ProcessBase: A Hybrid Process Management Platform

Moshe Chai Barukh and Boualem Benatallah

School of Computer Science & Engineering
The University of New South Wales, Sydney – Australia
{mosheb,boualem}@cse.unsw.edu.au

Abstract. Traditional structured process-support systems increasingly prove too rigid amidst today's fast-paced and knowledge-intensive environments. Commonly described as "unstructured" or "semi-structured" processes, they cannot be pre-planned and likely to be dependent upon the interpretation of human-workers during process execution. On the other hand, there has been a plethora of Social and Web 2.0 services to support workers with enhanced collaboration, however these tools are often used ad-hoc with little or no customisable process support. In order to address these challenges, we thus present: "*ProcessBase*", an innovative Hybrid-Processes platform that holistically combines *structured*, *semi-structured* and *unstructured* activities. Our task-model proposed encapsulates a spectrum of process specificity, including: structured to ad-hoc Web-service tasks, automated rule-tasks, human-tasks as well as lifecycle state-tasks. In addition, our hybrid process-model enables the "evolution/agility" from unstructured to increasingly structured process design; as well as the notion of "cases" representing repeatable process patterns and variations. We further propose an incremental process-knowledge acquisition technique for curation, which is thereby utilised to facilitate efficient "re-use" in the form of a context-driven recommendation system.

Keywords: Business Process Management, Hybrid Process, Case Management, Service Oriented Architecture, Web-Services, Web 2.0.

1 Introduction

Many processes are difficult to model due to the ad-hoc characteristics of these processes [1], which often cannot be determined before the process begins. While certain characteristics could be predicted, the actual activities and ordering may differ. More so, information may only become available during the process, thus making human-beings and knowledge-workers in control of these processes [2–5].

An emerging discipline to deal with such processes (commonly referred to as "unstructured" or "semi-structured" processes) is Case-Management. The importance is well recognised since knowledge-workers who make up 25-40% of a typical workplace play a vital role on the long-term success of an enterprise [3]. However, while research in this area correctly highlights the importance of combining knowledge with process [3], and calls for increased flexibility [4],

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 16–31, 2014.

most existing implementations are yet to embrace these requirements [2]. As a result, case-management has often only intensely been managed manually, in circumstances where traditional BPM suites would otherwise prove too rigid.

On the other end of the spectrum, major advances in Web-technology, including Web 2.0, crowd- and cloud- computing, has also influenced a new wave of process-support. Cultivated by the services-oriented paradigm, *Software-as-a-Service (SaaS)* tools are extensively being used to complete everyday tasks, [6, 7]. Albeit there remains significant shortcomings: (i) Firstly, the re-use of such ready-made Web-apps often implies conforming to the embedded work-process allowing little room for customisation; (ii) Alternatively, even if a collection of such tools are used for different portions of the process, this inevitably leads to "shadow processes" [8], often only informally managed by e-mail or the like; (iii) Yet if none of the above suffices, a support system would have to be "developed" from scratch, and even when leveraging existing apps, it still requires considerable technical/programming skills; (iv) Finally, without the required skills or lack of resources, it may likely resort to "homebrewed" solutions (e.g. spreadsheets and/or office applications), resulting in untidy and hard-to-maintain products.

Not surprisingly, process-support technology has thus typically been portrayed in two extremes [2, 9]: Either highly *structured* and almost procedurally executed processes supported by BPMS, WfMS, ERP, etc.; whilst many *unstructured* and ad-hoc processes strive for support from various SaaS tools. The reality however, is that most processes rarely fit into only one of these two extremes; rather they usually comprise (sub-)fragments of various types of organisational activities that include a mix (or spectrum) of structured activities to other activities that may be very ad-hoc, [9, 10]. Moreover, there exist a variety of process paradigms/models/representations that are best suited to a specific domain. For example, BPEL for structured flows, state-models for monitoring, rules for ad-hoc functionality, etc. While systems may support a partial-hybrid approach with one or two types, they generally compete rather than leverage inter-domain support. The main challenge is thus facilitating *end-to-end* process-support.

To address this, we propose *ProcessBase — A Hybrid-Process Management Platform*, consisting of an extensible platform that encourages a new breed of hybrid-process-driven applications. We define domain-specific types and functions to represent process abstractions from *structured* to *unstructured* activities; which is thereby exposed via a programmatic API in order to provide enhanced in-App process-support. Moreover, *ProcessBase* acts as a knowledge-base, for the efficient "curation" and "re-use" of process-knowledge, supported via a context-based recommendation system driven by an incremental acquisition technique. More specifically, we make the following main contributions:

- In Section 3, we begin by analysing the technological landscape, as well as tracking the evolution from structured to unstructured process support, with respect to existing work. We use this to demystify the various concepts, identify key characteristics and provide directives for our proposed work.
- In Section 5, we then propose a *domain-specific model for hybrid-processes*. Most importantly, we support: (i) The ability to capture possibly repeatable "patterns"; whilst also (ii) Allowing the "evolution/agility" from an early

unstructured to increasingly structure design. We address this by separating a hybrid-process *definition* from the actual executional *tasks* by introducing a *logical* layer that enables *modularity*, *virtual-ordering* and *hierarchy* of activities. Moreover, to support (iii) Case-based "variations" we integrate the notion of *cases* and *variations*. The logical layer thus enables the organisation of process-knowledge without governing the execution. We model the executional components as a variety of 5 task-types, aiming to cover the range of process-specificity, including: Structured (i.e. BPEL) tasks, ECA Rule-tasks, Human-tasks, Web-services tasks as well as lifecycle State-tasks.
- In Section 6, we propose a novel *context-based recommendation system* for more efficient "re-use" of process-knowledge, via an incremental knowledge-acquisition technique. The first work to propose this, as far as we know.
- In Section 7, we delineate our proposed programmatic *Hybrid Process-as-a-Service (HPaaS)* API. In Section 8, we evaluate our work by implementing the reference scenario over a comparative experimental study. Finally, we conclude with a summary and directions for future work at Section 9.

