Active-Workspaces: A Dynamic Collaborative Business Process Model for Disease Surveillance Systems

Robert Nsaibirni, Eric Badouel, Gaëtan Texier, Georges-Edouard Kouamou

Active-Workspaces: A Dynamic Collaborative Business Process Model for Disease Surveillance Systems

Nsaibirni Robert Fondze Jr[1,5], Eric Badouel[2], Gaëtan Texier[3,5], and Georges-Edouard Kouamou[4]
[1]LIRIMA, University of Yaounde 1, PO Box 812, Yaounde, Cameroon
[2]Inria and LIRIMA, Campus de Beaulieu, 35042 Rennes, France
[3]Centre d'épidémiologie et de santé publique des armées (CESPA),
UMR 912 - SESSTIM - INSERM/IRD/Aix-Marseille Université
[4]LIRIMA, ENSP, PO Box 8390, Yaounde, Cameroon
[5]Centre Pasteur du Cameroun, Yaoundé, Cameroun.

Abstract— *Flexibility and change at both design- and run-time are fast becoming the Rule rather than the Exception in disease surveillance processes. This is attributed to the diversity in public health threats, to continuous advances in domain knowledge, the increase in expert knowledge, and the diverse and heterogeneous nature of contextual variables. Disease surveillance is one such processes and it is characterized by collaborative work and decision making between users with heterogeneous profiles on processes designed on-the-fly. A model for disease surveillance processes should thus natively support flexible workflow design and enactment as well as human interactions. We show in this paper how the Active Workspaces model proposed by Badouel et al. for distributed collaborative systems provides this support.*

Keywords: Disease Surveillance, Business Process Modeling, Collaborative Systems, Active-Workspaces

1. Introduction

For over twenty years, public health information systems have prospered in all medical areas and activities, in line with the advances in health informatics and related technologies. These systems are identified by the American Medical Informatics Association (AMIA) as belonging to Public Health Informatics (PHI), a specific subdomain of Health Informatics, defined as "the systematic application of information and computer science and technology to public health practice, research, and learning [23]. The scope of PHI was described as "the conceptualization, design, development, deployment, refinement, maintenance, and evaluation of communication, surveillance, information, and learning systems relevant to public health." A recent article in the AMIA yearbook of medical informatics [27] introduces a review of English-language PHI publications in Medline (2012-2014), in which authors propose main essential services such as monitoring health, supporting diagnosis, investigating outbreaks, and evaluating systems. The systems providing these services could be considered as decision support systems since it uses data, documents, knowledge and/or models to identify and solve problems and make decisions.

Concerning syndromic surveillance, defined as the continuous monitoring of public health-related information sources for early detection of adverse disease events, numerous early warning systems are currently used by experts belonging to international, national or local public health institutions. This decreases the response delays, improves effectiveness, and reduces the health impact of the outbreak. According to Chaudet et al. [26][17], outbreak identification and confirmation are managed by epidemiologists during "situation diagnosis," which consists in validating (or revoking) an alarm (signal identified as aberrant or abnormal) and transforming it into an alert (real characterized outbreak), then proposing initial countermeasures.

In health domains, known for their complexity and uncertainty, carrying out situation diagnosis implies complex decision-making processes and involves a wide range of interrelated human, biological and/or environmental activities. A disease surveillance network is thus a socio-technical system which associates geographically distant medical stakeholders (up to a few thousand people in different specialties) with dedicated systems and technical tools (telephone, satellite, digital documentation, ...) collaborating to detect and manage outbreaks [28]. More so, disease surveillance is a semi-structured [16] process which entails that only high level tasks can be clearly defined prior to process execution since most of the activities are discovered at runtime as data becomes available. This increases the complexity as users have to design and run the process-workflow on-the-fly.

