Context-Aware Agile Business Process Engine: Foundations and Architecture

Irina Rychkova, Manuele Kirsch-Pinheiro, and Bénédicte Le Grand

Centre de Recherche en Informatique, Université Paris 1, Panthéon-Sorbonne
90, rue Tolbiac, 75013, Paris, France
{Irina.Rychkova,Manuele.Kirsch-Pinheiro,
Benedicte.Le-Grand}@univ-paris1.fr

Abstract. Future developments for enterprise process management must evolve from the current systems based on rigid, workflow based processes into context-aware, agile dynamic structures, which exploit local adaptability. In this idea paper, we define two forms of process agility. To enable these forms of agility, we present our vision of context-aware business process management based on declarative modeling combined with innovative context management and formal concept analysis. We finally describe the foundations and introduce the architecture of a context-aware agile business process engine (CAPE).

Keywords: business process agility, context awareness, declarative process modeling, formal concept analysis.

1 Introduction

Capacity to timely discover and to efficiently respond to rapid changes in the environment is a major goal of an *enterprise of the future*. According to [23][4], a firm's ability to adapt to dynamic environments depends first on the agility of its business processes. Therefore, design and development of new process management systems enabling process adaptation at run time are essential.

Lamport defines a process as a sequence of events occurring in system [16], where each event is triggered by an action. Accordingly, a business process can be seen as a sequence of events triggered by activities of business process actors or contextual events. The majority of existing methods for business process design follow *imperative* principles, implying that the order of events is predefined. As a result, all meaningful process events need to be determined and corresponding actions need to be predefined at design time. At run time, processes follow the configured model with limited possibilities to deviate from the predefined scenario.

Execution of a business process in a dynamic environment can be compared to navigating a ship towards its destination bay in uncertain waters. Very rarely can a ship follow blindly a predefined path: awareness about its current position and situation as well as navigation skills and dynamic path finding are essential to reach the destination. This idea paper reports on research, which is currently at its early stage of development. In this work, we discuss foundations and propose architecture for a system supporting dynamic and context-aware business process adaptability.

S. Nurcan et al. (Eds.): BPMDS 2013 and EMMSAD 2013, LNBIP 147, pp. 32–47, 2013.

First, we shift the traditional imperative paradigm for process design and exploit *declarative* principles: we represent a business process as finite state machine (FSM) [22] with a *state* representing a process situation at a given time and *state transitions* defining the possible process scenarios. The *triggering events* specify the underlying process semantics, i.e. conditions for state transitions. The FSM formalism makes the notion of *process activity* implicit while putting forward activity outcomes, which are modeled as triggering events. Therefore, the declarative process model focuses on "what" needs to be done in order to achieve the process goal and not on "how" it has to be done. This allows us to handle process events whose order of occurrence is *undetermined* and to define the corresponding handling scenarios at run time.

Navigation in a stormy ocean depends on a skillful skipper and his capacity to select a right action to ensure that no incorrect scenario is executed. We design initial *navigation rules* for process guidance based on Formal Concept Analysis and Galois lattices [2][6]. We specify the resulting process as a set of activities that can be dynamically assembled at run time into one of the (non-forbidden) process scenarios. In general, such process specification can offer infinitely many alternative scenarios and a possibility to deviate from one scenario to another during the execution. We formalize these properties of a process as the **1st form of agility**.

Navigation in a stormy ocean depends on the capacity of the skipper to select a right action at the right moment, and with respect to the current situation: we define the **2d form of process agility** as the ability to monitor and manage the process context and to dynamically select and/or alter the execution scenario accordingly. We extend the declarative process specifications with dynamic context models and mechanisms for dynamic context management [3][19][20].

We design *navigation* rules for processes guidance that handle both process events (events resulting from execution of process activities) and context events in a unified way. This is compatible with the FSM formalism: the nature of events triggering the state transition has no importance. The navigation rules ensure that no incorrect scenario will be executed with respect to a given context situation.

Novel combination of declarative modeling principles, context-awareness and Formal Concept Analysis is the main research contribution of this work. The architecture for a context-aware business process engine (CAPE) summarizes our findings.

The remainder of this paper is organized as follows: in Section 2, we discuss the state of the art related to business process design and associated agility problems; in Section 3 we formalize two forms of business process adaptability. These theoretical foundations are used in Section 4 to specify the architecture of our context-aware agile business process engine (CAPE). We finally illustrate our findings on an example and present the perspectives of this work.

2 Motivation and Related Work

Workflow-based approaches are highly efficient for process design and management assuming that: (*i*) all the events triggered by process activities are well determined; (*ii*) human involvement is limited to "accept", reject" or "select from the list" types of decisions; and (*iii*) the system within which a process is executed can be considered as closed: no external event affecting the process execution can occur.