2 Motivating Example

Consider the *"Software Development Change-Management"* process, as illustrated in the BPMN model shown in Figure 1.

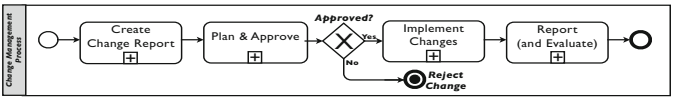

Fig. 1. Software Development Change Management Process

While the overall pattern may be followed, the specifics may vary between case-to-case. For example, a "formal" software-project often view changes as a non-typical event requiring a strict approval-process. However, even in a "formal" setting, structured activities may exhibit variations, but only based on preconceived conditions; an example is illustrated in Figure 2.

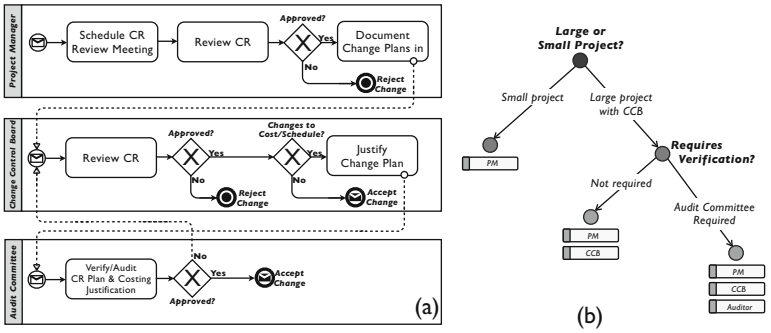

Fig. 2. (a) Formal Software Project Approval Process; (b) Process Variation Tree

In contrast however, "agile" (e.g. eXtreme programming) software-projects embrace change and thus are prone to a simplified approval process that could likely be reduced to a simple human-task, as illustrated in Figure 3. Moreover, in both cases, certain activities could nonetheless be inherently ad-hoc. For example, activities such as: *create change report*, and *implement changes* could directly depend upon the specific project's development environment. Such as, a change report could be generated using *GoogleDrive*, while another project could depend on a documents uploaded to *DropBox*. Likewise, some projects could employ *Git* while others may use *SVN*.

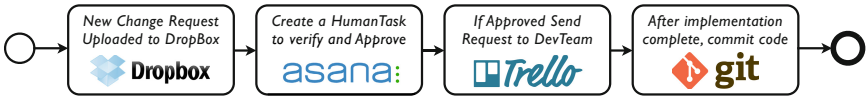

Fig. 3. Agile Software Development Change Management Process

3 Background and Related Work in Hybrid-Processes

Transitioning process-support from structured to unstructured domains, requires harnessing the capabilities that BPM had to offer for its application to unstructured processes; such that these ad-hoc style processes can be comparably visible, measurable and managed [8]. Essentially this means bridging the gap between *structured* and *unstructured* processes. We therefore dedicate this section in understanding the technological evolution and landscape, in order to recognise the potential gaps, from which we derive the main directives of our proposed work.

BPM vs Rule-based Systems. BPM and rule-based systems are two of the most conventional archetypical approaches, for structured versus less-structured support, respectively. BPMSs introduced the process-centric methodology, and offered a high-level model-driven approach that strongly appealed to non-technical domain-experts. However it suffered from a vital lack of agility. Rule-systems on the other hand, while inherently capable of dealing with the executional dynamics of orchestrations, their applicability in non-trivial contexts have meant limited success, due to the number of rules required to describe a process. The synergy therefore, between BPM and Rule-based systems has thus often been explored as a potential way for achieving the best of both worlds.

For instance, in 2008 the OMG joined forces with the BPM community and released the *Semantics of Business Vocabulary and Business Rules (SBVR)* standard. The goal was to express business knowledge in a controlled natural language, albeit it did not directly address the formal integration with process modelling diagrams. *Vanthienen et al.* thus proposed to implement SBVR into the business process management lifecycle using an SOA approach [11], consisting of a three-layer architecture. Similarly, *Agrawal et al.* proposed *Semantics of Business Process Vocabulary and Process Rules (SBPVR)* [12]. *Milanovi et al.* also offered to integrate BPMN with R2ML, developing a new modelling language *rBPMN (Rule-based Process Modelling Language)* [13], which extended existing BPMN elements with rule-based properties. Nonetheless these works are yet to

be well adopted in mainstream, likely because they overarch the extensive range of business rule-types (i.e. integrity, derivation, reaction and deontic rules), thus clouding simplicity with over-rich vocabulary and semantics, [4, 8].