Such a system in which users collaborate and share information intensively over a process model defined on-the-fly is termed a dynamic knowledge intensive system [2][11]. The modelling objective in such systems is not to completely automate the processes and their orchestrations but to provide users with expressive tools to permit them flexibly and efficiently create and run processes while making optimal use of the resources at their disposal. These tools can be grouped into four main categories:

1) **Tools for Real-time Iterative Workflow Construction and Orchestration:** As mentioned above, situation diagnoses for instance which is a major phase

in the syndromic surveillance process is an expert activity[17][26]. This means that the decisions and actions to be taken are determined by the expert usually based on incomplete non-pathogenic data. Thus the activity though standardised but remains highly unpredictable.

2) **Tools for User-Interactions:** Disease surveillance is a distributed collaborative activity (spatial and temporal) involving several stakeholders with diverse profiles[29][28]. These stakeholders interact (asynchronously) in myriad ways to find solutions to questions raised during disease surveillance[28].
3) **Tools for managing Exceptions and Uncertainty:** Disease surveillance data is usually described as being incomplete, non-pathogenic, and biased [17]. These are sources of uncertainty and inconclusive decision making. This uncertainty is even accentuated when attempts are made to predict future disease incidences. [30] presents uncertainty as one of the cross-cutting issues that all disease surveillance systems need to address.
4) **Tools to support Decision Making:** The main objective of monitoring diseases is to facilitate decision making and take timely action against public health threats[8][31][32]. In [31], PHI decision support is defined as the process of bringing relevant knowledge to bear to aid decisions involving the health and wellbeing of a population through the use of electronic information. Providing decision support is thus mandatory in all PHI information systems.

In this paper, we present an informal description of the Active-Workspaces model [1], a distributed, user-centric, and data-driven business process model built on guarded attribute grammars. Though the Active Workspaces model can be easily extended to address all of the four tools above, we limit this paper to showing how it provides support for Tools 1 and 2.

The rest of the paper is organized thus: section 2, presents related works in disease surveillance process modeling and business process modeling tools; section 3 presents an illustrative scenario; Sections 4 and 5 respectively elucidate the Active-Workspaces model with its user-centered collaborative constructs, and how the workspace can evolve. Conclusions and future works are stated in section 6.

2. Related Work

Research in public health informatics and disease surveillance in particular has focused on identifying trends/patterns in diseases, potentially viable data variables and sources, and developing novel methods of collecting, aggregating, analysing, and interpreting surveillance information. Little has been done to capture the activities, data, decision, and collaboration schemes that are involved in disease surveillance. In [6], [5], [19] and [8] high-level steps are presented with sample activities that can be carried out at each of them. They go further to characterise the environments (pre-conditions) that favour the application of each of these activities. These pre-conditions only become satisfied at run-time thus supporting our argument for iterative process design and execution.

Futhermore, business process modelling use cases have evolved so far from models that stress on the control and coordination of tasks using state-based formalisms like automata and petri-nets [18][21][14][20][13][10], through data-centric approaches [22][25][4] that use data to dictate the orchestration of activities in a business process, to artifact centric workflows [7][3][24][15] that combine data and activities in one whole (artifacts) and use state-based [24] or declarative [3] [7][15] constructs to guide the evolution of these artifacts in a business process. These techniques however are adapted for structured-domains since they lack the required flexibility needed in disease surveillance processes and place users in the external environment.

3. Illustrative Scenario

We describe below scenarios in syndromic surveillance to better motivate the work presented in this paper.

Several users participate in this scenario: clinicians, biologists, epidemiologists, and pharmacists. We suppose that an Influenza outbreak alarm has just been raised and an epidemiologist assigned to investigate the alarm. We recall that the investigation process aims at confirming the alarm into an alert or revoking it.