For a large category of processes, however, these assumptions do not hold: in health care, the same therapy may have different effects on different patients; in insurance, claim processing extensively involves human expertise and decision making; in business, many processes have to cope with evolving economic conditions and frequent changes of customer requirements. The limited capacity of imperative methods to deal with changes has been recognized by both researchers and practitioners [32]. Numerous approaches propose to improve the dynamic process adaptability:

In [26][15][27] the concept of *configurable* process is presented and different modeling formalisms to deal with process configurability are defined. Other works aim at improving run time adaptability through *modification* of the predefined workflow during execution [24]. At run time the process instances follow the preconfigured model with a limited adaptability via predefined variants or deviation.

The Declare framework [21][31] is a constraint-based system that uses a declarative language grounded in temporal logic. This framework supports process flexibility both at design and run time. Compared to our approach, Declare is activity-oriented; contextual information is not considered by this approach.

Solutions for processes characterized by "unpredictability" are reported in numerous works [30][17][1][7]. In [30], the foundations and collection of experience reports on Adaptive Case Management are presented. These works emphasize run time adaptability. Markus et al. [17] propose a framework to support emergent knowledge processes, implemented in the TOP Modeler tool. In [1] process instances are represented as dynamically moving through state space. This approach relies on automated control systems and implements declarative modeling principles. Burkhart et al. [7] propose to explore the capabilities of recommender systems to provide the user with intelligent process-oriented support at run time. While handling dynamic activity selection and configuration of processes "on the fly", the majority of proposed solutions demonstrate only limited capacity to deal with process contextual information in systemic and dynamic way.

Soffer and Yehezkel [29] introduce semantics for a declarative process model based on Generic Process Model (GPM). GPM is state-oriented; it captures the process context and reasons about process goals. Though based on different theories, this approach is the most related to the one presented in this paper.

In [25][28], authors use context information for process definition. Roseman et al. [25] consider that external context elements may influence business processes (e.g. weather influences processes of a call center in an insurance company). They incorporate such elements into business process modeling. Saidani et al. [28] also consider context in business process definition, in particular, the roles played by actors. In these works context information is specified only at *design time*. Mounira et al. [18] propose a process mining architecture to identify context variables influencing process activities. However, no specific model formalizing these variables is proposed.

3 Foundations for CAPE

In this section we define two forms of process agility and present our vision of context-aware business process management based on a fully declarative modeling combined with innovative context management and formal concept analysis.

3.1 Process Agility at Work: Agile Patient Care

As we could see in "House" American TV series[1], Patient diagnostics and treatment processes in a medical ward only partially rely on imperative procedures. The main challenge is to be aware of the patient situation and its evolution and to adjust the treatment accordingly. Contextual parameters that might be relevant and should be managed include (but are not limited to):

- Patient's measurable body conditions (temperature, blood pressure, heart rate);
- Patient's medical record;
- Patient's life style;
- Information about recent workload, leisure activities, trips.

Some of these parameters are stable (e.g. predispositions, allergies, etc.), others can evolve (e.g. new information about the patient's medical history, recent activities), and some others may change several times a day (e.g. body temperature). The capability to immediately react by canceling/prescribing new tests or medications is essential.

3.2 First Form of Process Agility

We define the first form of business process agility as *a capacity to handle unpredictable sequences of system* events. This implies that the order of process activity invocations is defined dynamically, at run time, and depends uniquely on the current situation (process state) rather than on a predefined execution scenario(s).

3.2.1 Declarative Approach to Process Specification

To ensure the first form of agility, we shift the traditional imperative paradigm in process specification and exploit *declarative* principles: we represent a business process as a finite state machine (FSM) – a state-transition system - that allows us to handle process events (and context events – see Section 3.3) with *undetermined* order of occurrence and to define the corresponding scenarios at run time.

A FSM [22] specifies a machine that can be at one state at a time and can perform a state transition as a result of a triggering condition (event or group of events). It is defined by a (finite) set of states and a set of triggering conditions for each transition.

Mathematical model:

A FSM can be defined as a quintuple $< Q, \sum, \delta, q_0, F >$ where:

[1] http://en.wikipedia.org/wiki/House_TV_series

$Q = \{S_0, S_1, \ldots S_n\}$- is a finite set of states;
$\sum = \{E_1, E_2, \ldots E_m\}$- is a finite set of triggering events (input alphabet);
δ: Q x $\sum \rightarrow$ P(Q) is the state transition relation;
$S_0 \in Q$ is the initial state;
$F \subseteq Q$ is the set of final states.

A business process can be modeled as a finite state machine where:

- A FSM *state* s ∈ Q represents a state of the process at a given time. It is described by the states of its parameters (e.g. order: {in preparation, delivered, paid}, patient: {admitted, in diagnostics, under treatment, discharged}).
- FSM triggering events ∑ represent events that may occur during the process execution (e.g. change of patient's body temperature resulting from medication etc.). Note, that process activities are implicit within FSM formalism: only the events resulting from an activity execution are observable.
- FSM transitions δ represent transitions between the process states.
- FSM final states F represent the process outcomes (order delivered, patient discharged, etc.) and can be associated with the process goal.