Event Driven Business Process Management (EDBPM). In an similar approach, EDBPM focuses primarily on "event-driven" *reaction-rules*. The motivation has been to merge BPM with Complex Event-Processing (CEP) platforms via events produced by the BPM-workflow engine or any associated (and even distributed) IT services. In addition, events coming from different sources and formats can trigger a business process or influence its execution thereof; which could in turn result in another event. Moreover, the correlation of these events in a particular context can be treated as a complex, business-level event, relevant for the execution of other business processes. A business process, arbitrarily fine or coarse grained, can thus be choreographed with other business processes or services, even cross-enterprise. Examples of such systems include: *jBPM* [14], and *RunMyProcess* [15]. However, these systems are usually implemented where the respective components sourced from BPM or CEP operate almost independently, (e.g. event-modeller vs. process-modeller; event-store vs. process-store; rules-engine vs. process-engine; process-instances vs. rules-instances, etc.). In fact, the only thing connecting these two systems together is the event-stream at the low-level, albeit this does not really directly benefit the process-modeller. These systems also tend to be dominated somewhat by the structured process side (e.g. a rudimentary process is always required, and even basic changes require restarting the process). They also do not encompass the full range of process-specificity support, however nonetheless they do provide the crucial step-ahead towards at least a partial hybrid-process methodology.

Case Management. As mentioned, the "case-management" paradigm has also been recognised as a promising approach to support semi-structured processes. Unlike traditional business-process systems that require the sequence and routing of activities to be specified at design-time (as otherwise they will not be supported) - case-management is required to empower the ability to add new activities at any point during the lifecycle and when the need arises, [4]. At the same time it also requires the ability to capture possibly repeatable process patterns, and variations thereof [3, 16]. However, although there has been several efforts to push this, (e.g. OMG is currently working on an appropriate standardisation), at present there are no concrete all-encompassing frameworks capable of adequately supporting these requirements. *Emergent Case Management* provides a slightly more modernised twist, suggesting a bottom-up approach. *Bohringer* [2], proposes such a platform which petitions the use of social-software (e.g. tagging, micro-blogging and activity-streams) in a process-based manner. It claims to empower people to be at the centre of such information systems, where the goal is to enable users to assign activities and artifacts independent of their representation to a certain case, which can be dynamically defined and executed by users. However, this work is currently only at its concept stage and is yet to be implemented and tested. Likewise case-management in general is rather yet only considered "a general approach" rather than being a "mature tool category".

Characteristics & Requirements for Hybrid-Processes. In light of the above analysis, we identify the following dimensions that may be used to *characterise* process-systems. Bridging the technological gap and avoiding fragmented support thus *requires* collectively supporting the various facets over a holistic model. Accordingly, this has precisely been the motivation of our proposed work.

Process Paradigms. Refers to the type of control-structure the process-system can handle, [17–20]. There are three main facets identified within this dimension: (A) *structured*; (B) *semi-structured*; and (C) *unstructured*.

Process Representation-Models/Languages. Represents the language, model or interface offered to the process-designer. Again, there are three identified facets: (A) *Activity-centric* models the flow of control between activities based on a specified sequence; (B) *Rules-centric* define statements that express a business policy, thus defining or constraining the operations of a "process", in a declarative manner; and (C) *Artifact-centric* have tasks (actions or events) defined in the context of process-related artifacts, as first-class citizens, [21–24].

4 *ProcessBase* Architecture Overview

Figure 4 illustrates the system design and interaction of the main components of the *ProcessBase* system, which are elucidated as follows:

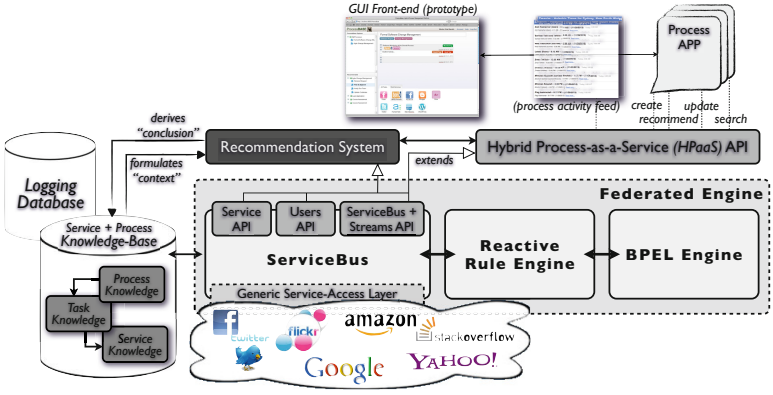

Fig. 4. ProcessBase System Architecture

The **Web-Services Layer** represents APIs available over the Internet, whose integration in processes offers vital potential: Services act as rich and real-time sources of data, as well as, providing functionality (software and tools), infrastructure building-blocks, collaboration mechanisms, visualisations, etc.

The **ServiceBus** components (leveraged from our previous work [6, 7]) acts as the middleware between outside Web-services and the platform back-end. Most importantly, it helps solve the inherent heterogeneity challenges: Services may differ in representation and access protocols, (e.g. SOAP vs. REST); as well as in message-interchange formats, (e.g. JSON, XML, CSV, or Media files, etc.). Moreover, APIs are constantly subject to change, (e.g. due to system updates,

when data-structures are improved, errors fixed or new components introduced). The *ServiceBus* overcomes this by utilising our previously proposed *Unified Services Representation Model (USRM)*, which abstracts low-level logic and masks heterogeneity thereby exposing a common access and data-interchange interface. It relies on service-integration logic organised in the Services Knowledge-Base.

The **RuleEngine** enables reactive capabilities, via Event-Condition-Action (ECA) rules. When event patterns are matched, and their conditions are satisfied, the specified actions are then fired. Likewise, the **BPELEngine** component is delegated for executing and managing BPEL processes, which may represent a complete or more often a segment of a larger unstructured process. In both cases, these two engine components are federated with the *ServiceBus* for detecting and logging instance as well as activity-level events of running processes.

