International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 06 | Sep-2015 www.irjet.net p-ISSN: 2395-0072

# A Survey of Contemporary Process Evolutionary Systems

**Mr. PaaraanjiMohan Suresh[1], Assistant Professor. Koteswarrao. kadiventi.[2]**

[1]*M. Tech Student, Department of Computer Science, Audisankara College of Engineering (Autonomous), Andhra Pradesh, India.*

[2]*Assistant Professor, Department of Computer Science, Audisankara College of Engineering (Autonomous), Andhra Pradesh, India.*

---

**Abstract:** ***Traditional information systems struggle with the requirement to provide flexibility and process support while still enforcing so medegree of control. Accordingly, adaptive process management systems(PMSs)have emerged that provide some flexibility by enabling dynamic process changes during runtime. .Based on the assumption that these process changes are recorded explicitly,we present two techniques for mining change logs in adaptive PMSs; However the dynamic nature of the modern business environment means these processes are subject to an increasingly wide range of variations and must demonstrate flexible approaches to dealing with these variations if they are to remain viable .The change processes discovered through process mining provide an aggregated overview of all changes that happened so far.Using process mining as an analysis tool wesho win this paper how better support can be provided for truly flexible processes by understanding when and why process changes become necessary.***

***Keywords****:Process-awareinformationsystems;processmining,changemining, flexibility*

## 1.INTRODUCTION

In order to retain their competitive advantage in today's dynamic marketplace, it is increasingly necessary for enterprises to streamline their processes so as to reduce costs and to improve performance. Moreover, it is clear that the economic success of an organisation is highly dependent on its ability to react to changes in its operating environment. To this end, Process- Aware Information Systems (PAISs) are a desirable technology as these systems support the business operations of an enterprise based on models of both the organisation and its constituent processes. PAISs encompass a broad range of technologies ranging from systems which rigidly enforce adherence to the underlying process model, e.g., workflow systems or tracking systems, to systems which are guided by an implied process model but do nothing to ensure that it is actually enforced, e.g., groupware systems[3]. Typically, these systems utilise an idealised model of a process which may be overly simplistic or even undesirable from an operational standpoint. Further-more the models on which they are based tend to be rigid in format and are not able to easily encompass either foreseen or unforeseen changes in the context or environment in which they operate. Up to now, there have not been any broadly adopted proposals or standards offering guidance for developing flexible process models able to deal with these sorts of changes. Instead most standards focus on a particular notation (e.g., XPDL, BPEL, BPMN, etc.) and these notations typically abstract from flexibility issues. Process flexibility can be seen as the ability to deal with both foreseen and unforeseen changes, by varying or adapting those parts of the business process that are affected by them, whilst retaining the essential format of those parts that are not impacted by the variations. Or, in other words, flexibility is as much about what should stay the same in a process as what should be allowed to change [5],[6]. Different kinds of flexibility are needed during the BPM life cycle of a process. Based on an extensive survey of literature and flexibility support offered by existing tools1, a range of approaches to achieve process flexibility have been identified. These approaches have been described in the form of a taxonomy which provides a comprehensive catalogue of process flexibility approaches for the control-flow perspective

## 2. PROBLEM ANALYSIS

Recently, many efforts have been under taken to make PAISs more flexible and several approaches for adaptive process management, like ADEPT, have emerged in this context. The basic idea behind the se approaches is to enable users to dynamically evolve or adapt process schemes such that they fit to changed real world situations. More precisely, adaptive PMSs support dynamic changes of different process aspects(e.g.,control and dataflow)at different levels(e.g.,process instance and

process type level).In particular, ad-hoc changes conducted at the instance level(e.g.,to add, delete or move process steps during runtime)allow to adapt single process instances to exceptional or changing situations. Usually, such ad-hoc deviations are recorded in change logs ,which results in more meaningful login formation when compared to traditional PAISs.

Adaptive process management technology has not addressed the fundamental question what we can learn from the additional change log information (e.g., how to derive potential process schema optimizations from a collection of individually adapted process instances. In principle, process mining techniques offer promising perspectives for this. However, current mining algorithms have not been de-signed with adaptive processes in mind,

## 3. PROPOSED SYSTEM

We have focused on the analysis of pure execution logs instead (i.e., taking obviously, mining adhoc changes in adaptive PMSs offers promising perspectives as well. By enhancing adaptive processes with advanced mining techniques we aim at a PMS framework, which enables full process life cycle support. However, the practical implementation of such a framework in a coherent architecture, let alone the integration of process mining and adaptive processes is far from trivial.

In particular, we have to deal with the following three challenges.

1. First, we have to determine which runtime information about adhoc deviations has to be logged and how this information should be represented to achieve optimal mining results.
2. Second, we have to develop advanced mining techniques that utilize change logs in addition to execution logs.
3. Third, we have to integrate the new mining techniques with existing adaptive process management technology.