Representation of a business process using states and transition is not new: [33] presents DecSerFlow language for Declarative modeling of service. Formal semantics for DecSerFlow is based on labeled transition systems. In [1][5], the process execution is presented as a trajectory in the process state space. The process model is specified as a set of formal rules describing the valid paths of the possible trajectories. Our approach extends this paradigm and uses the FSM formalism as a common ground for *process model, context model and formal concept model*.

Example: Agile Patient Care (continued)

To illustrate the FSM formalism, we develop our example on agile patient care and model a (simplified) patient treatment process as a FSM illustrated in Fig. 1:

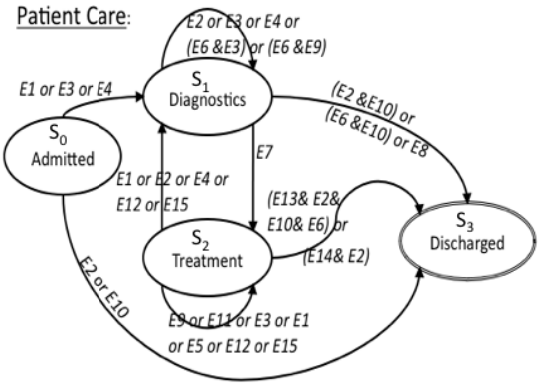


Fig. 1. A finite state machine (FSM) representing a patient treatment process. The events E and the state transitions they trigger are shown as labels

After a patient is *admitted* (S_0) to a medical ward (the initial state), a physician examines him in order to obtain information for *diagnostics* (S_1) and further *treatment* (S_2). Diagnostics may involve one or multiple examinations and/or generic or specific tests. The patient's case is then assessed and a *treatment* is prescribed. During the treatment, additional examinations can reveal new patient's condition and require to modify the assigned therapy and, even, to repeat diagnostics and assessment. Once the therapy is terminated and the patient's good condition is confirmed, the patient is *discharged* (S_3) from the ward. In this example, we identify four states: $Q=\{S_0, S_1, S_2, S_3\}$ *and* six process activities that can be executed during the process and trigger state transitions from S_0*:Admitted* to S_3*:Discharged* states (Table 1).

Table 1. Abstract activities and events defined for the Patient treatment process

Activity		**Avail. at:**	**Process events (Activity outcomes):**	
A1	Physical examination	S0, S1, S2	E1 E2 E3 E4	Confirms the declared symptoms No problem found New symptoms emerged Supplementary medical tests are required
A2	Medical laboratory test	S1, S2	E5 E6 E4	Positive results (anomalies detected) Negative results (no anomalies detected) Supplementary medical tests are required
A3	Specific medical tests	S1, S2	E5 E6 E4	Positive results (anomaly detected) Negative results (no anomaly detected) Supplementary medical tests are required
A4	Case Assessment	S1	E7 E8 E4	Diagnose confirmed, treatment assigned Diagnose not confirmed, patient discharged Supplementary medical tests are required
A5	Therapy	S2	E9 E10 E11 E3 E12 E13	Condition declined (e.g. symptoms increasing) Condition improved (e.g. symptoms decreasing) Side effects emerged New symptoms emerged Stable situation End of therapy
A6	Recovery	S2	E3 E14	New symptoms emerged End of recovery therapy
Context events:				
			E15	New medical / personal evidence received
			E3	New symptoms emerged
			E9	Condition declined, (e.g. symptoms increasing)
			E10	Condition improved (e.g. symptoms decreasing)

According to our formalism, an activity A is described with a pair $<S, E_A>$ where:

- $S \subseteq Q$ is the set of states from which this activity can (but not necessarily will) be invoked;
- $E \subseteq \sum$ is the set of events that can result from the activity execution and can potentially trigger a state transition.

For each activity A, the state transitions that can be triggered upon its termination can be calculated as: δ_A: S x E→ P(S).

For example, the activity A2 (Medical laboratory test) is specified as follows: S: {S1, S2}; E:{E5, E6, E7}. Note that some events can result from a process activity and can be context events (cf. Section 3.3) - independent from activities. (e.g. E10 – condition improved).

3.2.2 Formal Concept Analysis and Galois Lattices

Within our model, the partial ordering of process activities is determined by the state transition relation P(Q). This relation specifies the valid transitions with respect to the process goal (its final state) and ensures that invalid state transitions (e.g. to *discharge* a patient with *critical* temperature) will be avoided. This relation can be specified with *Formal Concept Analysis* (FCA) [2] [6]. FCA is a mathematical theory relying on the use of formal contexts[2] and Galois lattices defined below. The use of Galois lattices to describe a relation between two sets has led to various classification methods [8] [35]. Since then, they have been used in numerous areas to extract hidden knowledge from data. Let us first introduce FCA terminology [12].

