User Interactions in Dynamic Processes

Modeling User Interactions in Dynamic Collaborative Processes using Active Workspaces

Nsaibirni Robert Fondze Jr* — Gäetan Texier**

* LIRIMA, University of Yaounde 1
PO Box 812, Yaounde, Cameroon
Centre Pasteur of Cameroon
nsairobby@gmail.org

** Centre d'épidémiologie et de santé publique des armées (CESPA)
UMR 912 - SESSTIM - INSERM/IRD/Aix-Marseille Université
gaetan.texier@univ-amu.fr

ABSTRACT. Flexibility and change at both design- and run-time are fast becoming the Rule rather than the Exception in Business Process Models. This is attributed to the continuous advances in domain knowledge, the increase in expert knowledge, and the diverse and heterogeneous nature of contextual variables. In such processes, several users with possibly heterogeneous profiles collaborate to achieve set goals on a processes mostly designed on-the-fly. A model for such processes should thus natively support human interactions. We show in this paper how the Active Workspaces model proposed by Badouel et al. for distributed collaborative systems supports these interactions.

RÉSUMÉ. La flexibilité et la changement pendant la conception et l'éxécution sont de plus en plus centrale dans les modèles des Business Process. Ceci est dû aux avancées continues des connaissances dans divers domaines, à l'augmentation des connaissances des experts, et de la nature hétérogènes et multiple des variables contextuelles. Dans ces processus, plusieurs utilisateurs ayant des profiles hétérogènes collaborent à des fins communs sur un processus défini progressivement. Un modèle pour de tels processus doit donc supporter nativement les interactions utilisateur. Nous montrons dans ce papier comment le modèle des Active Workspaces proposé par Badouel et al. pour la modélisation des tels processus support les interactions utilisateurs.

KEYWORDS : Collaborative Business Process, Human Interactions Patterns, Active Workspaces

MOTS-CLÉS : Processus Collaboratif, Interactions Utilisateurs, Active Workspaces

1. Introduction

Flexibility and change are fast becoming the Rule rather than the Exception in Business Process Models. As domain knowledge advances and expert knowledge increases, data and process definitions are prone to change. The need for dynamic process models is continuously being felt. Moreover, it is safe to say that dynamic process models increase user satisfaction and motivation at work, and positively influence productivity.

In [16] processes are classified as tightly-framed, loosely-framed, adhoc-framed, or unframed, depending on their predictable and repetitive nature, and on the degree of dynamism they require. The move from tightly-framed process models to unframed process models is characterized by the increasing facilities to manage uncertainty and exceptions, and the increasing influence of users and expert-knowledge in process design and enactment.

We focus on adhoc-framed and/or unframed domains, where users carry out processes in a fair degree of uncertainty[2][16] because processes cannot be completely modelled at design time either due to their large numbers or because they are highly data-centric and will have to be discovered as data is produced and as the environment evolves. In these domains, users (knowledge workers) are central to the different processes. They perform various interconnected knowledge intensive tasks and have to make complex rapid decisions on process models defined on-the-fly[2].

An example of such a domain is the disease surveillance process in public health. The process usually goes through a continuous cycle of collecting, analyzing, and dissipating information about a health condition of interest with the aim of detecting and handling unwanted events in the general population[8]. Disease surveillance is characterized as being multi-user, multi-organizational, knowledge-intensive, and time-bound[8][5]. Users and/or organizations need to collaborate and make complex rapid (timely) decisions on a semi-structured process model[2].

Like most organizational structures, a majority of national disease surveillance systems place users in a hierarchical pyramid[8]. In each level of the pyramid, users are grouped into Roles to carry out related work. Communication between the different levels of the pyramid and between the different Roles is usually through the asynchronous exchange of messages.

Our objective in this paper is to illustrate how user interactions (collaboration) in dynamic processes is supported by the Active Workspaces model[1]. We start by presenting key forms of human interactions found in business processes, then we present a purely distributed and informal specification of the Active-Workspaces model and show how it supports these interactions.