The epidemiologist knows of the existence of the different actors listed above but cannot say a priori when or how he will need them during the investigation. Suppose for example that the indicator variable that produced the alarm was $pharmarcy\_sales$. He will start by contacting pharmacists in the epidemic zone to ensure that the sales hike is genuine, that is, it is not caused by some commercial campaign or a similar activity. The alarm is immediately revoked if the latter is true. Otherwise, he has to investigate more. Given the high sensitivity associated with using $pharmacy\_sales$ as an indicator, he decides to pursue tasks that use data tightly correlated with the outbreak. In this case clinical and laboratory diagnostic data. He contacts clinicians in health districts around the epidemic zone for consultation data and runs additional analysis. He requests that patients with Influenza symptoms be contacted and samples obtained if possible and that this be carried out systematically for all new patients presenting symptoms of Influenza. He can even go ahead to request that each sample be multiplied and sent to different biologists for laboratory analysis. This especially if he possesses the required resources or if several tests need to be carried out and he wants to maximize time by spreading the tests across several laboratories.

In parallel to the activities above, he also has to manage a number of support activities such as organizing the transportation of samples from health centers to laboratories, ensuring that the laboratories possess the required reagents and equipment to run the requested tests, etc. He also has to report regularly to public health officials to help them prepare the resources to contain the potential outbreak. He continuous to initiate and run activities collaboratively with other actors until he reaches a conclusion.

If on the other hand the indicator variable was different, say $school\_absenteeism$ or $triade\_calls$ or $consultation\_data$, a completely different set of activities will probably be executed. Futhermore, if this task was assigned to a different epidemiologist, it is not certain that he will run the activities in the same order, or even use the same set of activities. This is because the latter and their ordering highly depend on the experience and expertise of the user and on how much he knows of his environment. Hence the Knowledge-Intensive character of surveillance systems.

This scenario shows how complex resolving a simple task might become when new data becomes available and how unpredictable the surveillance process can be. A model for such a process should therefore provide flexible constructs for building and executing process workflows on-the-fly. The fact that the process model changes is the rule and not the exception.

We also note different forms of interactions between the users and their working environments and equipment (phones, computers, etc.), and among users. For example, the epidemiologist has to interact with his work environment to accept and complete the alarm investigation request and at some point he needs to communicate with other users by sending new requests. Suppose that for some reason in the middle of the investigation, the acting epidemiologist becomes unavailable, the activities he has carried out as well as the information he has gathered will have to be transfered to the new epidemiologist. This is another form of interaction between users: synchronizing expert data.

4. Active-Workspace : User-centered Flexible and Collaborative constructs

In this section we present a succinct informal definition of the Active-Workspaces model. We lay emphasis on the properties that are required to address the two preoccupations treated in this paper. A more formal and complete description of the model is found in [1].

4.1 Active-Workspace

The Active-Workspaces (AW) model is an asynchronous cooperation model in which each participating user is assigned a workspace. A user's workspace is an arborescent (mindmap-like) structure that holds all tasks in which the user is involved as well as the data required to resolve these tasks. The arborescent structure is reminiscent of the hierarchical organisation of tasks in which large complex tasks are broken down to small less complex ones. Each node of the mindmap has a *sort* s indicating the name of the task assigned to it. Task s can be further decomposed into subtasks $s_1, \ldots, s_n$ by applying *production* $P : s \rightarrow s_1, \ldots, s_n$. A node is said to be *closed* when one such production P has been applied to it, otherwise, it is an *open node*. In the former case, the node has successors corresponding to subtasks in the right hand side of P. If the right hand side of P is empty, then node s is a *leave* of the tree. Open nodes, also called *buds*, have no successor nodes. A bud represents a *pending task* that requires the attention of the user: the bud grows when the user decides to apply a production to it. When this happens, the bud becomes a closed node associated with the production and it has n successor nodes that are newly created buds given by the subtasks $s_1, \ldots, s_n$ in the right-hand side of the production.

The hierarchical decomposition of tasks is thus not predefined but depends on decisions made by the user at each step. In disease surveillance, this is particularly useful especially during situation diagnosis. For example, faced with an Influenza outbreak alarm, an expert has to decide whether to use an approach that integrates clinical information, laboratory diagnostic information, spatial data, more profound data analysis, etc. or to just stick with an approach that combines a few of these activities. These approaches can be captured in different productions with the same sort from which the expert can choose when necessary.