Mathematical model:

Let $K = (G, M, I)$ a formal context, where G is a set of *objects*, M is a set of *attributes*, and the *binary relation* $I \subseteq G \times M$ specifies the attributes of the different objects. Derivation operators noted $(.)^I$ are defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$A^I = \{m \in M \mid \forall g \in A : g\, I\, m\} \qquad B^I = \{g \in G \mid \forall m \in B : g\, I\, m\},$$

where A^I is the set of attributes, which are common to all objects from A and B^I is the set of objects which share all attributes from B.

A *formal concept* of the formal context (G, M, I) is a pair (A, B), where $A \subseteq G$ and $B \subseteq M$, $A = B^I$ et $B = A^I$. The set A is called the *extent* of concept (A, B) and B is its *intent*. A concept (A, B) is a specialization of concept (C, D) if $A \subseteq C$, which is equivalent to $D \subseteq B$. This is noted $(A, B) \leq (C, D)$. Reciprocally, the concept (C, D) is a generalization of concept (A, B). The set of all concepts and their partial order relation constitutes a lattice, called Galois lattice of the formal context K.

The major interest of a Galois lattice is the structure it provides through the conceptual clustering of objects according to their common attributes. This allows the identification of the most conceptually significant objects and attributes. Another interest of Galois lattices is that association rules can be inferred automatically from them. Several works have indeed applied FCA to the extraction of relevant association rules [13] or to perform sequential pattern mining [11].

Within our approach, process states Q, triggering events $\sum$ and process activities defined in Section 3.2 form a formal context and can be analyzed using Galois lattices. Process states and activities can be clustered revealing their conceptual properties: For example, we can determine activities that can be executed (or suggested for execution) under given conditions and with an objective to trigger a desired state transition.

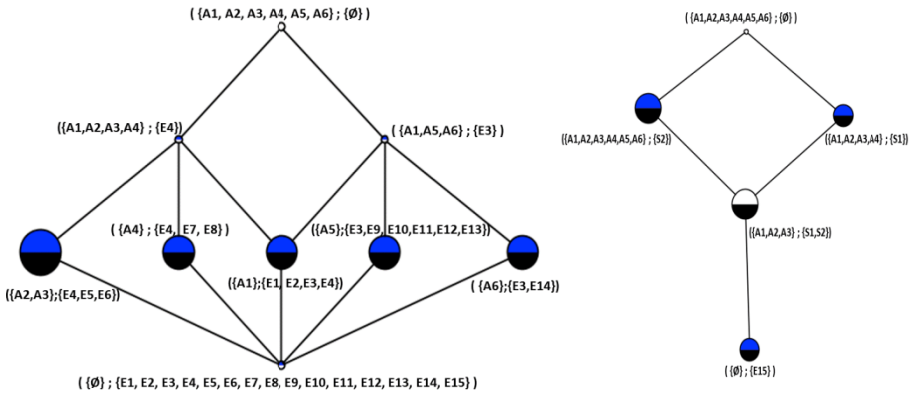
---

[2] The term "formal context" is specific to Formal Concept Analysis and refers to a binary relation. In the following, we will also refer to this as "formal context", as opposed to "context" which represents the environment.

Fig. 2 represents Galois lattices built from the formal context defined in Table 1. The lattice in Fig 2a clusters activities into (possibly overlapping) groups according to the common events shared by the activities within each group. Each node in the line diagram represents a concept of the lattice; its label is a couple of sets corresponding respectively to the extent and the intent of the concept. For example, the concept labeled as {(A2, A3); (E4, E5, E6)] contains the activities A2 and A3 which may all lead to events E4, E5 and E6. The lattice in Fig 2b represents the sets of activities enabled for each state[3].

We exploit Galois lattices as a recommender mechanism, which makes suggestions about activities to execute. This mechanism is explained in Section 4. Each concept of the lattice contains a set of activities (white labels) described by the events they share (grey labels). The links between concepts are generalization/specialization links. Note that nodes inherit events from concepts above them and that they inherit activities from concepts below them. The lattice has an upper bound (top concept), which contains all activities (through inheritance) but no event (i.e. there is no event associated with all activities). The lower bound of the lattice (bottom concept) contains no activity and all events.

Let us consider the concept containing only activity A1 and the concept containing only A4 in its extent (Fig. 2a). We see from the intent that A1 may have events E1, E2, E3 or E4 as its outcome and A4 may lead to events E4, E7 or E8. These two concepts are generalized by the concept {(A1, A2, A3, A4), {E4}}, which contains activities A1, A2, A3 and A4, all described by the common event E4. Along those lines, the concept {(A1, A2, A3), {S1, S2}} in Fig.2b shows the activities described by common states S1 and S2 – the states where these activities can be executed.



a) Activity – Event formal context b) Activity – State formal context

Fig. 2. Galois lattices

[3] We have separated events from states by creating 2 distinct lattices to make our methodology clear, as states are preconditions for activities whereas events are possible outcomes of these activities. It should be noted however that events and states may be associated to activities within a single formal context and therefore a unique Galois lattice.