This requires the provision of integrated tool support allowing us to evaluate our framework and to compare different mining variants.

## 4. CONTRIBUTION:

In our previous work, with ADEPT and ProM we have developed two separate frameworks for adaptive processes and for process mining respectively. While ADEPT has focused on the support of dynamic process changes at different levels, ProM has offered a variety of process mining techniques, e.g., for discovering a Petri Net model or an Event Process Chain (EPC) describing the behaviour observed in an execution log. So far, no specific ProM extension has been developed to mine for process changes.

This paper contributes new techniques for mining adhoc process changes in adaptive PMSs and discusses the challenges arising in this context. We first describe what constitutes a process change, how respective information can be represented in change logs, and how these change logs have to be mined to deliver insights into the scope and context of changes. This enables us, for example, to better understand how users deviate from predefined processes. We import ADEPT change logs in ProM, and introduce mining techniques for discovering change knowledge from these logs. As result, we obtain an abstract change process represented as a Petri Net model. This abstract process reflects all changes applied to the instances of a particular process type. More precisely, a change process comprises change operations (as Meta process steps) and the causal relations between them. We introduce two different mining approaches based on different assumptions and techniques.

**BACKGROUND INFORMATION**

This paper is based on the integration of two existing technologies: process mining and adaptive process management. This section gives background information needed to understand the implications and leverages of their combination

### 4.1 Process Mining

Although the focus of this paper is on analyzing change processes in the context of adaptive process management systems, process mining is applicable to a much wider range of information systems. There are different kinds of Process-Aware Information Systems (PAISs) that produce event logs recording events. Examples are classical work-flow management systems (e.g. Stafware), ERP systems (e.g. SAP), case handling systems, PDM systems, CRM systems (e.g. Microsoft Dynamics CRM), middleware (e.g. IBM's Web Sphere), hospital information systems, etc. These systems all provide very detailed information about the activities that have been executed. The goal of process mining is to extract information (e.g., process models, or schemas) from these logs.

Process mining addresses the problem that most "pro-cess owners" have very limited information about what is actually happening in their organization. In practice there is often a significant gap between what is predefined or supposed to happen, and what actually happens. Only a concise assessment of the organizational reality, which process mining strives to deliver, can help in verifying pro-cess schemas, and ultimately be used in a process redesign effort.

As indicated, process mining starts with the existence of event logs. The events recorded in such a logs should be ordered (e.g., based on timestamps) and each event should refer to a particular case (i.e., a process instance) and a particular activity. This is the minimal information needed. However, in most event logs more information is

present, e.g., the performer or originator of the event (i.e., the person / resource executing or initiating the activity), the timestamp of the event, or data elements recorded with the event (e.g., the size of an order). In this paper, we assume that event logs are stored in the MXML format. MXML is an XML-based format for representing and storing event log data, which is sup-ported by process mining tools such as ProM. Using our ProM import tool, it is easy to convert data originating from a wide variety of systems to MXML

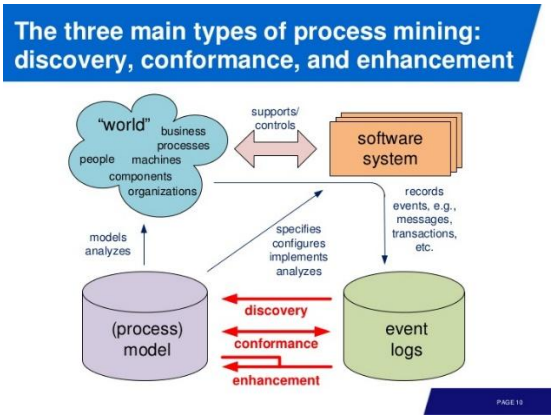


**Fig** : Overview showing three types of process mining: (1) Discovery, (2) Conformance, and (3) Extension.

The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs (e.g., in MXML for-mat). Clearly process mining is relevant in a setting where much flexibility is allowed and/or needed and therefore this is an important topic in this paper. The more ways in which people and organizations can deviate, the more variability and the more interesting it is to observe and analyse processes as they are executed. We consider three basic types of process mining [5].

• Discovery: There is no a-priori process schema, i.e., based on an event log some schema is constructed. For example, using the alpha algorithm a process schema can be discovered based on low-level events [5].