Also, Active Workspaces have two main structurally independent layers: an underlying guarded attribute grammar (GAG) model and a GAG execution engine. Any changes made to the underlying grammar are directly visible to the execution engine. This means that new production rules can be added to the grammar at any time and they are immediately available for subsequent task resolution. In the example above, if the expert wants to use an approach for which no defining production exists, he can instantly create one and use it.

4.2 Collaboration and User Interactions

Each workspace is associated with at least one service rendered by the user. A service is represented by a unique sort called the *axiom* of the grammar. The particularity of this sort is that it does not appear in the right hand side of any production of the grammar. Nodes whose sorts are axioms (service nodes) are directly attached to the root node of the workspace tree. The resulting sub-tree rooted at such a node is called an *artifact*. A service call therefore instantiates a new artifact, reduced to a single bud at the root of the workspace. This artifact then develops by the application of productions until it contains no open nodes, that is, the service has been completely rendered. In a multi-user context, we model collaboration between the different

workspaces. Each workspace is associated with at least one grammar identified by its axiom and a set of productions. The sorts of a grammar are either local to the grammar (that is, they appear at the left hand side of at least one production of the grammar), or external (that is, they make reference to axioms of other grammars). Applying a production is just like in a single user scenario with the difference that a sort at the right hand side of a production which references a different grammar will be interpreted as a call to an external service. Resolving this kind of open node provokes the creation of a new artifact in the workspace of the user to whom the grammar is attached. The behaviour of the workspace remains the same as in the single user scenario but for the fact that parts of an artifact will be developed at distant sites when service calls are made.

For example, in the syndromic surveillance scenario above, the epidemiologist requests the expertise of clinicians and pharmacists to investigate the alarm. The clinician in turn requests the services of biologists to run a series of tests on extracted samples. All these interactions between the users are materialised through service calls in the Active-Workspaces model.

4.2.1 Roles

Usually several users play the same role in a system. For example in disease surveillance, there exist several clinicians, several biologists, several epidemiologists, etc. This means that these users (in the same role) are attached to the same grammars after a local renaming of the local sorts. Technically, a role is defined by a generic grammar G and we obtain the disjoint union of these grammars as follows $\oplus(r :: R)G = \biguplus_{r::R} G[r]$ where r is a user who plays role R and $G[r]$ is the grammar obtained from G by replacing each sort (including the axiom s_0) by $s[r]$. Hence $s_0[r]$ represents service s_0 offered by r.

We note $G'\{G[r]\ where\ r :: R\}$ the grammar made up of $\oplus(r :: R)G$ and of a grammar G' that calls this role. That is, G' will at some point need to request a service from a user in this role. In G', we will find productions with parameters such as $P[r] : s \to s_0[r]$ expressing that when the user chooses production P to apply at an open node, he inputs a user r playing role R. The effective production is thus an instance of this generic production. We can also find in G productions of the form $P : s \to s_0[R]$ expressing that a service call is made to all users of the role R. In this case, the production has no parameters since the request will not be made to a particular user.

When a grammar needs to call several roles, we note $G\{G_1[r_1]\ where\ r_1 :: R_1; G_2[r_2]\ where\ r_2 :: R_2; \ldots\}$ and this construction can be applied hierarchically to model chained calls as follows: $G_1\{G_2[r_2]\ where\ r_2 :: R_2\ and\ G_2 = G\{G_3[r_3]\ where\ r_3 :: R_3\ and\ G_3 = G\{\ldots\}\}\}$. This constitution of roles is dynamic as new users can subscribe and/or un-subscribe from one or more roles at any moment. Adding a new user to a role poses no particular difficulty since it does not modify existing workspace specifications but only modifies productions which will be called subsequently. However, removing a user from a role might become problematic if there exist in his workspace artifacts with buds. We can in such a situation either forbid the user from unsubscribing from the corresponding role, or transfer the pending artifacts to the workspace of some user of the same role. Also, as we will see later on in this paper, it is possible for a user to define new productions and extend his local grammar. This means that two users with the same role and thus with identical grammars initially might later possess different grammars. In this case, a synchronization of the two grammars is necessary before the transfer operation.