2. User Interaction Patterns

By user interaction, we mean any form of communication between a user and a computer or between two or more users via a computer[14]. Users interact in protean ways to have work done on a variety of task categories. Tasks are seen as work to be done and either originate from service calls or from work-(re)distribution in a team (work transfer and work delegation). In the following paragraphs, we describe the different ways users can interact. Our descriptions are inspired from the IBM's Business Spaces [14] that

define a human workflow attached to the underlying process model and on observations from concrete disease surveillance scenarios at the Centre Pasteur of Cameroon.

2.1. User interactions

In dynamic processes in general, users collaborate in the context of resolving specific cases. A case is a concrete instance of a business process[1]. For example, a case can comprise all tasks that will be invoked due to the arrival of a patient at a hospital or due to some outbreak alarm produced by some automated disease surveillance algorithm. One of the participating users initiates the case by instantiating the main task and providing the initially needed information. He then proceeds with the initial assignments and orchestration of tasks (work) to the other participants.

A simple description of a user's working environments could be: each of the participating users possess a work-basket which contains pending pieces of work that have been assigned to the user. In like manner, team-baskets are used to share work among a group of individuals. Task definitions contain information about the roles that have the ability to carry them out.

2.1.1. Work assignment or service request

Though users collaborate on processes in a peer-to-peer fashion, there is always a coordinating user who besides doing work is charged with initiating processes, assigning work to users, and coordinating the orchestration of the entire process. Such users exist throughout the entire process hierarchy, each managing the coordination of work that originates from him/her. Assigning work to some user (respectively to a group of users) consists in placing the work description in the user's work-basket (respectively in a group's work-basket).

2.1.2. Claiming/Releasing work

Users claim and carry out work placed in their work-baskets either based on the work-priorities or on the availability of the required input. A user can on the other hand release work placed on his/her work-basket when for some reason he is unable to carry it out.

2.1.3. Completing work

When a user claims work from his work-basket, he can either use an existing process definition to carry out the work or define a new process to do so. In both cases, he explicitly choses the method to use and provides the required input data. For certain routine tasks, he uses a rule-based approach to define a default method to always apply.

2.1.4. Handling situations

One of the following situations may arise: a user might want to rollback and change the method he applied to resolve a task, or a user might become overbooked or unavailable or unable to complete work due to the unavailability of some input data. Such situations are handled in one or more of the following ways: undo, redo, release work, transfer work, delegate work, re-prioritize work. These strategies are applied to take into account new constraints and/or facilitate and quicken decision making.

3. User Interactions in Active-Workspaces

Explicitly described in [1], the Active-Workspaces (AW) model uses attribute grammars to represent tasks and their decomposition into sub-tasks. Inherited attributes are

used to pass data from the parent to the sub-tasks while synthesized attributes are used to return results from subtasks to parent tasks. Attributes are terms over an ordered alphabet and task triggering and execution is guarded by conditions on the inherited attributes (using First Order Logic formulas and Pattern Matching). Hence the name Guarded Attribute Grammars (GAGs) given to the underlying grammar on which Active-Workspaces are built. In this section, we will show how the Active-Workspace model supports the major aspects of user interactions presented in the previous section.

3.1. Active-Workspace: User-roles, Users, and Services

The main building block in the Active Workspaces model is the user (identified by his Active-Workspace) and collaboration between users is materialized by the exchange of services. Each user can play several roles. Services are attached to Roles and users only offer services that are attached to the Roles they play. An Active-Workspace contains:

– Guarded Attribute Grammars: A (minimal) GAG is defined for each new service in the system and copied into the workspaces of the users that offer the service (that is, users that play the role to which the service is attached). The axiom of the GAG specifies the name of the service and the productions (Business Rules) describe how this service is decomposed into subtasks. A service definition contains a unique sort s (the axiom), input variables t_i (eventually with guards), and output variables y_i.

$$s(t_1, \ldots, t_n)\langle y_1, \ldots, y_m\rangle$$

– Artifacts: These are process execution trees corresponding to concrete cases (work carried out by a user in his workspace). They hold data and computations pertaining to cases from their inception to their completion. The tree contains two types of nodes: Closed nodes corresponding to resolved tasks or tasks for which a resolution method has been assigned, and Open nodes corresponding tasks that await to be assigned a resolution method. Visually, an artifact is a tree with sorted nodes $X :: s$, where s is the sort of node X.

