the development of information systems to support them. We introduced FlowiXML a methodology for developing user interfaces for a workflow information system in a systematic way. The methodology provides designers with methodological guidance on how to derive user interfaces of workflow information. We have already experienced the benefits of FlowiXML in several real life case studies conducted at the University. In this paper we report on our evaluation of FlowiXML against the cognitive dimension framework.**

Workflow Information Systems; Bussiness Process; Cognitive Evaluation; model-based development

I. INTRODUCTION

Workflow Information Systems (WfIS) are considered vital in any organization, they do not have necessarily the expected impact on carrying out interactive and non-interactive tasks of these organizations. Many causes may explain this lack of exploitation: tasks were developed with the implicit assumption that they would primarily be performed by people, an organizational structure would be developed in which particular tasks are allocated to groups of users. Only then people consider whether computers – or rather, ISs – could partially support, or even take over, the work. Information technology professionals understand that there is a divide between the business side, with its business requirements, and the support that is being provided to address these requirements. Looking for ways to bridge this gap, a workflow could be considered as an appropriate mean to address this need. There has been a growing interest in Workflow Management Systems and flexible workflow support. However, when defining a workflow, software rarely supports designers in creating User Interfaces (UIs) corresponding to this workflow, i.e. the UI that helps end users to carry out their interactive work. WfIS is a concept that we introduced [7] to define the use of workflow technology for IS development, focusing on the UI development.

For several years we have worked in a methodology for developing WfIS called FlowiXML. The FlowiXML methodology has been described in our former work [7, 8, 9, 10, 11, 12, 13]. For the development of FlowiXML we conducted a thorough research and studied several other existing graphical notations. As our current experience, and the future applications allow us to conclude that the FlowiXML approach looks promising, we decided to perform a more formal evaluation of the current version of FlowiXML.

We rely on the cognitive dimension framework (CDF) [5] for the evaluation of the graphical notation of FlowiXML. The CDF is a tool to aid: non-HCI specialists in evaluating usability of information-based artifacts (summative evaluation); and designers to prompt possible improvements (formative evaluation) [6]. The name artifact refers to a combination of three elements: the notation, environment and medium. In our case the notation is consistent within the environment, and the medium is always a computer screen. Thus we strive for a formative evaluation, i.e., evaluating FlowiXML designs with respect to the impact that they will have on its users.

The chapter is structured as follows: next section describes FlowiXML its functionality and structure; then, the CDF is presented and its relevance to FlowiXML is motivated; section 4 evaluates FlowiXML against the fourteen dimensions of the CDF; finally, this chapter ends with the conclusion of this work. In the reminder of this chapter, terms from the CDF are written in italics just to be compliant with its vocabulary.

II. DEVELOPING WORKFLOW INFORMATION SYSTEMS WITH FLOWIXML: AN OVERVIEW

FlowiXML [7] is a method that provides means to specify a Workflow Information System (WfIS). It is composed on three major steps:

- Workflow information system requirements. This is the result of the elicitation of the organization, getting information by different means, such as: interviews, direct observation. The result serves as input to identify workflow elements.
- Workflow information system design. This step includes modeling of: workflow, organizational units, jobs, user stereotypes, processes, workflow allocation patterns and tasks. Mapping the workflow specification into a workflow information system.
- Workflow information system development. We consider the development of UI for: task models, allocation patterns, agendas, worklist. We include the

implementation of a workflow manager system that is based on the workflow designed in the previous step.

A. *Workflow Information System Requirements*

In a user-centered design it is expected to identify the end-user and get their description for the work they do. This early understanding of the work, which is carried out in the organization, requires constant meetings with end user to understand their tasks in their context (environment, available resources). During the stage of system requirements gathering, model elicitation is aimed at identifying in textual scenarios elements that are relevant for building a first version of models that will be further exploited in our method.

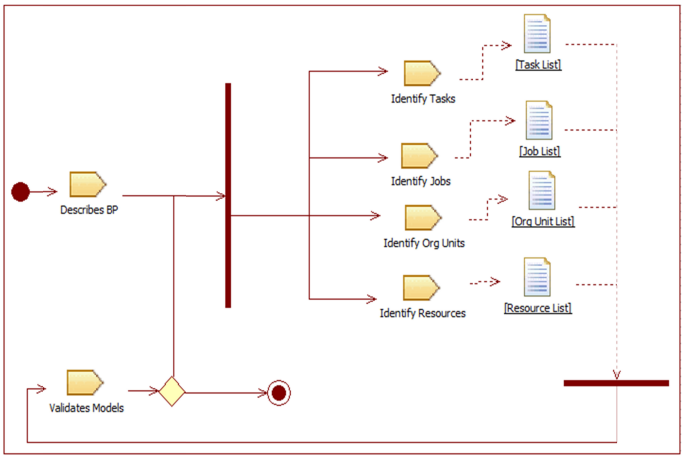

Figure 1. Resource allocation patterns selection.