4.3 Attributes and Guards

Productions are used to structure a user's workspace. They are however not sufficient to model the interactions and data exchanges between the various tasks associated with open nodes (buds). For that purpose, we attach additional information to open nodes using *attributes*. Each sort $s \in S$ comes equipped with a set of *inherited* attributes and a set of *synthesized* attributes. Values of attributes are given by terms over a ranked alphabet. Calling a *task* is written as $(y_1, \ldots, y_m) \leftarrow s(t_1, \ldots, t_n)$ where the $t_i's$ are terms denoting the values of the inherited attributes of task and $y_1, \ldots, y_m$ are (distinct) variables subscribing to the values of its synthesized attributes. The rationale is that we invoke a task by filling in the inherited positions of the form –the inputs– and by indicating the variables that expect to receive the results returned during task execution –the subscriptions– . A (business) rule R with underlying production $s_0 \to s_1 \ldots s_k$, which we note as $P[r] :: s_0 \to s_1[r] \ldots s_k$, is expressed using the following notation:

$$
\begin{array}{l}
s_0(p_1, \ldots, p_n) \ = \\
\textbf{input}(r, z_1, \ldots, z_l) \\
\ \textbf{do} \\
\qquad (y_1^{(1)}, \ldots, y_{m_1}^{(1)}) \leftarrow \mathbf{s_1}[r](t_1^{(1)}, \ldots, t_{n_1}^{(1)}) \\
\qquad \ldots \\
\qquad (y_1^{(k)}, \ldots, y_{m_k}^{(k)}) \leftarrow s_k(t_1^{(k)}, \ldots, t_{n_k}^{(k)}) \\
\qquad \textbf{return}\ (u_1, \ldots, u_m)
\end{array}
$$

This functional presentation stresses out the operational purpose of business rules: Each task has an input –inherited attributes– seen as parameters and an output –synthesized attributes– seen as returned values.

- The p_i's are patterns serving as *guards* for the rule.
- Variables z_l inside the **input** directive represent values not directly inherited from parent tasks (including users, r) and which will have to be provided by the user when the rule is chosen. This directive is omitted if no such variables exist in a rule.

- The u_j's describe the synthesized values produced when applying the rule.
- The expressions in the right-hand side are the subtasks that will be associated with the newly created open nodes.

The variables $y_i^{(j)}$ and the variables occurring in patterns are the input variables, they are pairwise disjoint and denote respectively the information synthesized by the subtasks and the information stemming from the context of the node. The $t_i^{(j)}$ and the u_j are terms over the input variables called the *semantic rules*. They provide respectively the values of the inherited attributes of the subtasks and the values of the synthesized attributes of the main task. In this way, the values of attributes determine the rules that are applicable to resolve a task. That is, rules that are applicable at a bud.

Below is a sample grammar that models the beginning of the alarm investigation process described in Section 3. The grammar depicts a service offered by an epidemiologist. We have written in **bold** names of external sorts that make reference to other grammars in distant sites. These sorts therefore have no defining rules in this grammar. The rules are labeled **R1** to **R3** with **R1** and **R3** having parameters which will have to be filled in by the epidemiologist during execution. These parameters indicate the effective users in whose workspaces the external service requests will be made. In **R2** and **R3**, we introduced guards $FALSE$ and $TRUE$. These guards automatically filter which of the two rules with sort *continue* to apply when the first task terminates. If the alarm is seen to be genuine, $TRUE$, the epidemiologist contacts a clinician sending the alarm information and a set of requests $Todos$. The result returned by this external service request is used to run additional analysis *runAnalysis* to confirm or revoke the alarm. Note that when **R3** is applied for instance, all its subtasks become buds (ready for execution). However, the *runAnalysis* bud will have to wait for the other task to complete due to variable dependences. This shows that though no predefined ordering exists between subtasks, an ordering can be introduced using variables synthesized within the subtree.