– Input Buffer: A mail box in which any service requests made to a user as well as local variables whose values are produced in distant locations are placed. In practical situations, it is divided into two; a personal inbox (work-basket) and a role-inbox (team-basket). The former contains task requests made to the user directly and the latter contains tasks made to a role the user offers which he can pick-up and execute.

– Output Buffer: Contains information produced locally and used elsewhere in the system. This includes information about distant calls to services offered by the active-workspace and distant synthesized attributes whose values will have to be produced locally in the active-workspace.

A *task* is therefore simply a guarded attribute grammar production (Business Rule). It is identified by its name (*sort*), its inherited attributes eventually with guards, its synthesized attributes, and a decomposition into subtasks showing how synthesized attributes are produced from inherited attributes. BR1 below is an example of a Business Rule.

$$\begin{array}{l}\text{BR1} :: \mathbf{caseAnalysis}(patient, symps, antecedents, checkRes, labResult) = \\ \quad \mathbf{do}\,(todo, alarm, alert) \leftarrow \text{manageAlarm}(patient, symps, antecedents, \\ \qquad\qquad\qquad\qquad labResult, checkRes) \\ \qquad () \leftarrow \text{manageAlert}(alert, patient, symps, checkResult) \\ \qquad \mathbf{return}(todo, alarm)\end{array}$$

The above task **caseAnalysis**, extracted from the disease surveillance scenario for the monitoring of cases of Ebola[7] depicts what an Epidemiologist does when he receives a suspect case declaration (an Ebola outbreak alarm). This task receives as input information about the patient, the different checks carried out on him, and his laboratory results. It is decomposed into two subtasks manageAlarm and manageAlert, and returns two synthsized attributes *todo* and *alarm*. In like manner, we give an example of an Active-Workspace system description.

$$
\begin{array}{l}
\text{diseaseSurveillance} :: \langle \\
\qquad \text{consultPatient}[\mathit{clinician}], \\
\qquad \text{laboratoryAnalysis}[\mathit{biologist}], \\
\qquad \text{caseAnalysis}[\mathit{epidemiologist}] \\
\rangle \\
\textbf{where} \\
\mathit{clinician} = \text{Alice} \mid \text{Bob} \\
\mathit{epidemiologist} = \text{Ann} \mid \text{Paul} \\
\mathit{biologist} = \text{Frank} \mid \text{Mary} \mid \text{Alice} \\
\text{diseaseSurveillance} :: \quad \% \text{ Modelled system} \\
\text{consultPatient} :: \quad \% \text{ Service offered by clinicians} \\
\text{laboratoryAnalysis} :: \quad \% \text{ Service offered by biologists} \\
\text{caseAnalysis} :: \quad \% \text{Service offered by epidemiologists}
\end{array}
$$

Three services (consultPatient, laboratoryAnalysis, and caseAnalysis) are modeled in this system each offered by a distinct role (*clinician*, *biologist*, and *epidemiologist* respectively). A total of six (6) active workspaces will be generated corresponding to each of the users in the different roles. Parametric Business Rules are used in specifying Business Rules that are service calls. These simply tag the rules with the attached roles.

3.2. Requesting a service and Resolving a case

3.2.1. Requesting a service

As mentioned earlier, whatever the organizational structure, users communicate essentially by rendering and requesting services. Communication is enhanced in the Active Workspaces model using *variable subscriptions*. *Subscriptions* are equations of the form $x = u$ used to model variables x whose values u are produced at a distant site. Thus when a user calls a distant service, the synthesized attributes in the service call become subscriptions to values that will be returned by the call. Each variable has a unique defined occurrence in some workspace and may have several used occurrences elsewhere. This is enhanced using name generators that produce unique identifiers for newly created variables in each workspace.

More formally, let us consider two users: a local user identified by his active workspace AW_1 and a distant user identified by his active workspaces AW_2. When a service call is made from AW_1 to AW_2, the following takes place:

– $X = s(t_1, \ldots, t_n)\langle y_1, \ldots, y_m\rangle$ is added to the output buffer of AW_1 indicating the distant service call. This is distinguished from local calls in that there exist no defining rule for task s in AW_1.