The WfIS requirements step (Figure 1) is concerned with the understanding of a problem by studying an existing organizational setting; the emphasis is put on identifying the elements involved in the business process description following identification criteria. The output of this phase is an organizational model, i.e., lists for: task, job, organizational unit, resource, which includes relevant actors and their respective tasks. This step uses as guidance three documents: task-meta-models definitions if there a need to look for a definition of a concept; task identification criteria that to identify tasks in the scenarios; and the most relevant work describing this step has been described in [13].

On practical bases, the domain expert (end-user) describes the business process, from this scenario the workflow designer identifies the tasks, the resources in charge to develop them, the unit where they are executed, and so on. After, he produces a classification of these concepts, which will be validated for the domain expert (end-user, workflow manager, and supervisor).

B. *Workflow Information System Design*

The WfIS design is composed of several models (workflow, process and task) and several actors are involved (Wf designer, end-user, domain expert and Wf manager). The first activity corresponds to the consolidation of the concepts identified in the elicitation of the scenario. The Wf designer is able to start the modeling:

- *Top-down approach*, i.e., starting with modeling the high level view of the problem (workflow model), then detailing the workflow models by adding process into the different organizational units. Finally a process model is refined by adding a task model on each task.
- *Bottom-Up approach*, i.e., starting with low level details (task modeling) and from that, starts to build process models and workflow models.
- *Middle out approach*, i.e., starting with middle level details (process modeling) then going to details with task modeling, and high level description with workflow.

The end-user is the responsible of validating task models. Notice by end-user we understand the person who actually is in charge of performing this task, i.e., the most qualified to say something about it. The process is validated by the domain expert. This is due to the fact that a process requires a higher level of understanding of the problem. In this view it could be any person in the organization that through the requirements analysis has been identified to be most familiar with the whole process modeled.

Finally the workflow model is validated by the Wf manager, who is the person or group of persons with an understanding of the whole workflow when processes are grouped. It is also, the Wf manager who is in charge of allocating tasks in a process to resources. The final result is the WIS design. All these design activities are also accompanied with guidance for the modeling activities. The ordering of the modeling activities does not follow any recommended order. We aimed at answering with the modeling activities to the questions related to the work done in the organization: what to do? How to do it? Where to do it? Who will carry out it? Whom?

The question "What to do?" is answered with the definition of a process indicates which tasks must be performed and in what order. Thus answering the question what to do? After having identified tasks that are part of a process then they have to be related to each other by means of process operators. We propose the use of Petri Nets notation for modeling processes. As guidance for process modeling, designers must rely on: Petri Nets Structure Rules [18], Identification Criteria [10], the WF Modeling Guideline by Example material available at (http://www.usixml.org/index.php?mod=pages&id=40).

The question "How to do it?" is answered with task modeling. For each task in a process a task model can be specified, not necessarily, to de-scribe in detail how the task is performed. By exploiting task model descriptions different scenarios could be conducted. Each scenario represents a particular sequence of actions that can successfully be performed to reach a goal. Task models do not impose any particular implementation so that user tasks can be better analyzed without implementation constraints. Our task model (Figure 2) represents a decomposition of tasks into sub-tasks linked with task relationships. It is an extended version of UsiXML task modeling [7] and compliant with the graphical notation of CTT [16].

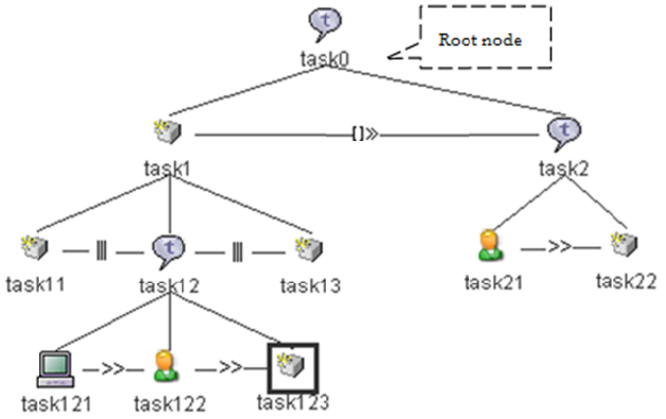

Figure 2. Task model representation.