3.3 Second Form of Process Agility

We define the second form of business process agility as *a capacity to adjust the process execution scenario according to the current contextual situation.* Process activities are assembled at run time, according to observed context[4] and with an objective to trigger a state transition required for achieving the process goal (defined as one of the final states of a FSM).

Dey [9] defines a context as *any information that can be used to characterize the situation of an entity (a person, place or object) that is considered relevant to the interaction between a user and an application*. The notion of context adopted in the literature is mostly user-centric and limited to physical aspects (e.g. location, user preferences, or user device)[19]. Together with Dourish [10], we argue that the notion of context is larger, and includes information related to organization and processes: "context – the organizational and the cultural context, as much as the physical context – plays a critical role in shaping action and also in providing people with the means to interpret and understand action". In our example, patient treatment process can be influenced by the emergence of new symptoms or the arrival of new resources (e.g. new medical people available, new personal evidence, etc.). The second form of business process agility consists in taking into account such context information during process execution.

The context parameters reflect our awareness about external and internal information about the process; they can be observed and measured. Even though context-awareness for business processes is addressed in the literature [25] [28] [18], the lack of formalism for context representation and management persists: many of the proposed context models are static (need to be defined at design), incomplete (consider only limited context information) and are often specific to workflow-based processes.

We argue that the number and kind of context parameters may vary from one situation (or process state) to another making it impossible to exhaustively model all required context information within a single (static) context model. The context model, therefore, needs to be dynamically instantiated from an appropriate metamodel according to specific (evolving) context dimensions.

3.3.1 Dynamic Context Modeling

The way context information can be exploited for business process flexibility depends on what information is observed and how it is represented. According to Najar et al. [19], the formalism chosen for representing context model determines the reasoning methods that can be used to perform system adaptation to the observed context. A context model (i) ensures the definition of independent adaptation processes and (ii) isolates this process from context acquiring techniques. The same applies to context-aware business process. We claim that the process context information should be acquired, modeled and formally analyzed at run time in order to adapt business process execution and to ensure business process flexibility.

[4] The term of "context" represents the environment.

Several context models have been proposed in the literature [19] [3]. Based on these, we define a common meta-model presented in Fig. 3. In this meta-model, we consider context as a set of context elements that are observed for a given subject (e.g. the patient, a device, a resource, etc.). Each subject can be associated with multiple context elements (location, status, etc.); for each context element, we observe values that can dynamically change. Subject and context elements can be semantically described using ontologies. Ontologies are considered as interesting to represent context elements [19]. They provide ways to semantically describe these elements and their relationships, but also reasoning mechanisms, notably inference rules.

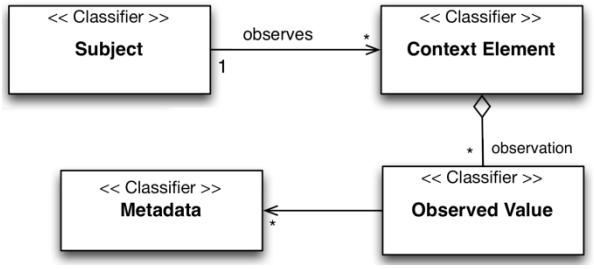


Fig. 3. Context meta-model considering context as a set of context elements

Mathematical model:

Based on context ontology, we represent events as logical constraints over context concepts and observed values. We formalize the context of a subject *s* in a time *t* as follows: $Context(s,t) = \{ Element(s, ce) \}$,
where $Element(s,ce)$ represents the observed value of the context element *ce* for the subject *s*.

Within our approach, process states Q defined using FSM formalism in Section 3.2 can be extended with the context information: Each state can be associated with a (set of) subject and its contextual elements to observe. FSM triggering events $\sum$ represent both (i) *context events o*ccurring during the process execution and/or (ii) events resulting from executed process activities – *process events.* Note that process activities are implicit within FSM formalism: only the events resulting from an activity execution are observable. As a result, a FSM processes both *contextual* and *process* events in a unified way. These events can be expressed using logical conditions on observed values of contextual elements.

In our example, a subject *s* – patient – is observed in all the process states. The contextual elements *ce* associated with a patient include his body temperature, blood pressure, patient's location, record etc. Patient's condition can decline (patient's temperature may evolve from '36.5°C' to '39.7°C') or new evidence about the patient can be received by e-mail. Triggering event E9 (condition declined – see Table 1) can be expressed using the following condition: $Element(\#patient, \#temperature) > 38.5$.

3.3.2 Context Model and Concept Analysis

Fig. 1 shows both the process and the context events that are taken into account by the patient treatment process. Context events affect state transitions and play the role of triggers, similarly to activity outcomes. For example, the patient treatment (S2) can be changed either as a result of physical examination outcomes (E1) or as a response to the new symptoms (E11), or because a new evidence about the patient (e.g. allergies, predispositions, past incidents, etc.) (E15).