– $Y = s(t_1, \ldots, t_n)\langle y_1, \ldots, y_m\rangle$ is added to the input buffer of AW_2, indicating that a distant service call has been made at node Y. This automatically creates a local node X and adds $Y = X$ to input buffer of AW_2 indicating where this service call is rooted in the the distant workspace.

– $x_i = u_i$ are added to the input buffer of AW_1, indicating that variables x_i in synthesized attributes y_i subscribe to the values of distant variables u_i. In like manner, $u_i = x_i$ are added to the output buffer of AW_2 indicated variable subscriptions it will have to fulfill. These subscriptions are fulfilled incrementally, that is, values are individually returned and sent to distant subscriptions as they are produced.

3.2.2. Task orchestration bus

Resolving a Case starts from an initialisation which consists in instantiating the root node of the main service with the axiom of the GAG. This creates an artifact with a single open node. The subsequent steps (micro steps) captured in the Active Workspaces model are sanctioned either by the application of business rules to open nodes or the consumption of a fulfilled subscription from its input buffer. Either way, executing a micro step adds data to the existing system and the only ordering on these steps is imposed by their data dependencies.

A business rule R is applicable at an open node X if its left hand side matches X and if any eventual logical expression on the variables in the inherited attributes evaluates to $TRUE$. This operation of pattern matching produces a substitution σ which is a redefinition of the variables in input positions in terms of variables in output positions of both the node X and the rule R. Several rules may match the open node and the choice of which to apply is made by the user. Once a rule is chosen, node X becomes closed and new open nodes $X_1, \ldots, X_n$ are created corresponding to subtasks on the right hand side of R. At the base, these open nodes are concurrently handled with an implicit ordering imposed by variable dependences. However, it is possible to add priorities, start- and due-time to tasks and hence to nodes and recommend a certain order in the execution of these tasks. These additions can be updated at any given moment to take into account new contextual realities. Open nodes for which no applicable rule is found correspond to services that have to be requested from a distant users.

Messages received at the input buffer also update the local configuration of the Active Workspace. These messages correspond either to the reception of a service call or to the fulfillment of a subscription. The former instantiates a root node for the corresponding service in the user's workspace while the latter recursively applies the effect of the subscription up the artifact tree.

3.2.2.1. Case Transfer, Delegation, and Synchronization

Case Delegation is naturally supported through service calls and is modeled in GAGs as terminal symbols and grammar axioms. A service is offered by a role and hence by users who play the role. A user cannot call a service he offers. In other words, users cannot call services attached to roles they play. Also, each service is designed to serve a particular role. That is, only users who play that particular role can call the service. Summarily, exchange of services only occur between roles and not within roles. However, users in the same role can communicate in two ways: *Case Transfer* and *Artifact Synchronization*.

In practice, *Case Transfer* is employed as a strategy to handle situations related to user unavailability and/or inability to complete work. To transfer a case, it suffices to transfer the initial service call to the new active-workspace and update the subscriptions accordingly. This creates a new artifact on which the distant user can start working.

Case Synchronization consists in weaving artifacts of the same service enacted in different workspaces. Practically, it can be used to share information between users working on the same case (for example after a case transfer). It can be either unidirectional (a

user shares his artifact with another user) or bidirectional two users synchronize artifacts in their workspaces. This feature considers artifacts as aspects and applies an operation reminiscent to the composition of aspects in aspect oriented programming.

3.2.2.2. Evolving the Active-Workspace

If we abstract the Active-Workspace model a level or two up, it becomes evident that this model has two major separate components: a dynamic underlying guarded attribute grammar specification, and an execution engine. New business rules, services, roles, and users added to the underlying grammar are automatically taken into consideration in subsequent executions of the system. This means that users can at any moment add, remove, or change the underlying grammar and these changes are directly visible (with no retrospective effect).

These two components form a single whole to provide users with the needed flexibility in designing, executing, and managing tasks in their active workspaces which by nature are perpetually evolving.

4. Discussion and Conclusion

Dynamic processes have been at the center of BPM research recently as per these reviews: [16] and [2]. Most of these research works have focused on flexible process design with users considered as part of the external environment[3][17][4][13][10]. A few other works show how exceptions and to some extent, uncertainty are managed in dynamic processes [12][9]. These works use a set of predefined exception handlers and again do not place users at a central position. The few researchers that have carried out work on user interactions have had to define an overlying user-workflow on a predefined process workflow[14][17]. These effectively enhance user interactions by adding flexibility to process enactment but lack flexibility in process design as the process has to be defined prior to its execution.