The question "Where to do it?" is answered with the organizational unit identification. We introduced an organizational unit concept; it describes the places where work is carried out. It can be a physical chamber in a builder (e.g. an operating room), but it can also be a complete floor in a hospital. The elements corresponding to the actual organizational structure in a specific organization are specified during the organization design. This part contributes to UI adaptation to different categories of users and security of IS by blocking access to UIs when the user does not have the permission to perform the task.

To answer the question "Who will carry out the work?" jobs and users are identified. This step consists in the description of the resources that are capable of doing work. In addition, a resource may have one or more associated jobs. Jobs serve as another grouping mechanism for resources with similar roles or responsibilities. A resource is a member of an organizational unit; he is owner of an agenda which using to manage the whole range of his tasks.

The question "to whom is actually allocated the task?" is answered through the definition of resource allocation patterns. Actual assignment of tasks to resources is performed according to workflow resource allocation patterns [17]. They are applied to different steps of the task life cycle: creation, allocation, in execution. Without considering the moment in which they are applied, designers must define how to allocate work to available resources. We use the 42 resources patterns that have been identified by Russell.

C. *Workflow Information System Implementation*

The last step of a workflow information system development is its implementation. This step consists on the implementation of the WfIS. We are primary interested in the UIs that are needed for the system rather than the management functionality of the workflow. We provide means to develop the UIs to:

- *Support user's tasks*. These UIs are the result of a specification of user' tasks using task models. Thus this generates most of the UIs for the WfIS.
- *Support user's communication* with UI for user's agendas and manager's worklist that must be updated accordingly as users received a new item or when they finish a task. This category of UIs even that its design is not complex their functionality it is and communication protocols are out of the scope of this work.
- *Support tasks allocation, resource patterns*, by different means such as: allocation, offering, delegation. We draw attention on these UIs and provide a solution on how to develop them. We called this UIs workflow user interface patterns (WUIPs).

As all the UIs are expected to be the result of the use of UsiXML methodology we consider its literature as guidance. UIs can be developed from a task model definition. So, we provide task models for the resource allocation patterns in order to design their UIs based on UsiXML methodology.

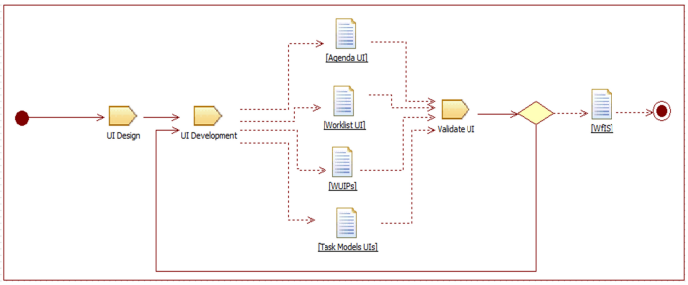

Figure 3. Implementation steps.

The set of activities (Figure 3) starts with the UI designer sketches the UI design that is then developed by a UI developer. Each UI is evaluated by the end-user. When the evaluation satisfies end-user expectations then it becomes part of the WfIS. The same set of activities is carried out for every single UI of the corresponding set. We included UI evaluation as part of the activity set but we do not provided any mean to perform such evaluation. UI evaluation has been largely addressed in the literature thus any method for UI evaluation can be used. Even more, UsiXML literature addresses usability aspects and guidelines evaluation in their methodology.

To develop UIs from task models, we rely on UsiXML method [19] that is compliant with the Cameleon Reference Framework [3]. This method is divided in four steps: task model, abstract user interface model, concrete user interface model and final user interface. UsiXML uses a set transformation rules to pass from one development step to another. Transformations are encoded as graph transformations performed on the involved models expressed in their graph equivalent. In addition, a graph grammar gathers relevant graph transformations for accomplishing the sub-steps involved in each step. A complete overview of UsiXML can be found in [19].

Finally, the workflow manager allows the simulation of the operation of a service of management of workflow. This second part of the program makes it possible to trace the work throughout their advance in the workflow. The user starts by charging the specification of a workflow realized previously thanks to the editor. This stage creates one or more new case

via the menu (new case). These cases are found at the beginning in the place of entry. Their advance will depend on the cartography established with the editor. When a case is in a place, it means that the next task could be associated a resource. In this moment the access to the option of assignment of resources is available. In figure 4 is depicted the color coding incorporated to the workflow editor. Each task to be performed within an organization unit has some resources allocated and available. In figure 4 a potential bottleneck is shown in red for transition 5.