• Conformance: There is a-priori process schema. This schema is used to check if reality conforms to the schema. For example, there may be a process schema indicating that purchase orders of more than one million Euro require two checks. Another example is the checking of the four eyes principle. Conformance checking may be used to detect deviations, to locate and explain these deviations, and to measure the severity of these deviations

• Extension: There is an a-priori process schema. This schema is extended with a new aspect or perspective, i.e., the goal is not to check conformance but to en-rich the schema. An example is the extension of a pro-cess schema with performance data, i.e., some a-priori process schema is used to project the bottlenecks on. Another example is the detection of data dependencies that affect the routing of a case and adding this information to the model in the form of decision rules

At this point in time there are mature tools such as the ProM framework, featuring an extensive set of analysis techniques which can be applied to real process enactments while covering the whole spectrum depicted in Figure 1. Any of the analysis techniques of ProM can be applied to change logs (i.e., event logs in the context of adaptive process management systems. Moreover, this paper also presents two new process mining techniques exploiting the particularities of change logs [5].

**4.2. Adaptive Process Management**

In recent years several approaches for realizing adaptive processes have been proposed and powerful proof-of-concept prototypes have emerged. Adaptive PMSs like ADEPT, for example, provide comprehensive runtime information about process changes not explicitly captured in current execution logs. Basically, process changescantake place at the type as well as the instance level: Changes of single process instances may have to be carried out in an ad-hoc manner to deal with an unforeseen or exceptional situation. Process type changes, in turn, refer to the change of a process schema at the type level in order to adapt the PAIS to evolving business processes. Especially for long-running processes, such type changes often require the migration of a collection of running process instances to the new process schema.

PMS frameworks like ADEPT support both ad-hoc changes of single process instances and the propagation of process type changes to running instances. Examples of ad-hoc changes are the insertion, deletion, movement, or replacement of activities. In ADEPT, such ad-hoc changes do not lead to an unstable system behaviour, i.e., none of the guarantees achieved by formal checks at build-time are violated due to the dynamic change. ADEPT offers a complete set of operations for defining instance changes at a high semantic level and ensures correctness by introducing pre-/post-conditions for these operations. Finally, all complexity associated with the adaptation of instance states, the remapping of activity parameters, or the problem of missing data is hidden from users. To deal with business process changes ADEPT also enables schema adaptations at the process type level. In particular, it is possible to efficiently and correctly propagate type changes to running instances.

**4.3 A FRAMEWORK FOR INTEGRATION**

Both process mining and adaptive processes address fundamental issues prevalent in the current practice of BPM implementations. These problems stem from the fact that the design, enactment, and analysis of a business processare commonly interpreted, and

implemented, as detached phases.

Although both techniques are valuable on their own, we argue that their full potential can only be harnessed by tight integration. While process mining can deliver reliable information about how process schemas need to be changed, adaptive PMSs provide the tools to safely and conveniently implement these changes. Thus, we propose the development of process mining techniques, integrated into adaptive PMSs as a feedback cycle. On the other side, adaptive PMSs need to be equipped with functionality to exploit this feedback information.

The framework depicted in Figure 2 illustrates how such an integration could be realized. Adaptive PMSs, visualized in the upper part of this model, operate on pre-defined process schemas. The evolution of these schemas over time spawns a set of process changes, i.e., results in multiple process variants. Like in every PAIS, enactment logs are created, which record the sequence of activities executed for each case. On top of that, adaptive PMSs can additionally log the sequence of change operations imposed on a process schema for every executed case, producing a set of change logs. Process mining techniques that integrate into such system in form of a feedback cycle may be positioned in one of three major categories:

• Change analysis: Process mining techniques from this category make use of change log information, be-sides the original process schemas and their variants. One goal is to determine common and popular variants for each process schema, which may be promoted to replace the original schema. Possible ways to pursue this go alare through statistical analysis of changes or their abstraction to higher-level schemas. From the initially used process schema and a sequence of changes, it is possible to trace the evolution of a pro-cess schema for each case. Based on this information, change analysis techniques can derive abstract and aggregate representations of changes in a system. These are valuable input for analysis and monitoring, and they can serve as starting point for more involved analysis (e.g., determining the circumstances in which particular classes of change occur, and thus reasoning about the driving forces for change).

• Integrated analysis: This analysis uses both change and enactment logs in a combined fashion. Possible applications in this category could perform a context-aware categorization of changes as follows. Change process instances, as found in the change logs, are first clustered into coherent groups, e.g. based on the similarity of changes performed, or their environment. Subsequently, change analysis techniques may be used to derive aggregate representations of each cluster. Each choice in an aggregate change representation can then be analysed by comparing it with the state of each clustered case, i.e. the values of case data objects at the time of change, as known from the original pro-cess schema and the enactment logs. A decision-tree