Active workspaces provide a holistic approach to dynamic process management with users, data, and processes being the essential building blocks. This model possesses to varying degrees the different forms of process flexibility presented in [16]. This explains why it naturally supports most forms of human collaboration in dynamic processes. We have used this model to show how such interactions can be supported. It is important to note that these operations might entail coupling the Active-Workspace model with external databases, knowledge bases, time servers, process performance monitors, etc. These certainly increase an overhead on the Active-Workspace model but have no negative effect on the specifications.

5. References

[1] Eric Badouel, Loic Helouet, Georges-edouard Kouamou, Christophe Morvan, and Robert Fondze Jr Nsaibirni. Active Workspaces : Distributed Collaborative Systems based on Guarded Attribute Grammars. *ACM SIGAPP Applied Computing Review*, 2015.

[2] Claudio Di Ciccio, Andrea Marrella, and Alessandro Russo. Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches. *Journal on Data Semantics*, pages 29–57, 2014.

[3] R. Hull, E. Damaggio, F. Fournier, M. Gupta, Fenno Terry Heath, S. Hobson, M. H Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín. Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles. In *Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA*, volume 6551 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2011.

[4] Kunzle V, Reichert M PHILharmonicFlows: towards a framework for object-aware process management *Journal of Software Maintenance and Evolution: Research and Practice*,2011

[5] M.M. Wagner, L.S. Gresham, and V. Dato. Chapter 3 - case detection, outbreak detection, and outbreak characterization. In M.M. Wagner, A.W. Moore, and R.M. Aryel, editors, *Handbook of Biosurveillance*, pages 27 – 50. Academic Press, Burlington, 2006.

[6] International Society for Disease Surveillance. Final Recommendation: Core Processes and EHR Requirements for Public Health Syndromic Surveillance. Technical report, ISDS, 2011.

[7] R. Nsaibirni, G. Texier and GE. Kouamou. Modelling Disease Surveillance using Active Workspaces. *Conference de Recherche en Informatique (CRI), Yaounde*, 2015.

[8] Centers For Disease Control World Health Organization. Technical Guidelines for Intergrated Disease Surveillance and Response in the African Region. *Technical report, WHO/CDC, Georgia, USA* 2001.

[9] Andrea Marrella, Massimo Mecella, Sebastian Sardina SmartPM: An Adaptive Process Management System through Situation Calculus, IndiGolog, and Classical Planning *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, {KR}* 2014, Vienna, Austria, July 20-24, 2014

[10] Roger Atsa Etoundi, Marcel Fouda Ndjodo, and Ghislain Abessolo Aloo. A Formal Framework for Business Process Modeling. *International Journal of Computer Applications*, 13(6):27–32, 2011.

[11] Claudio Di Ciccio, Andrea Marrella, and Alessandro Russo. Knowledge-intensive Processes: An overview of contemporary approaches. *CEUR Workshop Proceedings*, 861:33–47, 2012.

[12] Reichert M, Rinderle S, Kreher U, Dadam P Adaptive Process Management with ADEPT2 *ICDE*, 2005

[13] ter Hofstede AHM, van der Aalst WMP, Adams M, Russell N Modern Business Process Automation: YAWL and its Support Environment. *Springer*, 2009

[14] Friess Michael Business spaces for human-centric BPM , Part 1: Introduction and concepts. *IBM DeveloperWorks* 2011.

[15] Roman Vaculín, Richard Hull, Terry Heath, Craig Cochran, Anil Nigam, and Piyawadee Sukaviriya. Declarative business artifact centric modeling of decision and knowledge intensive business processes. In *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, number Edoc, pages 151–160, 2011.

[16] Wil M. P. van der Aalst. Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, 2013:1–37, 2013.

[17] W. M. P. van der Aalst, M. Pesic, H. Schonenberg Declarative workflows: Balancing between flexibility and support *Computer Science - Research and Development*, 2009:99–113, 2009