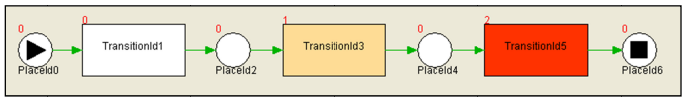

Figure 4. Three possible states of a transition.

III. COGNITIVE DIMENSION FRAMEWORK

The CD framework comes with 14 dimensions which focus on different aspects of a notation, each of which is cognitively relevant, giving a 'profile'. The profile determines the suitability for various tasks [6]. Each dimension can either be positive or negative, depending on the application in which the notation is applied. This context is referred as the type of activity from which we will get the preferred profile. Green [5,6] distinguishes four possible applications:

- *Incrementation*, refers to adding new items.
- *Transcription* refers to translating items from one domain of application to another.
- *Modification*, refer to modifying an item.
- *Exploratory Design*, refer to combining incrementation and modification.

FlowiXML is relevant to the four types of applications. *Incrementation* is an application of the CDF that fits to our needs; the design of the workflow is incremental. For instance, adding a new organizational unit to a workflow model; adding a user stereotype to a job. *Transcription* is also applicable to FlowiXML as it offers the possibility to translate some of its internal representations to external representations, for instance, the requirements elicitation table can be translated into a Microsoft Excel table. Also, to translate external representations into internal representations, for instance, to transcribe a workflow model expressed in the Business Process Modeling Notation into FlowiXML notation workflow editor. In FlowiXML rearranging and changing a process model in a workflow model is a regular task thus the artifact is *modifiable*. This characteristic is intrinsic to the problem we are dealing with as the organization is a dynamic entity thus provoking constant changes in the workflow design. FlowiXML adopts an exploratory design with the characteristic that the desired end state is not known in advance. When designing workflow information systems, the final result of the workflow model is never know in advance, one of the goals of workflow information systems is to enhance and improve business process with the introduction of automation of their process. Then constant refinement is desirable before reaching the, at that moment, the desired final state.

IV. EVALUATING THE COGNITIVE DIMENSION OF FLOWIXML

The evaluation of graphical notations is not an exact process which often results in variegated and/or subjective conclusions [DeBo06]. However, it gives us hints to improve the notation and the quality of the designs it produces. Since the early conception of the tool and during its, still under construction, evolution there have been significant changes always based on users feedback after intense used of the tool. Even that FlowiXML is composed of several notations, mainly, task, process and workflow models. We analyze every notation in each cognitive dimension, if it was something to say about it, each model has just one notation associated. Also, being related to the software tool the reader might confuse the usability of the software with the usage of the methodology. The methodology can be used independently of the software support. As many of the selected cognitive dimensions are related to each other, i.e., changes in one can affect other. On the Many of the cognitive dimensions are pair wise independent: any two can be varied independently, as long as some third dimension is allowed to change freely. The CDF defines 14 cognitive dimensions; we explore them and determined which of them were of interest for our artifact. The list is described in the next paragraphs.

A. Viscosity

Viscosity is a property of the system as a whole. It refers to resistance of the sys-tem to change. This means that it becomes hard after modifying our model to get a desire state. Changes may be related to different operations, such as: adding, removing, modifying, consequently viscosity may be very different depending on the operation and the operator. For instance, viscous is the operation of removing an organizational unit while fluid could be adding a task to a process. It is known that viscosity might be problematic in an exploratory design if not tackled correctly [5,6] for at least one reason: redesigning in a graphical editor usually requires much tedious work, and frequently many similar alterations need to be made to different parts of the picture.

Adding, removing and/or modifying text in the model elicitation tool is considered less viscous for all the advantages that it offers, such as: automatic identification of action verbs and the transcription of textual scenarios into a task spreadsheet.

Adding, removing and/or modifying a task in the task spreadsheet editor is considered less viscous there is no affection on the task list when an operations is done on the task identification, on the contrary the transcription of the changes to the workflow editor is automatically done.

Adding, removing, and/or modifying a job from the job editor is considered viscous because of its impact in the model might require manual adaptation of the process model to determine the role in charge to perform the task previously assigned to other roles. Similarly, adding, removing, and/or modifying a user stereotype is considered viscous because of its impact in the model. Likewise, when operations are on organizational units where a whole restructuration of the workflow might be needed if an organizational unit disappear.

In any case organizational changes have a viscous nature per se and it is good to have a viscous system supporting operations related.