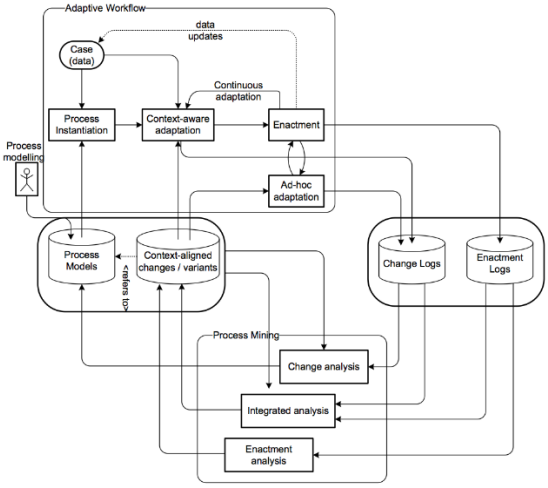


**Fig:** Integration of Process Mining and Adaptive Process Management

Analysis of these change clusters provides an excellent basis for guiding users in future process adaptations, based on the peculiarities of their specific case.

• Enactment analysis: Based solely on the inspection of enactment logs, techniques in this category can pinpoint parts of a process schema which need to be changed, e.g. paths having become obsolete. Traditional process mining techniques like control flow mining and conformance checking can be adapted with relative ease to provide valuable information in this context. For example, conformance checking, i.e. determining the “fit” of the originally defined process schema and the recorded enactment log, can show when a specific alternative of a process schema has never been executed. Consequently, the original pro-cess schema may be simplified by removing that part. Statistical analysis of processenactment can also high-light process definitions, or variants thereof, which have been rarely used in practice. These often clutter the user interface, by providing too many options, and they can become a maintenance burden overtime. Removing (or hiding) them after a human review can significantly improve the efficiency of a process management system.

## 5.CHANGE MINING

In this section we describe novel approaches for analyzing change log information, as found in adaptive PMSs. First, we describe how change logs can be mapped onto the MXML format used for process mining. This makes it possible to evaluate the application of traditional process mining algorithms to change logs. Subsequently, we explore the nature of change logsin more detail. This is followed by an introduction to the

concept of commutativity

### 5.1. Mapping Change Logs to MXML

Change log information can be structured on a number of different levels. Most of all, change events can be grouped by the process definition they address. As we are focusing on changes applied to cases, i.e. executed instances of a process definition, the change events referring to one pro-cess can be further subdivided with respect to the specific case in which they were applied (i.e. into change process instances). Finally, groups of change events on a case level are naturally sorted by the order of their occurrence.

The described structure of change logs fits well into the common organization of enactment logs, with instance traces then corresponding to consecutive changes of a pro-cess schema, in contrast to its execution. Thus, change logs can be mapped to the MXML format with minor modifications. Listing 1 shows an MXML audit trail entry de-scribing the insertion of a task "Lab Test" into a process schema,

<AuditTrailEntry><Data>
<Attribute name="CHANGE.postset">Deliver_report </Attribute>
<Attribute name="CHANGE.type">INSERT </Attribute>
<Attribute name="CHANGE.subject">Lab_test </Attribute>
<Attribute name="CHANGE.rationale">Ensure that blood values are within specs.
</Attribute>
<Attribute name="CHANGE.preset">Examine patient </Attribute>
</Data><WorkflowModelElement>INSERT.Lab_test </WorkflowModelElement><EventType>complete </Event Type><Originator>N.E.Body </Originator>
</AuditTrailEntry>

Listing 1: Example of a change event in MXML.

As discussed in the previous subsection, mapping process change logs to the existing MXML format for execution logs enables the use of existing mining algorithms (e.g., as implemented within the Prom framework) for mining change logs as well. In the following we discuss how "well" these algorithms perform when being applied to change logs. The underlying evaluation has been carried out using an extension of the ADEPT demonstrator. For evaluation purposes, the change processes generated by the different mining algorithms are compared along selected quality criteria. The most important criterion is how "well" a change process reflects the actual dependencies between the operations contained within the input change log. As for process instance I2, for example, change operation op4 depends on previous change operation op3. This dependency should be reflected as a sequence op3 −→ op4 within the resulting change process

### 5.2 Evaluation of Existing Mining Techniques

As discussed in the previous subsection, mapping process change logs to the existing MXML format for execution logs enables the use of existing mining algorithms (e.g., as implemented within the ProM framework) for mining change logs as well. In the following we discuss how "well" these algorithms perform when being applied to change logs. The underlying evaluation has been carried out using an extension of the ADEPT demonstrator. For evaluation purposes, the change processes generated by the different mining algorithms are compared along selected quality criteria. The most important criterion is how"well" a change process reflects the actual dependencies between the operations contained within the input change log. As for process instance I2, for example, change operation op4 depends on previous change operation op3 (cf. Figure 4). This dependency should be reflected as a sequence op3 −→ op4 within the resulting change processing our evaluation we analyzed the α Algorithm, the Multi-Phase Miner, and the Heuristics Miner. All of these algorithms reflect the actual dependencies

