Knowledge-intensive Processes: An Overview of Contemporary Approaches
Claudio Di Ciccio, Andrea Marrella and Alessandro Russo

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

Claudio Di Ciccio (cdc@dis.uniroma1.it)
1st International Workshop on Knowledge-intensive Business Processes (KiBP 2012)
Friday, June the 15th, Rome, Italy
Business processes

“Degree of structure” in business processes [19]

- **Structured**: Fully predictable
- **Structured with ad hoc exceptions**: Subject to changes in business rules
- **Unstructured with pre-defined fragments**: It can not be modeled as a whole
- **Unstructured**: It can not be modeled as a whole
The structured process

The classical ("imperative") model

- Represents the whole process at once
- The most used notation is based on a subclass of Petri Nets (namely, the Workflow Nets) [53]
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes
Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes
Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes
Workflow Nets (WfNs)
Modeling structured processes
Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)
Modeling structured processes
Workflow Nets (WfNs)
Modeling structured processes
Workflow Nets (WfNs)
Modeling structured processes

Workflow Nets (WfNs)

AND-split

XOR-split

AND-join

XOR-join
Process Mining

Definition

- Process Mining [54], also referred to as Workflow Mining, is the set of techniques that allow the extraction of process descriptions, stemming from a set of recorded real executions (logs).
- ProM [55] is one of the most used plug-in based software environment for implementing workflow mining (and more) techniques.
  - The new version 6.0 is available for download at www.processmining.org
Process Mining

Definition

- Process Mining involves:
  - Process discovery
    - Control flow mining, organizational mining, decision mining;
  - Conformance checking
  - Operational support
- We will focus on the control flow mining
- Many control flow mining algorithms proposed
  - $\alpha$ [AalstEtAl2004] and $\alpha^{++}$ [WenEtAl2007]
  - Fuzzy [GüntherAalst2007]
  - Heuristic [WeijtersEtAl2001]
  - Genetic [MedeirosEtAl2007]
  - Two-step [AalstEtAl2010]
  - ...
A real discovered process model

“Spaghetti process” [54]
Knowledge-intensive processes

- Require the intervention of skilled and knowledgeable personnel.
- Staff acquire their knowledge through their experience of working on similar cases and through collaboration with more experienced colleagues.
- These staff have to deal with issues that can be ambiguous and uncertain and that require judgment and creativity.
- Managing knowledge so it stays within the organization and is passed quickly to new members of staff is a challenge.
The General Care Process

Structured

Structured with ad hoc exceptions

Unstructured with pre-defined fragments

Unstructured

General Care Process

Patient Registration

Assessment

Diagnosis and Investigation

Treatment Plan Definition

Care Plan

Treatment Delivery

Treatment Actions

Treatment Review

Case File Closure

Patient Discharge

Feedback

Case Data
Healthcare Processes [31,46]
From structured to knowledge-intensive processes

- **Organizational and Administrative Processes**
  - patient admission/transfer/discharge procedures, lab tests scheduling, etc.
  - structured, stable and repetitive processes, reflecting routine work with low flexibility requirements
    - possible options and decisions (alternative paths) that can be made during process enactment are statically pre-defined at design time
    - possible exceptions and deviations that can be encountered are predictable and defined in advance, along with the specific handling logic
  - typical setting for the adoption of **procedural process/activity-centric** approaches for process modelling, automation and improvement
    - explicit design-time definition of tasks, execution constraints, participants, roles and input/output data (control-flow + resources + data perspectives)

- **Diagnosis and Treatment Processes**
  - loosely structured or semi-structured processes, with high degree of flexibility
  - no predefined models can be specified, and little automation can be provided
    - focus on decision support
  - **knowledge-intensive processes**
Healthcare Processes

The knowledge-intensive nature of medical processes

- Medical processes reflect knowledge work, decision making and collaboration/coordination activities performed in a healthcare setting [3, 37]

- **Clinical decision** making is highly **knowledge-driven**, as it depends on
  - medical knowledge and evidence
  - case- and patient-specific data (including patient’s past medical history)
  - clinicians’ expertise and experience