The process modeling is an activity that we limit to the structuration of the workflow in Petri Nets. When operations over the Petri net take place we could imagine different scenarios with viscous or less viscous impact. Adding, removing or modifying a process of the Petri Net might require lines connecting it to other boxes will have to be moved (viscosity problem known as knock-on). Then most each line will have to be redrawn individually (problem known as repetition). Allocating a task has been explored in deep. Task allocation patterns have constraints while combined. This has been explored and the tool checks the validity of any attempt to add, change, and remove an allocation pattern. The problem becomes viscous when such changes need to be propagated, knock-on, and new jobs satisfies the new allocation model. The task model is viscous due to its hierarchical structure. Deleting leaf tasks might not produce a lot of work in reorganizing the tree structure but when the deletion is done on a parent node then all the lines connecting it to other nodes will have to be moved, knock-on. Then most each line will have to be redrawn individually, repetition. As a whole, we consider the viscosity of FlowiXML as acceptable as most of the operations it supports are less viscous and those which are viscous are due to the intrinsic nature of their representation (Petri Nets and Task Models) and the operations (organizational changes).

B. Hidden Dependencies

Hidden dependencies refer to a one-way pointer, where A points to B but B does not contain a back-pointer to A. This problem is of interest because there a lot of dependencies in the workflow that they might be explicit as much as possible.

The text of the requirements has hidden dependencies, as it is possible that deleting text might imply to delete task, organizational units, and so. It is hard to imagine how from a textual description possible dependencies might be highlighted to prevent harmful. Similarly, to the previous aspect, task dependencies cannot be easily foreseen at this level. However, the spreadsheet provides information that is relevant to the user to prevent him from possible dependencies, such as: rationale of the task. Of course the cognitive load to recognize that dependency is significant, particularly for non-experts. Thanks to graphical notation selected in FlowiXML it is clear and explicit dependencies between job, organizational unit and processes. Petri Nets make the dependencies explicit. The task model that details the work to be performed in a task of the process model is hidden in the process model view; this information is not fully available, unless the task model editor is launched. Task allocation patterns have been explored to identify their dependencies which become explicit. Task models use a hierarchical structure with links showing dependencies explicitly. Overall the hidden dependencies are not a problem in FlowiXML there were avoid thanks to the explicitness of the notations (task model and process model). However, intermodel dependencies have not been fully considered. We support forward transcription and keep consistency from the model elicitation until the process model. When a change is made on other models such as the process model the backward traceability is not supported until the requirements elicitation model. A solution to this problem is foreseen by having a rule like: When a new task I added to the process model then create text in the model elicitation tool: The new task is performed in the organizational unit X by the job a, after the precedent tasks, and gives input to the next task. This is not a full textual description but then it could be refined by the user.

C. Premature Commitment

Constraints on the order of doing things force the user to make a decision before the proper information is available. It is known [Gree98] that experts recognize potential problems much earlier, perhaps not from looking ahead but by familiarity with likely sources of difficulty. We notice that this is something that is problematic because our methodology, ideally, follows a series of steps from which decisions made have an impact. This is why we recognize this feature and act to prevent the conflict. From the textual scenario is vital to identify most relevant elements of the structure but novice users lack to understand and highlight the correct experienced this type of problems with the user of the system. Clearly differentiating a task, from process and from workflow is vital at early stages of the development. This is why we provide guidance, with our workflow identification criteria, to non-expert users of the methodology to properly select among the options; this solution is known as decoupling. Planning task allocation patterns is vital before selecting them. We provide guidance with our tables showing the constraints and dependencies between different patterns. Workflow designers have information that could help them to make the right decision about the type of resource allocation pattern to be assigned. Overall FlowiXML and the nature of our methodology force some premature commitment while modeling the workflow. Research has shown that expert designers frequently treat potential trouble-spots differently, putting them aside for later treatment or attacking them early [Gree98]. For novice experts we provide methodological guidance trying to avoid problems derived from wrong decisions.

D. Secondary notation

Secondary notation refers to extra information carried by other means than the official syntax. All concepts from the workflow model can be exported to a XML-format that can be processed to check the reachability of the tasks and completeness of the model. The organization also can be viewed as a hierarchical diagram. Thus, FlowiXML addresses the need of the secondary notation for most of their modules by expressing the graphical notation as a XML-based format. However, in some cases when the second notation is not controlled in FlowiXML there is no way to be consistent. For instance, imagine a mapping between the workflow model and the workflow model used by the BPMN notation. Any change in the BPMN notation makes incomplete the mapping.

E. Visibility

Visibility refers to extra information carried by other means than the official syntax. The exploratory design is encouraged to uses juxtaposition (giving the ability to put two or more

items visible at the same time) as a seed for problem-solving [5,6].