There are two **Storage** components: The *Knowledge-Base (KB)* extends the Services programming base with knowledge about hybrid-processes. Moreover, this combined Services+Process KB also maintains incremental *knowledge-capture rules* (different from event-rules mentioned earlier). Such a rule serves to map a process "context" (rule-condition) to an existing process "definition" (rule-conclusion). In this manner, when a new process starts formulation, and a similar "context" can be detected, the **Recommendation System** may suggest the closest matching process "definition" that could be re-used, either directly, or to create a template from. The other storage component is the *Logging-Database*, which curates ongoing process-instance data and artifacts, such as events and interactions data from services and tasks, for later analysis and/or processing.

Finally, applications can be written over the programmatic **Hybrid Process-as-a-Service (HPaaS) API**, which may be embedded into applications. (For instance, we have implemented a prototypical GUI front-end, for better support.)

5 Domain-Specific Model for Hybrid-Processes

In Figure 5, we presents the overall hybrid-process model:

5.1 Hybrid-Process Definition

At the highest level, a `HybridProcess` contains a set of logical `Activities` which in turn contains a set of functional `Tasks`. This provides a *light-weight* definition model for hybrid-processes. During execution, instance data may be recorded as `ProcessInstanceMessage`s, which encapsulates a `Fact`, representing a data artifact from either an *`event`* or *`action`*, with various structure depending on its origin. A process may also require a `CorrelationCondition` (or set thereof called a `CorrelationSet`) to be specified, this is required in order to correctly partition messages and thus distinctly manage different running process-instance. We support the following types of correlation-conditions:

– *Key/Reference* based, refers to two messages being correlated if they share a field that are equal in value, (e.g. $M_x.f_i = M_y.f_j$);

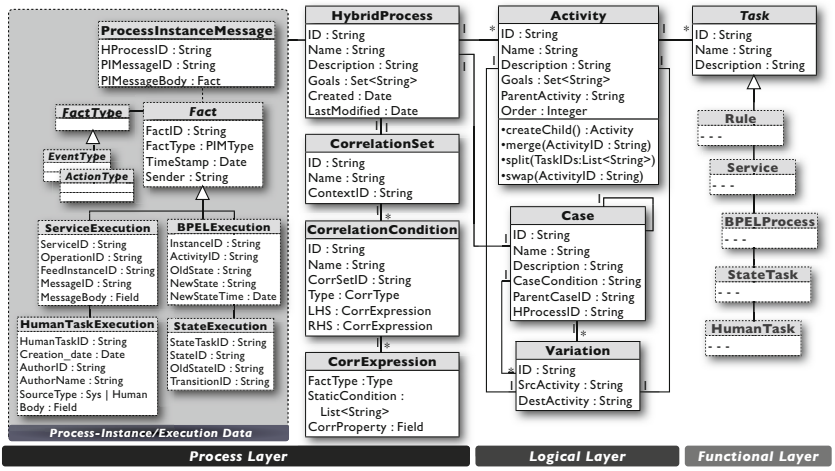

Fig. 5. Domain-Specific Model for Hybrid-Processes

- *Direct Reference* based, refers to when a message M_x can be directly correlated with a message M_y, by introducing a special uniquely identifiable field from M_x into M_y, (e.g. $M_x.f_i = M_y.\hat{f}$, where $\hat{f} := f_i$);
- *Semantic* based, refers to a special *reference-based* condition, where a relationship between messages M_x and M_y can be inferred using semantic-knowledge that are computed over the relative fields, (e.g. $email \equiv e-mail$).

5.2 Process Cases and Variations

In the absence of a process-schema based approach, unstructured processes are usually defined and managed as *instance*-only. However, even while such unstructured or ad-hoc processes may not be precisely repeatable, they may often have recurring elements and "patterns" that could be "re-used". Moreover, a pattern could also exhibit various case-based "variations". This is often expressed as *templates*, accessible via a template-library, or derived from existing instances. In our platform we adopt the latter approach. However, in either case a template

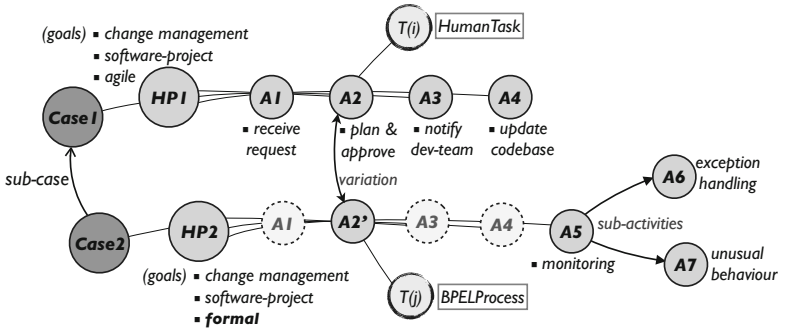

Fig. 6. Example of Hybrid Process Model (showing key nodes and relationships)

represents a light-weight, customisable at run-time abstraction for organising process components; unlike process-schemas, which effectively pre-define an executable program. We therefore refer to this layer as the *logical* layer, as it does not govern the actual execution/function of the hybrid-process.

To support process "patterns": An `Activity` entity represents a *logical* work-item, that may: be `order`ed between one another; contain a sub-activity (or chain thereof); `merge` with another activity; `split` into two sub-activities; as well as `swap` ordering if needed. Likewise, in order to support case-based "variations": we adopt the notion of `Case` and `Variation`. When a new hybrid-process begins it belongs to a root-case; sub-cases may then be defined which inherit the parent's constituent activities and tasks, with the exception of any variations specified.