Between the change operations quite "well" for simple processes and a restricted set of change operations. The quality of the mined change processes decreases rapidly (i.e., dependencies are generated by the mining algorithms which are actually not existing and the change processes become less and less meaningful) if different change operations are applied and the underlying processes become more complex. The fundamental problem is that process changes tend to be rather infrequent, i.e., compared to regular logs there are relatively few cases to learn from. Therefore, the completeness of change logs, i.e. their property to record independent (i.e. parallel) activities in any possible order, cannot be taken for granted due to their limited availability. This has been simulated by using an

incomplete subset of change logs, as can be expected in a real-life situation. Our experiments with applying existing process mining algorithms to change logs have shown that their suitability in this context is limited. In the following section, we explore the nature of change in an adaptive system and the associated logs in more detail to find a more suitable means for detecting whether an observed ordering relation is actually necessary.

### 5.3 Motivation: Characterization of Change Logs

Change logs, in contrast to regular enactment logs, do not describe the execution of a defined process. This is obvious from the fact that, if the set of potential changes would have been known in advance, then these changes could have already been incorporated in the process schema (making dynamic change obsolete). Thus, change logs must be interpreted as emerging sequences of activities which aretaken from a set of change operations. In Section 5.1 it has been defined that each change operation refers to the original process schema through three associations, namely the subject, pre-set, and post-set of the change. As all these three associations can theoretically be bound to any subset from the original process schema's set of activities1, the set of possible change operations grows exponentially with the number of activities in the original process schema. This situation is fairly different from mining a regular process schema, where the number of activities is usually rather limited (e.g., up to 50–100 activities). Hence, the mining of change processes poses an interesting challenge. Summarizing the above characteristics, we can describe the meta-process of changing a process schema as a highly unstructured process, potentially involving a large number of distinct activities. These properties, when faced by a process mining algorithm, typically lead to overly precise and confusing "spaghetti-like" models. In order to come to a more compact representation of change processes, it is helpful to abstract from a certain subset of ordering relations between change operations. When performing process mining on enactment logs (i.e., the classical application domain of process mining), the state of the mined process is treated like a "black box". This is necessary because enactment logs only indicate transitions in the process, i.e. the execution of activities. However, the information contained in change logs allows to trace the state of the change process, which is in fact defined by the process schema that is subject to change. Moreover, one can compare the effects of different (sequences of) change operations. From that, it becomes possible to explicitly detect whether two consecutive change operations can also be executed in the reverse order without changing the resulting state. The next section introduces the concept of commutativity between change operations, which is used to reduce the number of ordering relations by taking into account the semantic implications of change events.

### 5.4Commutative and Dependent Change Operations

When traditional process mining algorithms are applied to change logs, they often return much unstructured, "spaghetti-like" models of the change process. This problem is due to a large number of ordering relations which do not reflect actual dependencies between change operations. The concept of commutativity is an effective tool for determining, whether there indeed exists a causal relation between two consecutive change operations. As discussed in can be characterized as transforming one process schema into another one. Thus, in order to compare sequences of change operations, and toderive ordering relations between these changes, it is helpful to define an equivalence relation for process schemas.

### 5.5 Approach 1: Enhancing Multi-phase Mining with Commutativity

Mining change processes is to a large degree identical to mining regular processes from enactment logs. Therefore, we have chosen not to develop an entirely new algorithm, but rather to base our first approach on an existing process mining technique. Among the available algorithms, the multi-phase algorithm has been selected, which is very robust in handling ambiguous branching situations (i.e., it can employ the "OR" semantics to split and join nodes, in cases where neither "AND" nor "XOR" are suitable). Although we illustrate our approach using a particular algorithm, it is important to note that any process mining algorithm based on explicitly detecting causalities can be extended in this way (e.g., also the different variants of the α-algorithm). The multi-phase mining algorithm is able to construct basic workflow graphs, Petri nets, and EPC models from the causality relations derived from the log. For an in-depth description of this algorithm, the reader is referred. The basic idea of the multiphase mining algorithm is to discover the process schema in two steps.