All aspects of the organizational modeling are visible and available to be used. In case that a modification takes place menus offer the option to open the appropriate editor. Menus are always contextualized within the different windows of each tool. For instance if a new user stereotype is added, the workflow designer can compare with an existing workflow model what is the job that the user stereotype needs to be allocated to a task. The process model offers a map view that allows the designer for a quick scan of the model. This miniature map-view reduces the visibility problem in large problems. Task modeling using a hierarchical structure presents a visibility problem when the task model deep (level of the tree) increases. Even that it is not yet implemented, we could imagine a similar map-view as the one used for the process model to visually scan the task model.

Selecting a resource allocation pattern is a complex task that demands perfect understanding and knowledge about the patterns. We have visibility problems for this activity because we do not provide any possibility to the workflow designer to compare a pattern assigned with other task, side-by-side viewing. Although we tried to organize and present them in a consistent way via the WUIPs we did not found clear understanding by the users. Also, it would be nice to provide some guidance to remind the user the meaning of the patterns and their applicability, although we provide this information on paper, which reduce to some extent the visibility problem. So as for the table of constraints for resource allocation patterns, to prevent the user from of the consequences of their decision they can check the table on paper, ideally this should be presented digitally on the screen within the tool.

FlowiXML considers the importance and relevance of visibility. We consider that visibility is acceptable with some aspects still to be improved. For instance, Problems might be related to workflow diagrams or task model diagrams when they grow. So far, from the case studies we have not yet identified this problem but with more complex problems they might arise.

F. Abstractions

Abstraction is a group of elements to be treated as one entity, either to lower the viscosity or to make the notation more like the user's conceptual structure. We identified different abstractions. The *abstraction barrier* is the minimum number of abstractions that must be known before the notation can be used. *Abstraction-hungry* refers to systems can only be used by deploying user-defined abstractions. *Abstraction-tolerant* refers to a system that permits but do not require user-defined abstractions. *Abstraction-hating* refers to systems that do not allow users to define new abstractions (and typically contain few built-in abstractions). Personalization of the notation is desired but has not been addressed in FlowiXML. More meaningful icons should be used to denote jobs (doctor icon, secretary icon), organizational units (factory drawing, hospital drawing, university drawing), or even for the task model, anybody would be to choose the representation of their stylistics. Some predefined or any custom loaded from a file. Also, it might be of interest to provide means to the users to store patterns of their design [14], for instance, the task model of a system login, would save designers some time. However, FlowiXML does not allow the creation of custom notations thus this feature is not related to FlowiXML. Not addressing the *abstraction* dimension would be *harmful* for non experts just when modifications are to be implemented.

Abstractions refer to the number of models and the nature they have in order to help workflow designers to design workflow models. We consider that we provide a sufficient number of abstraction that do not hide any information, thus the user does not need to think about building new concepts on top of our notation to design their workflow models. This means that no new constructions are expected in order to create new models. However, the workflow designer must integrate concepts to build a workflow model, for instance, an organizational unit is composed of jobs, user stereotypes, process models, task models, and this is something that must be done by hand but does not represent new abstraction to our system.

G. Closeness of Mapping

Closeness of mapping refers to the close is your artifact to the representation domain. In FlowiXML, there is no way to know the size of the system you are designing (number of processes, jobs). However, a workflow design using the Petri Net notation could be considered a common ground in the workflow community. So as the task modeling notation is very well known in HCI communities. Relying on a notation that does not change a common understanding of the domain of the problem is important and this is why we always tried to rely on concepts and notations that were known. Relying on a notation close to the domain of application is important. In FlowiXML we did survey different notations, so as methodologies, and we tried to add here our solution to meaningful concepts closed to the domain of application. This is the case of the Petri nets for workflow representation, the notation of the task model, the task resource allocation patterns.

H. Consistency

Consistency is one of the most important aspects of usability. We can discuss consistency at different levels with regard to FlowiXML. First, there consistence in the way all models and concepts are represented, meta-models using UML class diagrams. Second, all models use the same UIDL thus keeping consistency at the language representation. The notation selected for the different models is consistent with existing knowledge on the different topics. Finally, the tool implemented lacks of consistency for the nodes manipulation in the graphical editors for Petri Nets and Task Models. This inconsistency is related to selection and manipulation, release operations.