Galois lattices are particularly adapted to our context-aware approach: we use Galois lattices to analyze FSM triggering events (see Section 3.2) regardless their nature (process events or context events). Indeed, FCA has already been successfully coupled to context modeling [34]. The authors have proposed a spontaneous, personalized and dynamic way of defining and joining user communities. They use a lattice-based classification and recommendation mechanism that analyzes the interrelations between communities and users, and recommends new communities, based on user interests and preferences.

4 Architecture for CAPE

The functionality of a process engine assuring the agility forms introduced above can be described as follows: this engine enables the activities for execution, captures the events (internal or external) and generates state transitions (Fig. 4). Considering that the process objective is specified by its final state (or states), the engine should also assure an appropriate guidance for the decision maker (a human agent): it recommends him the activities that would maximize chances to trigger a transition leading towards the process goal.

We define the following primary elements for a context-aware process engine: activity repository, context monitor and navigation manager, as illustrated in Fig. 4.

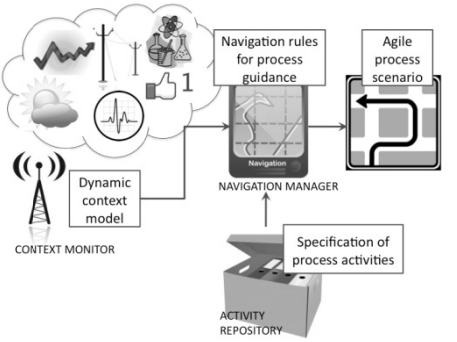


Fig. 4. CAPE architecture: the context monitor, the activity repository and the navigation manager

4.1 Activity Repository

Activity repository represents a set of activities related to a business process. Each activity is specified with states S from which it can (but not necessarily must) be executed during the process, and with the set of events E representing its possible outcomes as defined in Section 3.2.1. The pair <S,E> is an activity signature. These signatures are further used by Navigation Manager. The activity repository represents crew skills and technical capacity of a ship.

4.2 Context Monitor

Context monitor is in charge of observing, at run time, context elements and subjects from the environment. Its role in CAPE architecture is similar to a watchman in a ship: it observes navigation conditions and reports them to the navigation manager.

Context monitor is based on the context meta-model described earlier. Similar to [20], it recognizes context elements through plugins (software components dynamically connected to the architecture), which feed the architecture with dynamic information about a given context element from a subject (e.g. a plugin reading patient's heart rate or medical resource's location from his/its id card). Context monitor can be dynamically extended, by adding new plugins, for observing new context elements and subjects. Context values dynamically observed by context plugins measure the process position in the process state space (for instance, a new plugin for observing the availability of a new analysis equipment). This position is further used by the navigation manager, which recognizes context events and interprets them accordingly.

4.3 Navigation Manager

Navigation manager makes navigation decisions, like a skipper in a ship. It determines one (or several) plausible activity to execute with respect to the process goal and the contextual situation. Specifically, navigation manager takes into account the current state of the process, the contextual parameters and the signatures of activities defined in the activity repository. Based on the *navigation rules*, it determines a set of activities *enabled* in a given situation and calculates those of them that will have a highest probability to result in the desired outcome.

Navigation engine is the core element of CAPE architecture: it links together the other elements. We illustrate the functioning of CAPE on our example:

Example: Agile Patient Care (continued)

Once a patient has been admitted (S0) and the result of his physical examination (A1) has confirmed the declared symptoms (E1), he is in state S1. The targeted final state is S3: discharge the patient. The goal of the navigator is to provide recommendations in order to reach this state as quickly as possible, while respecting medical protocols and taking into account the contextual situation. In the following, we describe the sequence of operations performed by the navigator to meet this goal.

1. Selection of the next transition towards the targeted final state

Fig. 1 illustrates the various possible paths from one state to another. The underlying graph is directed, nondeterministic, with possible cycles. From this graph the

navigator computes the most efficient path from S1 to S3, both in terms of length and of events triggering probability. Let us suppose here that this ideal path consists in using the direct link from S1 to S3. The next step, in our example, is to trigger the transition from S1 to S3.

2. Identification of relevant activities

The navigator now has to identify the events, which may[5] trigger a transition from S1 to S3. According to our FSM (Fig.1) and the state transition map, the triggering conditions for a transition S1 - S3 are the following:

- Additional physical examinations and/or medical laboratory tests reveal no anomaly or problem and the patient himself feels better - (E2 | E6)&E10, or
- Case assessment does not confirm the diagnosis (E8)

To identify the activities that can be of a maximum utility with respect to our objective (transition S1-S3):

- Navigator identifies the activities in our repository that can ensure a desired outcome (a combination of events described above) using the Galois lattice illustrated in Fig. 2. For our example, the event E2 can result from the activity A1; E6 from A2 or A3; E10 from A5 or spontaneously as an external event, as the patient's condition may improve independently from any activity from the medical staff.
- Navigator selects those activities that are enabled at the state S1 (according to their specification) using the Galois lattice illustrated in Fig. 2. For our example, the activities A1, A2, A3, A4 are available in S1 (In Diagnostic) state, whereas A5 is not.