R1:

$\mathbf{investigateAlarm}(Alarm) =$
$\mathbf{input}(pha)$
 $\mathbf{do}$
  $(real) \leftarrow \mathbf{contactPharm}[pha](Alarm)$
  $(results) \leftarrow continue(real, Alarm)$
  $\mathbf{return}\ (results)$

R2:

$continue(FALSE, Alarm) =$
$\mathbf{return}\ (False\_Alarm)$

R3:

$continue(TRUE, Alarm) =$
$\mathbf{input}(cli)$
 $\mathbf{do}$
  $(lab\_res, Patient\_data) \leftarrow$
   $\mathbf{contactClinician}[cli](Alarm, Todos)$
  $(analysis\_res) \leftarrow$
   $runAnalysis(lab\_res, Patient\_data)$
  $\mathbf{return}\ (analysis\_res)$

In some situations it is necessary that semantic rules are not given by plain terms but by more general functional expressions. This is in particular the case when one invokes a service to all individuals playing some particular role. For instance assume that the right-hand side of a rule contains a call of the form $(y) \leftarrow s[r :: R](x, y[r :: R])$. Then each individual playing role R must resolve task s to produce a synthesized result y using an inherited attribute x as well as the values y synthesized by other users of the same role. This means that to produce his results, each user uses the results produced by other users. Now, a variable y synthesized by a sort $s[r :: R]$ can only be used elsewhere in the form $y[r :: R]$, that is, a vector indexed by elements of R. Such vectors of variables cannot be used directly within terms. One might add projections to extract the variable associated with a given individual $r :: R$, which we would write $y[r]$. But in general we are not aware of a particular individual in a given role (and moreover as noted before this set of individuals can vary in time) and one is rather interested in stating conditions such as "there exist $r :: R$ such that $y[r]$" or "for all $r :: R$, $y[r]$", or even "there exist at least 3 individuals $r :: R$ such that $y[r]$", "at least 50% of $r :: R$ verify $y[r]$" etc. when variable y holds a boolean value. More generally one can express the semantic rules using any kind of functional expressions as long as the values of inherited attributes evaluate to terms so that they can be matched against patterns.

For example in scenario described in section 3, when a laboratory test is sought from several biologists (call them pete, bob, john, ...), and they need to each return a lab test result, $labTR$, the value returned by each of them is accessed as follows: $labtTR[pete :: biologist]$, $labtTR[bob :: biologist]$, ... It is also possible to use these in conditions like "there exist $labTR[r :: biologist]$" which checks if there exist any biologist who as already provided a result, "at least 3 $labTR[r :: biologist]$" which asserts that at least three biologist have provided results for the lab test, etc. These conditions coupled with terms are useful to drive the application of other rules at buds.

4.4 Temporal Dependencies and Constraints

Time is a critical and determining factor in user-satisfaction, cost reduction and increased productivity in business processes. In disease surveillance in particular,

timeliness is a major metric used to assert and/or evaluate the effectiveness and relevance of the process. Due to space constraints and given the extensiveness of this topic, we only present high level temporal constraints and dependencies.

We add a time-dimension to the Active-Workspaces model using the concepts defined in [34] and [33] based on Allen's Intervel Algebra[35]. These works identify the following intuitive temporal constraints: *Must Start On (MSO)*, *Must Finish On (MFO)*, *As Soon As Possible (ASAP)*, *As Late As Possible (ALAP)*, *No Earlier Than (NET)* and *No Later Than (NLT)*. These constraints are attached to tasks at specification time and are used by a scheduler to control task start and end times. The specifications of the scheduler are beyond the scope of this paper. All constraints for subtasks are defined and interpreted relative to some reference point, usually the start and end times of the parent task or of sibling tasks. For instance, if data collection and data analysis are subtasks of the disease surveillance task, both subtasks can be defined to start ASAP, but the constraint on the collection task interpreted relative to the parent task and that on the analysis task interpreted relative to the collection task. The MSO and MFO constraints are strict and force the task to start or stop at exactly some time-point from the reference time. If no constraint is specified for a task, it is assumed that the task starts ASAP and finishes ASAP. Such a task is immediately executed when all necessary inputs become available and finishes as soon as all computations complete.