As an example, consider the first hybrid-process (denoted *HP1*) shown in Figure 6, based on the *agile change-management process* we described earlier. It contains four activities, and since for an agile project, the *plan and approve* activity is implemented using a *HumanTask*. However, consider now a variation to this process for a *formal software project* instead. A new hybrid-process *HP2* can be defined as a sub-case, such that all activities are inherited (thus avoiding replication). However the designer specifies a variation: a new Activity *A2'* to replace the original *A2*, having the approval task implemented as a *BPELProcess* task instead. Similarly, additional activities (such as *A5-7*) can also be added.

5.3 Functional Tasks

We have identified a set of 5 domain-specific functional tasks that together encapsulate the required range of process-specificity:

Automated Rule Task. An automated `RuleTask`, shown in Figure 7, represents an ECA-style rule with a set of `EventTypes` and `ActionTypes`. A specialised `TemporalEvent` is also defined to enable triggering rules at specific times, or as part of temporal event conditions. `Conditions` are expressed as the triple $< path\_expr, comparator, value >$. Where a *path_expr* defines the query to reach the attribute value of the event message instance. While some event-types may have predefined message-models (e.g. a BPEL instance event), other types of messages may vary (e.g. from Web-services). However, as mentioned, the heterogeneity challenges are solved due to the *ServiceBus* middleware offering a uniform message-interchange format, [6, 7]. In the remaining sections for each task-type we define event and action types that extend the abstract event and action types defined here.

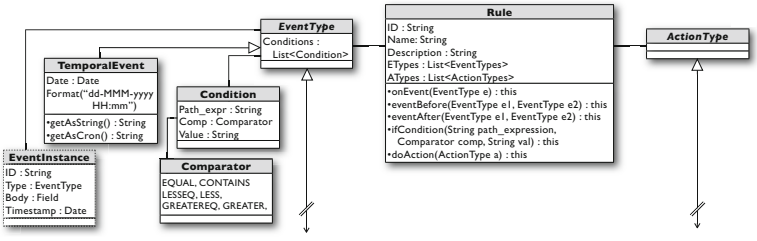

Fig. 7. Automated Rule Task Data-Model

Web-Service Task. To enable integration of Web-services, as illustrated in Figure 8, we define a `ServiceTask`. This is basically precisely akin to the model of `Service` defined in our previous work, [6, 7], we thus omit elaborating on the details. We support both WSDL and RESTful services, albeit the model could be abstracted into a unified set of entities, namely: `Service`, `OperationType`, `FeedType` (reference to a generic feed-endpoint), and `FeedInstance` (a specialised instance feed-type, with specific parameters defined, e.g. `&id=123`).

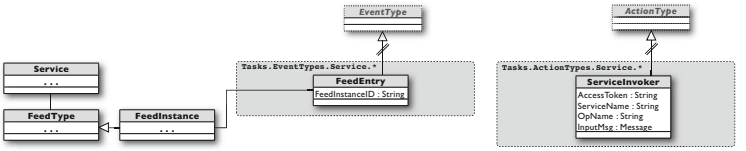

Fig. 8. Web-Service Task Data-Model

Structured-Process Task. Although well-structured process-support technology may not be feasible for a complete overall process, these frameworks are nonetheless useful in the case of routine and repeatable "fragments" of the overall process. We implement this type of task as a `BPELProcess`, as illustrated in Figure 9.

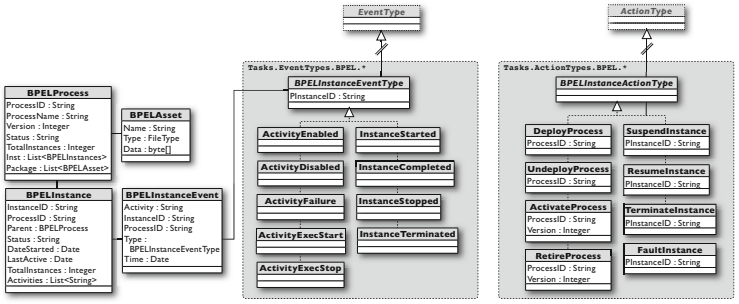

Fig. 9. Structured Process BPEL Task Data-Model

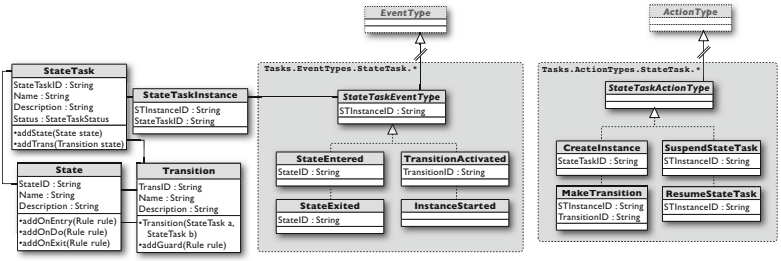

Fig. 10. Lifecycle State-Task Data-Model

Lifecycle-State Task. Data and resources are central to any process. However, since many process systems tend to be activity-centric, data-artifacts manipulated by these processes are seen as second-class citizens. In contrast, the "artifact"-centric approach stipulates an artifact modelled to have both an *information* and *lifecycle* model, [22]. We implement this archetype, as a Lifecycle `StateTask` as illustrated in Figure 10, consisting of `State`s and `Transition`s. Modelled after a finite-state-machine (FSM), there are three kinds of state-actions (in our model represented as a `Rule` - where a pure action could just be with no event or condition): (i) *onEntry* is activated when the state is entered; (ii) *onDo* after finishing the entry-action and anytime while in that state; (iii) *onExit* when the state is deactivated. Likewise, in FSM terms, a transition is modelled as an *event*, *guard* and *action*. A guard is effectively a condition, which thus means we again re-use the notion of `Rule` which can thereby also be attributed to the `Transition` entity.