I. Diffuseness

Diffuseness refers to Expresses the verboseness of a notation. The notation for FlowiXML must not be diffuse. We consider that the different notations used are not diffuse, and are composed of a reduced set of components that make them easy to understand and, what is more important to apply. However, modeling resource allocation patterns however did not show to be clear, even with the WUIPs. The understanding

and knowledge require to apply them requires to keep in mind a long description and examples of their applicability. Unfortunately, we did not find a simpler representation but at least we provide some guidance by creating the drawing shown in Figure 5.

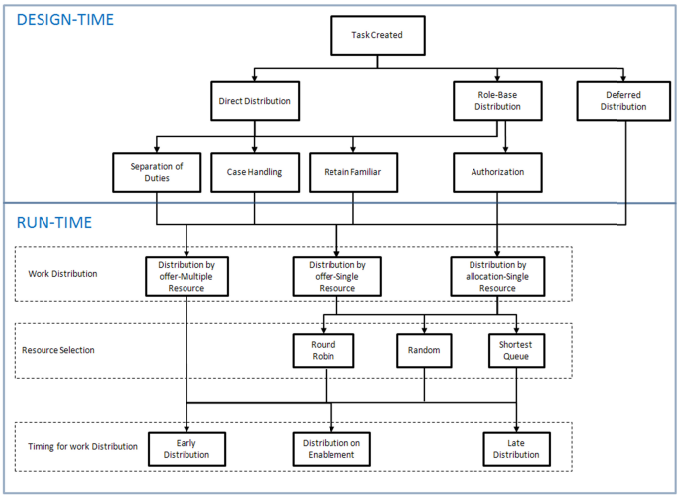

Figure 5. Resource allocation patterns selection.

J. *Error-proneness*

Error-proneness evaluates the notation in terms of the errors it produces. Errors are produced from different sources but when they are caused by the notation itself then there is something to verify in our artifact. The cognitive load due to diffuseness of the notation or inconsistency within the different notations used in FlowiXML might be source of errors. Not being consistent with the interaction the user has with the whole artifact proved to be a source of user's errors and dissatisfaction. Particularly because the way nodes are manipulated in the task model editor is very different from the manipulation of nodes in the workflow model editor. If we separate and evaluate each software tool this problem was not identified, for instance, as Petri Nets come along with three basic elements, connector, transition, and place, where connections are just possible between a state and a place, thus, the error were not possible. Related to the models, task attributes we invite users to use the task life-cycle to understand task' attributes.

K. *Hard mental operations*

High demand on cognitive resources was considered as a result of considering other criteria, such as: viscosity, consistency, diffuseness, reduced but yet extent set of abstractions. Nevertheless, as discussed is other sections, the results are not perfect. Particularly with respect of the task allocation patterns, from users experience using the FlowiXML it was learnt that task allocation patterns were hard to understand and off course to use correctly. However, there were some cases in which users showed that their use was straightforward. Without being an expert in psychology, we adventure to say that cognitive capacities for abstraction made a difference when using FlowiXML.

L. *Progressive evaluation*

The *progressive evaluation* means that the work in progress can be easily checked for as far as it is finished. The evaluation of the workflow modeling can be assessed at any time via the different software tools. In an exploratory design it is important to have the opportunity to validate the work in progress constantly. FlowiXML provides a mechanism to check the reachability of the workflow model, completeness of the model, and provide a checklist, this serves to evaluate the progress of the workflow modeling.

M. *Provisionality*

The *provisionality* refers to the degree of the premature commitment, i.e., how hard is going back from our actions. Considering the premature commitment that we have identify for FlowiXML, which we considered to be important, we consider the degree of commitment to be reasonable. We meant that it is possible to go back in the decisions made but in some cases it might imply more than just deleting and adding a concept.

N. *Role-expressiveness*

Role expressiveness refers to the facility that the notation to split into parts. In FlowiXML the designer can jump to see the work from the organizational point of view and then look at details of the organizational units or the details of the process. Task attributes are not easily reached and might be even not all available for manipulation. Distinguish the different components or a logical block in a notation is a key feature for modeling. We considered that FlowiXML covers this dimension.

V. CONCLUSION

The cognitive dimension framework is a powerful tool to evaluate an artifact. FlowiXML is an artifact that is composed of a notation, a method and software tools to support the development of workflow information systems. The use of the cognitive dimension framework to evaluate FlowiXML showed to be very valuable as it help us identify problems in the guidance and the software support. Thus, they represent the starting point for future improvements of the whole artifact. The future work of FlowiXML certainly includes those findings got from evaluating our artifact against the cognitive dimension framework. Also, the evaluation showed that there were many aspects well covered that reinforced our impression that we are going to the right track.