First a model is generated for each individual process instance. Since there are no choices in a single instance, the model only needs to capture causal dependencies. Using causality relations derived from observed execution orders and the commutativity of specific change operations, it is relatively easy to construct such instance models. In the second step these instance models are aggregated to obtain an overall model for the entire set of change logs. The causal relations for the multi-phase algorithm are derived from the change log as follows. If a change operation A is followed by another change B in at least one process instance, and no instance contains B followed by A, the algorithm assumes a possible causal relation from A to B. In the example log introduced in instance a change operation deleting "Inform Patient" followed by another change, inserting the same activity again. As no other instance contains these changes in reverse order, a causal relation is established between them. a Petri net model of the change process mined from the example change log instances in The detected causal relation between deleting and inserting "Inform patient" is shown as a directed link between these activities. Note that in order to give the change process explicit start and end points, respective artificial activities have been added. Although the model contains only seven activities, up to three of them can be executed concurrently. Note further that the process is very flexible, i.e. all activities can potentially beskipped. From the very small data basis given in where change log instances hardly have common sub sequences, this model delivers a high degree of abstraction. When two change operations are found to appear in both orders in the log, it is assumed that they can be executed in any order. An example for this is inserting "x-ray" and inserting "Lab Test", which appear in this order in instance I8, and in reverse order in instance I9. As a result, there is no causal relation, and thus no direct link between these change operations in the model shown i Apart from observed concurrency, as described above, we can introduce the concept of commutativity-induced concurrency, using the notion of commutativity introduced in the previous subsection From the set of observed causal relations, we can exclude causal relations between change operations that are commutative. For example, instance I2 features deleting activity "xRay" directly followed by deleting "Inform Patient". As no other process instance contains these change operations in reverse order, a regular process mining algorithm would establish a causal relation between them. However, it is obvious that it makes no difference in which order two activities are removed from a process schema. As the resulting process schemas are identical, these two changes are commutative. Thus, we can safely discard a causal relation between deleting "xRay" and deleting "Inform Patient", which is why there is no link in the resulting change process shown in Commutativity-induced concurrency removes unnecessary causal relations, i.e. those causal relations that do not reflect actual dependencies between change operations. Extending the multi-phase mining algorithm with this concept significantly improves the clarity and quality of the mined change process. If it were not for commutativityinduced concurrency, every two change operations would need to be observed in both orders to find them concurrent. This is especially significant in the context of change logs, since one can expect changes to a process schema to happen far less frequently than the actual execution of the schema, resulting in less log data.

### 5.6 Approach 2: Mining Change Processes with Regions

The second approach towards mining change logs uses an approach based on the theory of regions and exploits the fact that a sequence of changes defines a state, i.e., the application of a sequence of changes to some initial process schema results in another process schema. The observation that a sequence of changes uniquely defines a state and the assumption that changes are "memory less" (i.e., the process schema resulting after the change is assumed to capture all relevant information) are used to build a transition system. Using the theory or regions, this transition system can be mapped onto a process model (e.g., a Petri net) describing the change process. In Definition 2 we already used the concept of a transition system to describe the behavioural aspect of a process schema. However, now we use it as an intermediate format for representing change processes. As indicated in we do not advocate transition systems as an end-user language. Any modelling language having formal semantics can be mapped onto a transition system. The reverse is less obvious, but quite essential for our approach. Therefore, we first explain the "theory of regions" which allows us to translate a transition system into a graphical process model.

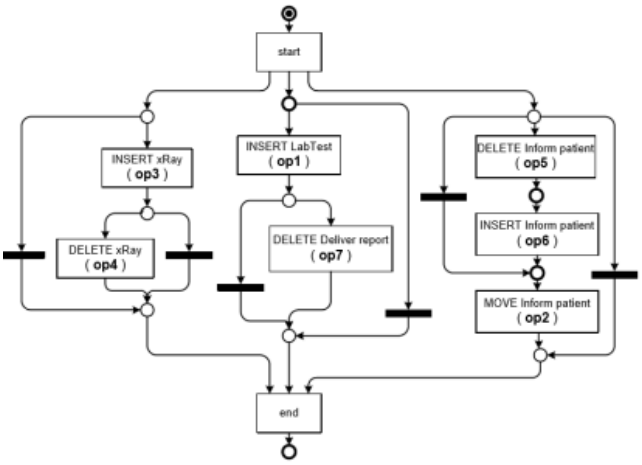


**Fig1:** Mined Example Process (Petri net notation)

### 5.7 Comparing Both Approaches

We have introduced two new process mining approaches based on the characteristics of change logs. The first approach is based on the multi-phase algorithm However, the original algorithm has been enhanced to exploit information about commutativity of change operations. If there are independent changes (i.e., changes that operate on different parts of the schema), it is not necessary to see all permutations to conclude that they are in parallel. The second approach is based on the observation that given an original schema and a sequence of change operations, it is possible to reconstruct the resulting process schema. This can be used to derive a transition system where the states are represented by possible (intermediate) process schemas. Using regions such a transition model can be translated into an equivalent Petri net describing the change process. In this section, we applied the two approaches to an example log. This allows us to compare both. The Petri net in Figure 2 is very different from the one in Figure 1. This illustrates that both approaches produce different results, i.e., they provide two fundamentally different ways of looking at change processes. It seems that in this particular example, the first approach performs better than the second. This seems to be a direct consequence of the small amount of change log instances (just nine) in comparison with the possible number of change operations. When there is an abundance of change log instances, the second approach performs better because it more precisely captures the observed sequences of changes. Moreover, the second step could be enhanced by generalization operations at the transition system level, e.g., using commutativity.