Human Task. Although there are several options for integrating human-worker frameworks into our platform, we have chosen to leverage *Asana*, due to its popularity, integration with other tools, and ease-of-use [25]. The model for a `HumanTask` has been illustrated in Figure 11. The entity `Story` represents any change or human/system activity performed during the execution of some human-task; which we represent in our system as events.

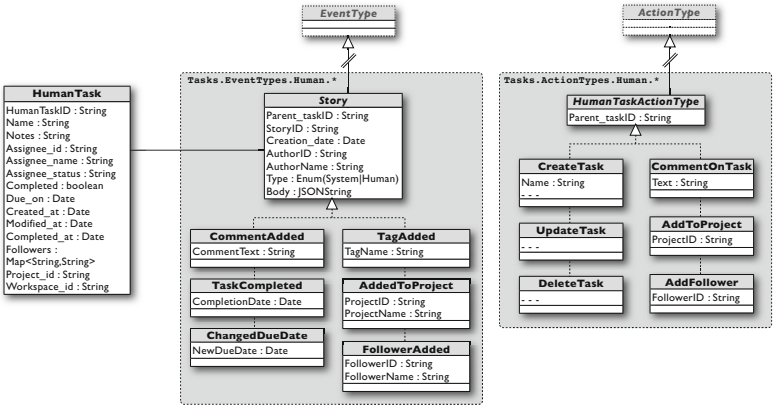

Fig. 11. Human-Task Data-Model

6 Context-Based Recommendation System

Current techniques for re-use usually utilise process schema or template libraries. However, this does not prove efficient with a large and increasing number of process definitions, cases, and variations. In *ProcessBase*, we propose a novel automated recommendation approach based on the currently detected "context" of a hybrid-process definition. This means given the context, the system may suggest the closest matching existing process definition, that could then be re-used and/or customised as required.

A context is matched based on existing process-knowledge. The hybrid-process model we presented thus far inherently curates this type of knowledge. However to make the system efficient in responding with a recommendation, we extend our model with knowledge-acquisition rules (denoted `kRules` to differentiate from ECA-rules) to incrementally capture process-knowledge. This means whenever an existing process is created anew, modified, or sub-case created, a new/updated context triggers a new knowledge-rule to be incrementally added.

To model `kRules`, we adopt the knowledge acquisition method *Ripple-Down-Rules (RDR)* [26], due to its simplicity, and its successful application in many other domains. However, it has never been applied to process-knowledge acquisition for the purpose of context-based re-use. We make use out of the *Single-Conclusion RDR (SCRDR)* approach, where the general form of this rule has two main components: if *[condition]* (i.e. when does the rule apply), do *[conclusion]* (i.e. what to recommend as a result). The knowledge-rules are organised in a tree-like hierarchy (in the order they are created). A new rule may be added as a child to another rule via a ***true*** branch (denoted $\triangle+$, if the current rule condition validates true but extra conditions are added), or via a ***false*** branch (denoted $\triangle\pm$, if the condition does not (fully) match, so a variation of the rule is created). During evaluation, starting at the top node, the inference engine tests whether the next rule node is true or false. If a rule node is true, the engine proceeds with the child nodes and again tests if they are false or true. The last rule node that evaluates to true is the conclusion given.

Applying this to our model, the `kRule` *condition* is thus represented as a hybrid-process "context", while the *conclusion* is a pointer to the matching hybrid-process, e.g. "HProcess_id". There could in fact be many different dimensions to formulate a process-context, for example: (i) the set of *goals* of the overall process; (ii) the set of *goals* of each constituent activity; (iii) the order of activities; (iv) the hierarchy of activities; (v) the type of Tasks assigned to each activity, etc. We've found the conjunction of the first two sufficient enough to formulate a viable context, (however, this could be customised as required).

To give an example (shown in Figure 12): Consider a designer starts with a blank process and simply specifies the process goals "software-project", "change-management". Assuming so far only an *agile* process has been defined, the system finds ***Rule1*** and thus recommends ***HP1***. The designer may then create a sub-case

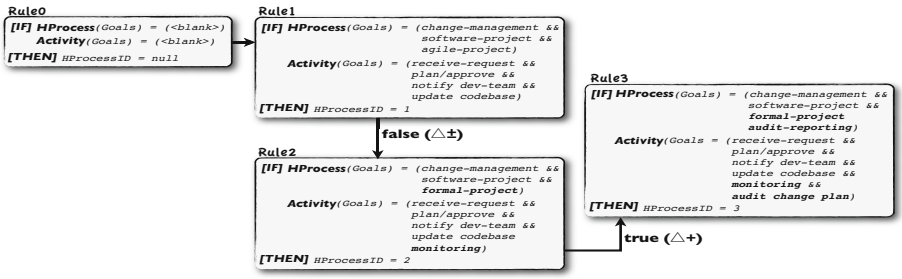

Fig. 12. Incremental Knowledge-Rules `kRules`

(for a "formal-project"), resulting in a new process *HP2*. However, since a new context has been defined, a new (*Rule2*) is added, as a variation to the parent. In another scenario, consider now the designer requires a more sophisticated formal change-process with *auditing*. Again, the designer may specify the relevant goals. This time we assume the process previously exists from someone else. Starting at the top, *Rule1* is checked however, since it evaluates to *false*, it proceeds down the "false" branch and encounters *Rule2*. Since it evaluates to *true*, it proceeds down the "true" branch ending at *Rule3*. The system thus recommends *HP3*, from which a *copy/template* can be created in order to be re-used.