Also, based on the temporal constraints that exist between tasks and their data dependencies, we deduce temporal dependencies between tasks within a business rule. By temporal dependency, we mean any relationship between two tasks in which the start or end of one depends on the start or end of the other. The following four temporal dependency relationships are possible: *Start to Finish (SF)*, *Start to Start (SS)*, *Finish to Start (FS)*, and *Finish to Finish (FF)*. For instance, the data collection and data analysis tasks described in the previous paragraph have an SS relationship written $SS(data\ collection, data\ analysis)$ meaning that data analysis cannot start until data collection has started. In like manner, an SF, FS, or FF relationship between two tasks S_1 and S_2 respectively means that S_2 cannot finish until S_1 starts, S_2 cannot start until S_1 finishes, and S_2 cannot finish until S_1 finishes.

Lastly, additional temporal components, Lag-Time and Lead-Time can be added to temporal dependencies to respectively account for waiting times between tasks and for overlapping tasks.

5. Workspace Evolution

The Active-Workspaces model is adapted for "Open Systems" in which the actions of users are not explicitly specified at design time. These systems are distributed and evolve dynamically with users playing a primordial role. They need to continuously design and run parts of a business process and collaborate with each other. Even when task specifications exist, the effective actions a user undertakes (deciding which task to run, providing input data, etc.) are not specified in advance.

In section 4 we presented two ways in which a workspace can evolve. A user can either explicitly add new productions to an existing grammar or obtain productions defined by another user when their workspaces are synchronized. Also, as noted in [1], the process always interacts with external tools such as databases, email systems, time servers, etc. the so-called side-effects. These external systems complement the active-workspaces model. They allow that real world activities like extracting samples from patients, sending messages, etc. be associated to a rules. Also, these external tools can be used to extract information from enacted artifacts to build dashboards or to feed some local database that are later used to guide the user on her choice of the rule to apply for a pending task. They may, in a more coercive fashion, suggest a specific rule to apply or even inhibit some of the rules. Some information from dashboards or contained in a local database can also be used to populate some input parameters of a rule in place of the user.

6. Conclusion and Future work

In this paper, we characterized the process of monitoring diseases and health conditions of interest for unwanted events as being user-driven and data-centric. The unpredictable nature of the process further justified its knowledge-intensive characteristic. We identified four major modeling use cases that should be fulfilled by disease surveillance process modelers. The active-workspaces model can be extended to offer all four use cases. In this paper, we explicitly present how the active-workspaces model can address the first two use cases namely: dynamic workflow construction and execution, and user interactions. A prototype for the Active Workspaces model which is currently under construction will further demonstrate its pertinence and applicability.

Due to space constraints, we left out certain aspects of the Active Workspaces model which of course we will gladly add in an extended version if this paper is considered for publication. These aspects include:

- The architecture of an Active Workspace: this comprises, the underlying Guarded Attribute Grammar (GAG) engine; the Active Workspaces server that manages users and roles (adding/removing users and/or roles, subscribing/un-subscribing a user from a role), and managing communication between workspaces.
- The user interface: with visualizations of artifact trees, interfacing with external tools, enacting workflows, etc.
- GAG specifications of the key steps in the scenarios described in this paper.
- The extensive formal specification of temporal constraints and dependencies.

We are currently extending the Active Workspaces model to integrate external support for workspace construction using data mining techniques, process mining techniques, and connecting the model with a disease surveillance knowledge base. These will be necessary to demonstrate how the Active Workspaces model can be used to manage uncertainty and effective decision making, two of the use cases identified at the beginning of this paper.

Acknowledgement:
Research by the author NSAIBIRNI Robert FONDZE Jr leading to this result has been partially funded by the US Department of Health and Human Services (DHHS) through the ASIDE PROJECT (www.asideproject.org) pioneered by the Institut Pasteur of Paris and the Centre Pasteur du Cameroun.