- **Patient case** management is the result of **knowledge work**
  - clinicians react to **events** and **changes** in the clinical context on a **per-case basis**
  - decisions and actions are driven by diagnostic-therapeutic cycles [31]
    - interleaving between observation, reasoning and action

- **Patient state** represents the **shared knowledge** that
  - drives the clinical decision making
  - evolves as a result of performed actions, made decisions and collected data
  - enables the definition of eligibility criteria and preconditions for the enactment of specific actions and (sub)procedures
Healthcare Processes

The goal-driven nature of medical processes

- The activities and their execution order in the actual care plan can not always be predetermined
  - continuous interleaving and overlapping of process modeling and execution
  - possibility to define templates and collections of pre-defined activities and process fragments to be composed and instantiated
- The care delivery process evolves through a series of intermediate goals or milestones to be achieved
  - goals are gradually defined, depending on case unfolding, acquired knowledge and previously achieved (or missed) goals
  - changes in patient state and clinical environment may modify/invalidate goals
  - actual diagnostic/therapeutic steps to achieve goals are influenced by declarative knowledge representing domain- (e.g., drug interactions) or site-specific (e.g., availability of resources, lab tests or instruments) constraints
- Clinical processes as continuous goal-driven knowledge acquisition processes
  - actions/decisions produce knowledge
  - knowledge supports subsequent actions/decisions and drives goal definition
The Role of Clinical Guidelines (CGs)

A combination of procedural and declarative knowledge

- **CGs**: systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances

  - goals: standardize clinical procedures, improve care quality, reduce costs and medical errors

- CGs capture medical evidence stemming from statistical knowledge and clinical trials

  - provide generic care processes and recommendations for abstract classes of patients
  - patients, physicians and execution context are “idealized”

- CGs are NOT prescriptive processes

  - act as blueprints/templates that provide evidence-based decision support
  - need to be adapted and personalized to obtain concrete medical pathways

- Evidence-based and procedural knowledge complemented by additional knowledge layers

  - clinicians’ basic medical knowledge
  - site-specific constraints
  - patient-related information
The Role of Clinical Guidelines (CGs)
A combination of procedural and declarative knowledge
Several computer-interpretable languages (and execution environments [27]) have been proposed for modelling and executing CIGs (e.g., ProForma, GLARE, Guide [42, 61])

- **task-based paradigms**: modelling primitives for representing actions, decisions and patient states, linked via scheduling constraints
  - rigid flow-chart-like structure
- **process/activity centric approach**
  - capture procedural knowledge in CIGs
  - focus on control-flow dimension

**Limited uptake in practice**

- lack of flexibility in presence of deviations, exceptions and events
  - efforts required to continuously tailor/adapt models to specific medical settings and changing conditions

**Recent convergence** between CG and BPM research communities
Representing and Executing Clinical Guidelines
Basic components interaction

- Patient-related Data (actual + historical)
- Site-specific constraints and policies
- Clinical Data
- Procedural and declarative knowledge
- Clinical Guidelines
- Basic Medical Knowledge
- Goals
- Actions
- Decisions
- Actual Medical Plan

Events and changes
Can “classical” BPM Support Clinical Processes?

- **Process/Activity Centric Models**
  - Clinical procedures can not be completely specified in advance nor fully automated
    - High variability and flexibility requirements make modelling effort useless
  - Procedural process definitions may unnecessarily limit possible execution behaviours
    - Over-specified or over-constrained models
    - Little acceptance by clinicians
  - Limited support for handling deviations and uncertainty
  - Actions and decisions do not directly depend on scheduling and completion of other activities

- **Declarative Constraint-based Models [32, 40]**
  - Increase flexibility wrt possible execution behaviours
    - Specification of a (minimal) set of constraints to be satisfied, defined as relationships among tasks
    - No rigid control-flow structure
  - Focus is still on tasks/activities
    - Limited support for data-oriented modelling

Limitation of existing approaches: missing integration between processes, data and knowledge.
Object-Aware and Artifact Centric Models \[5,51\]

Towards Patient-Centric Adaptive Case Management

- Clinical process support requires **object-awareness** \[5\]
  - full integration of processes with data models consisting of object types and object relations
  - data as first-class citizens