7 Hybrid-Process-as-a-Service *(HPaaS)* API

We have exposed *ProcessBase* over a set of APIs. The benefit of this means hybrid-processes may be embedded in-Apps, and thus integrated at the programmatic level. We provide both a Java-client library (for backend integration); as well as, a RESTful API (suitable for front-end integration). In this section we show snippets of code highlighting the main features of the platform.

We have organised into two main APIs, as follows: Firstly, the `TasksAPI.*` offer CRUD operations over individual tasks (in cases where it could be used stand-alone and outside any process definition - this could be useful in very ad-hoc domains); Secondly, the `ProcessBase.*` API offer CRUD operations on hybrid-process definitions, in addition to other required operations.

TasksAPI.*	▪ `String id = create(Task t)` : registers the task on the knowledge-base	C
	▪ `Task task = get(String t_id)` : gets the task from the knowledge-base	R
	▪ `bool result = update(Task t)` : updates the task on the knowledge-base	U
	▪ `bool = result = delete(String t_id)` : deletes the task from the knowledge-base	D
	▪ `Fact fact = execute(ActionType at)` : executes the specified *Action*, returns data as a Fact.	
	▪ `String sub_id = subscribe(EventType et)` : creates a subscription to this *Event*	
	▪ `void addEventListener(sub_id, (@EventCallback)Object, String "handler_id")` : registers an event callback handler - such that events are asynchronously "pushed" to the callback	
ProcessBase.*	▪ `String id = create(HybridProcess hp)` : registers the hybrid-process on the knowledge-base	C
	▪ `HybridProcess hp = get(String hp_id)` : gets the hybrid-process from the knowledge-base	R
	▪ `bool result = update(HybridProcess hp)` : updates the hybrid-process on the knowledge-base	U
	▪ `bool result = delete(String hp_id)` : deletes the hybrid-process from the knowledge-base	D
	▪ `bool result = suspend(String hp_id)` : suspend processing of the specified hybrid-process	
	▪ `bool result = resume(String hp_id)` : resume processing of the specified hybrid-process	
	▪ `HybridProcess hp = createSubCase(String hp_id)` : create a sub-case of the h-process	
	▪ `HybridProcess hp = createTemplate(String hp_id)` : create a template of the h-process	
	▪ `HybridProcess hp = createCopy(String hp_id)` : create a copy of the specified h-process	
	▪ `HybridProcess hp = recommend(HybridProcess hp)` : invoke the recommendation system	

Using again the examples we described in Sections 5.2 and 6, starting with a simple/empty process *(Line 1)*, the recommender system can be invoked *(Line 2)*, which finds the closest process being for an "agile" software project. The designer can modify this by creating a sub-case (or template) *(Line 3)*, and then proceed to define a new "monitoring" activity, *(Lines 4-6)*. The monitoring activity posts a tweet-notification (e.g. "thanks for your patience!"), in the event the approval process has taken longer than 1-week to complete.

```
1.  HybridProcess hp = new HybridProcess.HybridProcessBuilder("formal_chng_mngmt")
                                        .addGoal("software-project")
                                        .addGoal("change-management");
2.  HybridProcess hp_ = ProcessBase.recommend(hp);
3.  hp = ProcessBase.createSubCase(hp_);

4.  BPELProcess f_approval = new BPELProcess.BPELProcessBuilder("formal_approval")
                                            .asset("approval.bpel");

5.  Rule delayed_approval = new Rule.RuleBuilder("delayed_approval")
                                    .eventAfter(new TemporalEvent("0,0,*,*,0"),
                                                new ..BPEL.InstanceCompleted())
                                    .onCondition(…)
                                    .doAction(new ServiceInvoker("Twitter","postTweet",…));

6.  Activity monitoring = new Activity.ActivityBuilder("monitoring")
                                      .setGoals("…")
                                      .addTask(Twitter)          //"Twitter" ServiceTask
                                      .addTask(approval)         //BPELProcess Task
                                      .addTask(delayed_approval) //Automated Rule Task
7.  hp.addActivity(monitoring);
    ...
8.  ProcessBase.create(hp);
```

8 Evaluation and Analysis

A total of 5 potential platforms were considered: *Enhydra Shark*, *JawFlow*, *JBoss jBPM*, *JOpera*, *WFMOpen*; out of which the top-2 were chosen based on shortest installation and initial testing time; and quality of user-docs. We conducted 3 experimental studies, each comprising 4 comparative executional alternatives: (a) *ProcessBase*; (b) *jBPM*; (c) *JOpera*; and (d) Pure Java code-based solution.

Usability Study. Usability involves the criterion of *learnability* and *efficiency*. The former assessed by the time to install and run the initial tests: *ProcessBase* resulted in $26m$ and $39m$ respectively; compared to averages of $159m$ and $193m$. The latter measured as the time to successfully implement the reference scenario: *ProcessBase* again proved superior in $72m$, in contrast with an average of $203m$.

Productivity Study. Given the task was fixed, productivity was measured based on the total number of lines-of-code (LOC) in order to produce the solution. The results in Figure 13(a-d), presents a distributed measure of LOC.