ACKNOWLEDGMENT

We acknowledge the support of the "Repatriacion" program by CONACYT and the support of the ITEA2 Call 3 UsiXML (User Interface extensible Markup Language – http://www.usixml.org) European project under reference #2008026 and its support by Région Wallonne, DGO6.

REFERENCES

[1] Blackwell, A.F. & Green, T.R.G. *A Cognitive Dimensions questionnaire optimised for users*. In A.F. Blackwell & E. Bilotta (Eds.) Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group (PPIG), Cozenza, Italy, April 2000 pp. 137-152.

[2] Blackwell, A.F., Britton, C., Cox, A. Green, T.R.G., Gurr, C.A., Kadoda, G.F., Kutar, M., Loomes, M., Nehaniv, C.L., Petre, M., Roast, C., Roes, C., Wong, A. and Young, R.M. *Cognitive Dimensions of Notations: Design tools for cognitive technology*. In M. Beynon, C.L. Nehaniv, and K. Dautenhahn (Eds.) Cognitive Technology 2001 (LNAI 2117), Springer-Verlag, pp. 325-341.

[3] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, vol. 15, no. 3, 289–308. 2003.

[4] De Boeck, J., Raymaekers, C. Coninx, K., *Comparing NiMMiT and Data-Driven Notations for Describing Multimodal Interaction*, Tamodia 2006 (Diepenbeek, Belgium), October 2006.

[5] Green, T.: Cognitive dimensions of notations. In: People and Computers, Cambridge University Press, Cambridge, UK (1989) 443–460.

[6] Green, T., Blackwell, A.: Cognitive dimensions of information artefacts: a tutorial. Available on: http://www.ndirect.co.uk/thomas.green/work stuff/papers/ (2005)

[7] Guerrero, J., Vanderdonckt, J., Gonzalez Calleros, J.M., *FlowiXML: a Step towards Designing Workflow Management Systems*, Journal of Web Engineering, Vol. 4, No. 2, 2008, pp. 163-182.

[8] Guerrero García, J., Lemaigre, Ch.,Vanderdonckt, J.,González Calleros, J.M.,*Model-Driven Engineering of Workflow User Interfaces*, Proc. of 7th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2008 (Albacete, 11-13 June 2008), Springer, Berlin, 2008

[9] Guerrero, J., Vanderdonckt, J., González Calleros, J.M.,Winckler, M.,*Towards a Library of Workflow User Interface Patterns*, Proc. of 15th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2008 (Kingston, July 16-18, 2008), Lecture Notes in Computer Sciences, Vol. 5136, Springer, Berlin, 2008, pp. 96-101

[10] Guerrero, J., Vanderdonckt, J., Lemaigre, Ch.,*Identification Criteria in Task Modeling*, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 7-20.

[11] Guerrero, J., Lemaigre, Ch., Gonzalez Calleros, J.M., Vanderdonckt, J., *Towards a Model-Based User Interface Development for Workflow Information Systems*, International Journal of Universal Computer Science, Vol. 14, No. 19, 2008, pp. 3236-3249

[12] Guerrero Garcia, J., Vanderdonckt, J., González Calleros, J.M., *Developing user interfaces for community-oriented workflow information systems*, Chapter 16, in D. Akoumianakis (ed.), "Virtual Communities of Practice and Social Interactive Technologies: Lifecycle and Workflow Analysis", IGI Global Inc., Hershey, 2009, pp. 307-329

[13] Lemaigre, Ch., Guerrero, J., Vanderdonckt, J., *Interface Model Elicitation from Textual Scenarios*, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 53-66.

[14] Montero Simarro, F., Vanderdonckt, J., *Generative pattern-based design of user interfaces.* LSM Working papers 08/13, April 2008. Available online at: http://www.uclouvain.be/cps/ucl/doc/iag/documents/WP_08-13.pdf

[15] Nielsen, J., *Coordinating User Interfaces for Consistency.* (Ed.) Academic Press Professional, Inc., San Diego, CA, 1989.

[16] Paternò, F. (2000) Model-based design of interactive applications. Springer Verlag, Berlin.

[17] Russell N., van der Aalst, W.M.P., ter Hofstede, A.H.M. & Edmond, D. (2005) Workflow Resource Patterns. In the 17th Conference on Advanced Information Systems Egineering (CAISE'05). Porto, Portugal. 13-17 June.

[18] van der Aalst, W.M.P. The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1):21--66, 1998.

[19] Vanderdonckt, J., A MDA-Compliant Environment for Developing User Interfaces of Information Systems, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31.