### 5.8 Towards Learning about the Context of Change

Understanding how process change information can be represented in logs and how these logs can be mined to deliver valuable insights into the scope of change delivers insights of how processes deviate from predefined routines. This is a significant move towards understanding why such changes occur, viz., the drivers for change). These drivers can be found in the context of a process. In general terms, the context of a business process is made up by all the relevant information that is available at some stage during the execution of a business process, and that could potentially have influenced decisions in this process. It can be seen as the set of process data and information that is relevant to the process execution but typically not defined in the process definition itself, which, following existing classification schemes would at least include the control flow logic, involved informational data, and organizational resources. Context information can be retrieved from a wide range of potential data sources. Enactment logs, for instance, often include information about time and value of a data modprocess could be investigated together with the reasons for the change decisions taken along the execution of a process. This can be achieved by looking at change process models and the decision points contained within. However, while these change process models themselves are already helpful in developing an understanding of the drivers for change, they cannot be used to actually learn from the change. Learning can be interpreting as deriving information from an adaptive PMS. The fundamental premise is that cases in which a certain change has been applied will exhibit distinct patterns in their context information. As the set of potential context information can be very large, identifying the pivotal data elements, or patterns thereof, which are unique for a specific change, somehow resembles looking for a needle in a haystack. Fortunately, existing Machine Learning (ML) techniques can solve this problem. Classification algorithms, for instance, take for input a classified set of examples, the so-called training set. Once this set has been analyzed, the algorithm is capable of classifying previously unknown examples. Training a decision tree algorithm with such a classified set may then provide decision trees that visualize how decisions about process change were being made. Other classification algorithms from ML can generate a set of classification rules.These classification algorithms by definition focus on specific decisions, i.e., one branching point in the process, and are thus dependent on the mining of a change process model in the first place.An alternative to this approach is the mining of association rules. Here, every case is regarded

as a set of facts, where a fact can both be the occurrence of a change operation as well as a context attribute having a specific value. After identifying frequent item sets, the algorithm can derive association rules. These rules describe, for instance, that for a large fraction of cases where an additional x-ray was inserted, the patient was older than 65 years and the doctor was female, an additional blood screening was inserted.

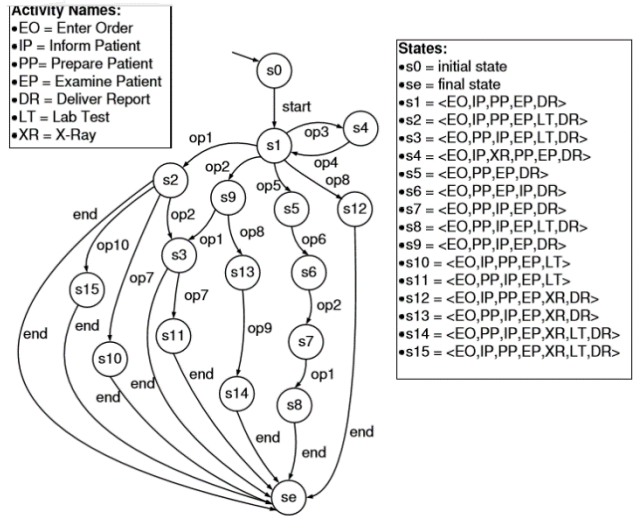


Figure2: Transition system based on the change log shown

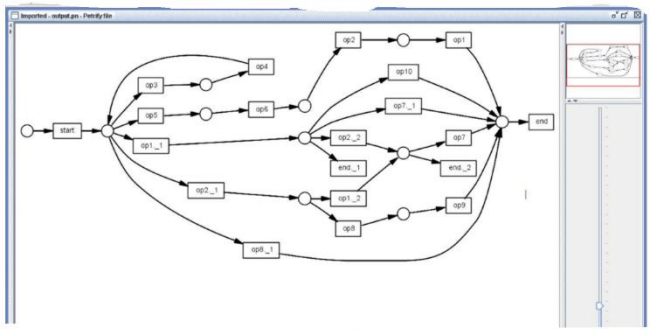

**Fig 3:** *Screenshot of ProM showing the Petri net obtained for the change log depicted*

Association rules are derived in a global manner, viz., the order in which change operations occur is not taken into account. This can be beneficial especially when there are hardly any causal relations between change operations. Association rules may discover tacit relationships between change operations and context data that could not be captured by classification. In summation, the application of ML techniques appears promising for the identification of the drivers for change from the context of a process, and for relating them to one another. We believe that this structured approach can deliver precise results while still remaining feasible in practical settings, and can thus a be foundation for the future design of self-adapting PMSs.