Thus, the activities A1 – A4 are potential candidates for execution in the state S1.

3. Selection of recommended activities based on their utility

We calculate an utility of each scenario as its likelihood to trigger the transition S1-S3. For our example, A5 is not available at S1 and therefore, E10 can be expected only as a context event (i.e. we can observe this event when it happens but cannot control it). The following viable scenarios can be evaluated:

a) A physician prescribes medical tests and/or makes additional physical examinations according to the declared symptoms. The spontaneous improvement of the patient is expected.
b) A physician assesses the case based on the patient's history (and examinations made upon patient's admission to the ward).

A combination of these scenarios is also possible.

Depending on the probabilities of transition S1-S3 resulting from each of the aforementioned scenarios and also, on the probability to obtain the event E2 as a result of A1, E6 from A2 or from A3, and E8 from A4, the navigator may recommend specific activities to the medical staff.

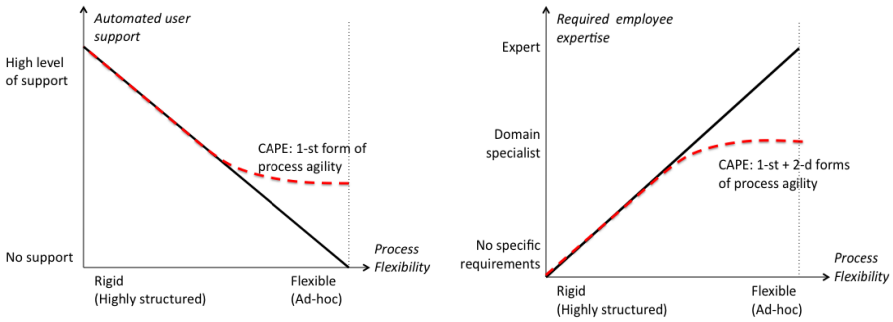
[5] Our graph is nondeterministic: one event can trigger multiple state transitions (e.g. E11 – side effects emerged – can be handled both in the therapy states or can trigger a transition to the diagnostics state. This nondeterminism, in general case, can be resolved by refining the states and transitions, and by specifying the new navigation rules. Alternatively, a probability p can be associated with each of the possible transitions.

5 Discussion and Future Work

While providing a rich toolbox for process modeling, verification, and post-execution analysis, traditional workflow-based information systems are considered to be inflexible. Therefore, organizations nowadays are searching for a compromise between the automated user support and process flexibility at run time. Fig. 5a illustrates this dependency: the x-axis shows a degree of process flexibility (from highly structured to unstructured, ad-hoc processes); the y-axis shows a level of automated user support that can be assured by the system. The solid line depicts the dependency: the more flexible a process, the lower the user support provided for it; and conversely: the higher the expected level of user support, the more structured a process has to be.

According to Burkhart et al. [7], "With increasing runtime flexibility, employees are confronted with an expanding decision space and they are needed to possess more expertise in dealing with the processes they are involved in." Fig. 5b illustrates this dependency: the x-axis shows a degree of process flexibility; the y-axis shows a level of user expertise required for handling this process. The solid line depicts that more flexible a process is, the higher expertise level of an employee should be.

The two forms of agility formulated in this paper aim at relaxing the requirements both for human expertise for a given level of process flexibility and for process rigidity (predefined structure) for a desired level of automation. *Declarative approach* for process design and use of formal methods enable a set of automated techniques for process analysis and validation based on model checking and theorem proving. Thus, they improve the level of automated user support allowing maximum run time flexibility. *Context awareness and formal concept analysis* enable automated recommendations and identification of alternatives. Their joint use provides flexible guidance to end users at run time and supports them with an expertise required for the process handling.



a) Process flexibility vs. Automated user support

b) Process flexibility vs. Level of employee expertise required

Fig. 5. Relations between process flexibility and automated support / required level of user expertise. The solid line depicts the current trends in BPM; the dashed line depicts how CAPE architecture presented in this work can possibly change these trends

This idea paper reported on research, which is currently at its early stage of development. According to the IS research framework [14], (i) we specified our **business problem** as a lack of automated support for business process agility; (ii) then we defined a relevant **knowledge base** for our research and outlined the foundations for CAPE; (iii) we **built our design artifacts**: we introduced the two forms of process agility – **the constructs** to be used for reasoning about business processes; we also defined **a model** and **a method** for specification of agile business processes based on FSM abstraction and formal concept analysis. We extended this model with the dynamic context model. We combined these artifacts in and proposed the architecture for context-aware agile business process engine (CAPE).

This work accomplishes the first part of the "build-evaluate" loop [14]. **Evaluation** of our designed artifacts and their **refinement** will be addressed in future work. More specifically, we envisage to demonstrate the utility of our proposed architecture, first, by developing detailed scenarios, then, by simulating them, and, eventually, by implementing CAPE architecture and studying its usability in real business environment. Usability metrics for CAPE will be also discussed in future publications.