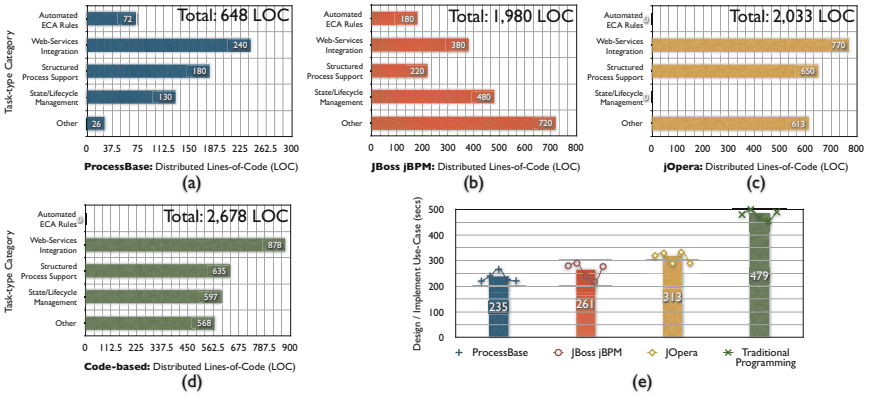

Fig. 13. Experimental Results of *Productivity* and *Performance* Studies

Performance Study. Finally, we measured the round-trip time (i.e. from when the change-request was issued, until updates were committed into *Git*). We repeated this study 5 times, taking the median; results as presented in Figure 13(e).

9 Conclusions

The work in this paper as far as we know, proposes the *first* all-encompassing complete hybrid-processes platform. Moreover, we propose an architecture where existing process-support technology (either domain-specific or partial-hybrid) can be leveraged, rather than compete with each other. In addition, our work is the first to propose a novel recommendation system using process context-detection - based on an incremental knowledge acquisition technique. Experimental results shows superior performance across all evaluated dimensions: *usability*, *productivity* and *performance*. Above all, we are optimistic this work provides the foundation for future growth into a new breed of enhanced process-support.

References

1. Marjanovic, O.: Towards is supported coordination in emergent business processes. Business Process Management Journal 11(5), 476–487 (2005)
2. Böhringer, M.: Emergent case management for ad-hoc processes: A solution based on microblogging and activity streams. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 384–395. Springer, Heidelberg (2011)
3. BPTrends: Case management - combining knowledge with process (July 2009)
4. de Man, H.: Case management: A review of modelling approaches (January 2009)
5. Holz, H., Rostanin, O., Dengel, A., Suzuki, T., Maeda, K., Kanasaki, K.: Task-based process know-how reuse and proactive information delivery in tasknavigator. In: Conference on Information and Knowledge Management, pp. 522–531 (2006)
6. Barukh, M.C., Benatallah, B.: ServiceBase: A programming knowledge-base for service oriented development. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013, Part II. LNCS, vol. 7826, pp. 123–138. Springer, Heidelberg (2013)
7. Barukh, M.C., Benatallah, B.: A toolkit for simplified web-services programming. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds.) WISE 2013, Part II. LNCS, vol. 8181, pp. 515–518. Springer, Heidelberg (2013)
8. Olding, E., Rozwell, C.: Expand your bpm horizons by exploring unstructured processes. Technical Report (2009)
9. Bernstein, A.: How can cooperative work tools support dynamic group process? bridging the specificity frontier. In: CSCW, pp. 279–288. ACM, New York (2000)
10. Keen, P.G., Morton, M.S.S.: Decision support systems: an organizational perspective, vol. 35. Addison-Wesley Reading, MA (1978)
11. Vanthienen, J., Goedertier, S.: How business rules define business processes. Business Rules Journal 8(3, March) (2007)
12. Agrawal, A.: Semantics of business process vocabulary and process rules. In: Proceedings of the 4th India Software Engineering Conference, pp. 61–68. ACM (2011)
13. Milanovic, M., Gasevic, D., Wagner, G.: Combining rules and activities for modeling service-based business processes. In: 2008 12th Enterprise Distributed Object Computing Conference Workshops, pp. 11–22. IEEE (2008)

14. JBoss: jbpm, http://www.jboss.org/jbpm/
15. RunMyProcess, https://www.runmyprocess.com/
16. Swenson, K.D., et al.: Mastering the unpredictable. How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done
17. Berry, P.M.: Intelligent workflow - state of the art in workflow
18. Manolescu, D.A.: Workflow enactment with continuation and future objects. SIGPLAN Not 37(11), 40–51 (2002)
19. Wang, J., Kumar, A.: A framework for document-driven workflow systems. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 285–301. Springer, Heidelberg (2005)
20. Manolescu, D.: Micro-workflow: A workflow architecture supporting compositional object-oriented software development. Technical report, USA (2000)
21. Bhattacharya, K., Caswell, N.S., Kumaran, S., Nigam, A., Wu, F.Y.: Artifact-centered operational modeling. IBM Systems Journal 46(4), 703–721 (2007)
22. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. 32(3), 3–9 (2009)
23. Bhattacharya, K., Gerede, C.E., Hull, R., Liu, R., Su, J.: Towards formal analysis of artifact-centric business process models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
24. Bhattacharya, K., et al.: A model-driven approach to industrializing discovery processes in pharmaceutical research. IBM Syst. J. 44(1), 145–162 (2005)
25. Asana: Asana project managements
26. Richards, D.: Two decades of ripple down rules research. Knowledge Eng. Review 24(2), 159–184 (2009)