## 6. RELATED WORK

Although process mining techniques have been intensively studied in recent years Agrawal no systematic research on analyzing process change logs has been conducted so far. Existing approaches mainly deal with the discovery of process schemas from execution logs, conformance testing, and log-based verification. The theory of regionshas also been exploited to mine process schemas from execution logs, e.g. from logs describing software development processes. However, execution logs in traditional PMSs only reflect what has been modeled before, but do not capture information about process changes. While earlier work on process mining has mainly focused on issues related to control flow mining, recent work additionally uses event-based data for mining model perspectives other than control flow (e.g., social networks, actor assignments, and decision mining. In recent years, several approaches for adaptive process management have emerged, most of them supporting changes of certain process aspects and changes at different levels. Examples of adaptive PMSs include. Though these PMSs provide more meaningful process logs when compared to traditional workflow systems, so far, only little work has been done on fundamental questionslike.

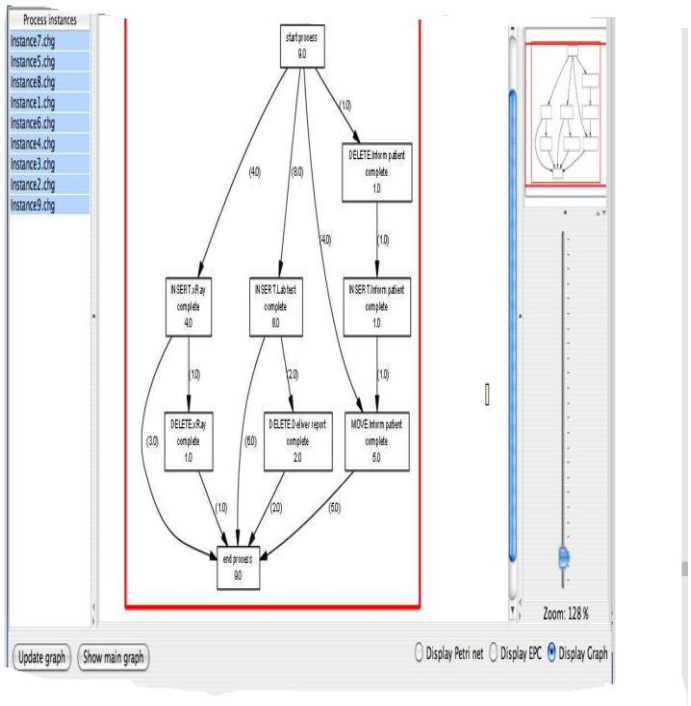

**Fig:**Change Mining Plug-in within ProM

We can learn from this additional log information, how we can utilize change logs, and how we can derive optimized process schemas from them.

## 7. REFERENCES

1. M. Hammer, Beyond Reengineering: How the Process-Centered Organization is Changing Our Work and Our Lives. New York, NY, USA: Harper business, 1996
2. K. Ploesser, J. C. Recker, and M. Rosemann, "Towards a classification and lifecycle of business process change," in Proc. BPMDS, vol. 8. 2008, pp. 1–9.
3. H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining. 2003, pp. 226–235.
4.I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, Mar. 2003.

5.W. M. P. van der Aalst, M. Rosemann, and M. Dumas, "Deadline-based escalation in process-aware information systems," Decision Support Syst., vol. 43, no. 2, pp. 492–511, 2011

6.J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," J. Mach. Learn. Res., vol. 8, pp. 2755–2790, Jan. 2007.

## 8. CONCLUSION&FUTURE WORK:

Thispapergave anoverview ofhowcomprehensive support fortrueprocessflexibilitycanbeprovidedbycombiningadaptivePMSswithadvancedprocessminingtechniques. Theintegrationofprocessminingwithadaptive PMSenables theexploitationofknowledge about process changes from change logs. We have developed two mining techniques and implemented them as plug-ins for the ProM framework, taking ADEPT change logs in the mapped MXML format as in-put. Based on this we have sketched how to discover a (minimal) change process which captures all modifications applied to a particular process. This discovery is based on the analysis of (temporal) dependencies between change operations that have been applied to a process instance. Meaningful, compact representations of the change process.

## BIOGRAPHY:

P.M. Suresh received Bachelor's degree in Computer Science from PDCE, College, India. He is currently working towards Master's degree at Audisankara College of Engineering. His research interest's includes Datamining.

Mr **Koteswarrao. Kadiventi**, M. Tech Assistant Professor, Department of Computer Science, Audisankara College of Engineering Gudur.Received B. Tech in jntuAnantapur affiliated to 2010. Received M. Tech degree in Audisankara College of Engineering Gudur 2012 having 3years of teaching experience. Interest in areas are Data mining.