References

1. Andersson, T., Bider, I., Svensson, R.: Aligning people to business processes experience report. Software Process: Improvement and Practice (2005)
2. Barbut, M., Monjardet, B.: Ordre et classification. Algebre et combinatoire, Tome 2, Hachette (1970)
3. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (2010)
4. Bider, I., Johannesson, P., Perjons, E.: Do workflow-based systems satisfy the demands of the agile enterprise of the future? In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 59–64. Springer, Heidelberg (2013)
5. Bider, I.: Towards a Non-workflow Theory of Business Processes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 1–2. Springer, Heidelberg (2013)
6. Birkhoff, G.: Lattice Theory, 1st edn. Amer. Math. Soc. Pub., Providence (1940)
7. Burkhart, T., Weis, B., Werth, D., Loos, P.: Towards Process-Oriented Recommender Capabilities in Flexible Process Environments–State of the Art. In: 45th Hawaii Int. Conf. on System Science, HICSS, pp. 4386–4395 (2012)
8. Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In: 10th Conf. on Machine Learning, pp. 33–40. Kaufmann (1993)
9. Dey, A.: Understanding and Using Context. Personal and Ubiquitous Computing 5, 4–7 (2001)
10. Dourish, P.: Seeking a foundation for context-aware computing. Human Computer Interaction 16(2-4), 229–241 (2001)
11. Egho, E., Jay, N., Raissi, C., Napoli, A.: A FCA-based analysis of sequential care trajectories. In: 8th Int. Conf. on Concept Lattices and their Applications (2011)
12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
13. Hamrouni, T., Ben Yahia, S., Nguifo, E.M.: GARM: Generalized Association Rule Mining. In: CLA 2008, pp. 145–156 (2008)
14. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28(1), 75–105 (2004)

15. La Rosa, M., Dumas, M., ter Hofstede, A.H.M., Mendling, J.: Configurable multi-perspective business process models. J. Information Systems 36(2) (2011)
16. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM 21(7), 558–565 (1978)
17. Markus, M.L., Majchrzak, A., Gasser, L.: A design theory for systems that support emergent knowledge processes. Mis Quarterly 26, 179–212 (2002)
18. Mounira, Z., Mahmoud, B.: Context-aware process mining framework for Business Process flexibility. In: iiWAS 2010, Paris (2010)
19. Najar, S., Saidani, O., Kirsch-Pinheiro, M., Souveyet, C., Nurcan, S.: Semantic representation of context models: a framework for analyzing and understanding. In: 1st Workshop on Context, Information and Ontologies (CIAO 2009), European Semantic Web Conference (ESWC 2009), pp. 1–10 (2009)
20. Paspallis, N.: Middleware-based development of context-aware applications with reusable components, PhD Thesis, University of Cyprus (2009)
21. Pesic, M., Schonenberg, H., van der Aalst, W.M.: DECLARE: Full support for loosely-structured processes. In: 11th IEEE Int. Enterprise Distributed Object Computing Conference, EDOC 2007, pp. 287–287 (2007)
22. Plotkin, G.D.: A structural approach to operational semantics (1981)
23. Raschke, R.L., David, J.S.: Business process agility (2005)
24. Reichert, M., Rinderle, S., Dadam, P.: ADEPT workflow management system: In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 370–379. Springer, Heidelberg (2003)
25. Rosemann, M., Recker, J., Flender, C.: Contextualization of Business Processes. Int. J. Business Process Integration and Management 1(1), 46–60 (2007)
26. Rosemann, M., van der Aalst, W.M.P.: A Configurable Reference Modelling Language. Information Systems (2007)
27. Rychkova, I., Nurcan, S.: Towards Adaptability and Control for Knowledge-Intensive Business Processes: Declarative Configurable Process Specifications. In: 44th Hawaii Int. Conf. on System Sciences, HICSS, pp. 1–10 (2011)
28. Saidani, O., Nurcan, S.: Towards Context Aware Business Process Modeling. In: 8th Workshop on Business Process Modeling, Development, and Support (BPMDS 2007), CAiSE 2007 (2007)
29. Soffer, P., Yehezkel, T.: A state-based context-aware declarative process model. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 148–162. Springer, Heidelberg (2011)
30. Swenson, K.D.: Mastering The Unpredictable: How Adaptive Case Management Will Revolutionize The Way That Knowledge Workers Get Things Do, p. 354 (2010)
31. Van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science-Research and Development 23(2), 99–113 (2009)
32. Van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data & Knowledge Engineering (2005)
33. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
34. Vanrompay, Y., Ben Mustapha, N., Aufaure, M.A.: Ontology-based User Preferences and Social Search for Spoken Dialogue Systems. In: 7th Int. WS. on Semantic and Social Media Adaptation and Personalization, pp. 113–118 (2012)
35. Wille, R.: Line diagrams of hierarchical concept systems. Int. Classif. 11 (1984)