References

[1] Eric Badouel, Loïc Hélouët, 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* **15**(3): 6–34, 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. Hull, E. Damaggio, and R. De Masellis. Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. ... *on Distributed event-* ..., pages 51–62, 2011.

[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] Object Management Group Omg. Business Process Model and Notation V2.0. Technical Report December, 2010.

[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] G Texier and Y Buisson. From epidemic outbreak detection to anticipation. *Revue d'Epidemiologie et de Sante Publique*, 2010.

[18] W. M. P. Van Der Aalst. the Application of Petri Nets To Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01):21–66, 1998.

[19] Clementine Calba, Flavie L Goutard, Linda Hoinville, Pascal Hendrikx, Ann Lindberg, Claude Saegerman, and Marisa Peyre. Surveillance systems evaluation: a systematic review of the existing approaches. *BMC public health*, 15:448, 2015.

[20] W. M P Van Der Aalst and a. H M Ter Hofstede. YAWL: Yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[21] W.M.P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639–650, 1999.

[22] W.M.P. van der Aalst, R.S. Mans, and N.C. Russell. Workflow Support Using Proclets: Divide, Interact, and Conquer. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2009.

[23] W Yasnoff, P W O'Carroll, D Koo, R W Linkins, and E M Kilbourne. Public health informatics: improving and transforming public health in the information age. *Journal of public health management and practice : JPHMP*, 6(6):67–75, 2000.

[24] Kamal Bhattacharya, Cagdas Gerede, Richard Hull, Rong Liu, and Jianwen Su. Towards Formal Analysis of Artifact-Centric Business Process Models. *Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM2007. LNCS*, 4714:288–304, 2007.

[25] David Cohn and Richard Hull. Business Artifacts : A Data-centric Approach to Modeling Business Operations and Processes. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 32(3):1–7, 2009.

[26] Hervé Chaudet, Pellegrin Liliane, Jean-Baptiste Meynard, Gaëtan Texier, Olivier Tournebize, Benjamin Queyriaux, and Jean-Paul Boutin. Web Services Based Syndromic Surveillance for Early Warning within French Forces. *Studies in health technology and informatics (2006)*, 124:666–671, 2006.

[27] H P Lehmann, B E Dixon, and H Kharrazi. Public Health and Epidemiology Informatics: Recent Research and Trends in the United States. *Yearbook of medical informatics*, 10(1):199–206, 2015.

[28] Liliane Pellegrin, Charlotte Gaudin, Nathalie Bonnardel, and Hervé Chaudet. Collaborative activities during an outbreak early warning assisted by a decision-supported system (ASTER). *Int. J. Hum. Comput. Interaction*, 26(2&3):262–277, 2010.

[29] Daniel Zeng, Hsinchun Chen, Carlos Castillo-Chavez, and William B. Lober. *Infectious Disease Informatics and Biosurveillance*, volume 28. 2012.

[30] Nkuchia M. M'ikanatha and John K. Iskander. Concepts and Methods in Infectious Disease Surveillance. *John Wiley & Sons, Ltd, 2014*.

[31] Brian E Dixon, Roland E Gamache, and Shaun J Grannis. Towards public health decision support: a systematic review of bidirectional communication approaches. *Journal of the American Medical Informatics Association : JAMIA*, 20(3):577–83, 2013.

[32] Zaruhi R Mnatsakanyan and Joseph S Lombardo. Decision Support Models for Public Health Informatics. *John Hopkins APL Technican Digest*, 27(4):332–339, 2008.

[33] Denis Gagne and André Trudel. *Fisher, L. (Ed), 2008 BPM and Workflow Handbook*, The Temporal Perspective: Expressing Temporal Constraints and Dependencies in Process Models, pages 247-260.

[34] Denis Gagne and André Trudel. Time-bpmn. In *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing*, CEC '09, pages 361–367, Washington, DC, USA, 2009. IEEE Computer Society.

[35] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.