- Rich data and information model
  - explicit representation of domain-relevant objects/artifacts (patient, medical orders, lab reports, etc), their attributes and inter-relations
  - characterization of objects/artifacts evolution and behaviour in terms of *lifecycles*

- Data- and Event-driven modelling and execution
  - data models enables the definition of activities
  - activities enabled by triggering events, constrained by conditions over data attributes/states (e.g., ECA-like rules)
  - executed activities produce changes on attribute values, object/artifact relations and states

- Explicit representation of goals
  - goal achievement induced by event occurrence and changes in the information model
"If you want a second opinion, I'll ask my computer."
Business Process Adaptation

- **Process Adaptation** represents the ability of the implemented processes to cope with *exceptional circumstances* and to deviate at run-time from the execution path prescribed by the process.

- Existing PMSs provide support for the handling of:
  - *expected exceptions*, which can be anticipated and thus be captured in the process model [50].
  - *unanticipated exceptions*, which are usually addressed through structural ad-hoc changes of single process instances [65].
Dynamic Processes

A subclass of KiBP’s

- We call **dynamic process** the workflow where the sequence of tasks depends upon the specifics of the context
  - for example, which resources are available and what particular options exist at that time
  - it is often unpredictable the way in how it unfolds.
- This is due to either
  - the **high number of tasks** to be represented,
  - their **unpredictable nature**, or
  - a **difficulty to model the whole knowledge** of the domain of interest at design time.
- **Processes for Emergency Management**: new situations coming from the environment might be such that the PMS is no more able to carry out the process instance.
Adaptation of Dynamic Processes

Off-line adaptation

- In general, for a dynamic process there is not a clear, anticipated correlation between a change in the context and corresponding process changes, since
  1. the process may be **different every time it runs** and
  2. the **recovery procedure strictly depends on the actual contextual information**

- In collaborative and real-life scenarios, a PMS should provide
  - intelligent failure handling mechanisms and
  - enriched process models.

- The use of **AI techniques** seems very promising in this direction.

- **Off-line adaptation** through planning [23,45,22] and learning techniques [20] allows to build on-the-fly the recovery procedure to deal with a specific exception.
  - During the process execution, when an exception occurs, a new repair plan is generated by taking into account constraints posed by the process structure and by applying or deleting actions taken from a **given generic repair plan**, defined manually at design time or inferred from past executions.
Adaptation of Dynamic Processes

Run-time adaptation

- Some recent approaches on process adaptation allow to synthesize a recovery procedure **without defining** at design time **any recovery policy**.
- In [34], a **run-time automatic synthesis** of the recovery policy is devised by **integrating planning techniques** on top of a PMS.
  - Each task is described in terms of its preconditions and effects, and can be considered as a single step that consumes input data and produces output data.
  - Process Adaptivity in [34] is the ability of the PMS to reduce the gap from the expected reality $\psi(s)$ – the (idealized) model of reality that is used by the PMS to reason – and the physical reality $\phi(s)$ – the real world with the actual values of conditions and outcomes.

\[ \begin{align*}
\phi(s) & \xrightarrow{\text{Pre}_i} t_i \xrightarrow{\text{Eff}_i} \phi(s+1) \\
\psi(s+1) & = \phi(s+1)
\end{align*} \]

*for each execution step*

*if* $\phi(s+1)$ *is different from* $\psi(s+1)$

*then* adapt

---

**Physical reality** at situation $s$. A situation is an history of actions occurred so far.

**Expected reality** is changed as the effects of the task are the desired ones.

**Each task has a set of effects that turn the “old” physical reality $\Phi(s)$ into $\Phi(s+1)$.**
Run-time Adaptation in [34]

- If a **discrepancy** between the two realities is sensed:
  - a **planning problem** is built, by taking $\phi(s)$ as the initial state, $\psi(s)$ as the goal and the set of task definitions as the planning domain;
  - a planner is invoked by giving as input the planning problem just defined;
  - the aim is to find a recovery procedure that turns $\phi(s)$ (the initial state) into $\psi(s)$ (the desired expected state).

- This general framework is based on execution monitoring [70] formally represented in Situation Calculus [48] and IndiGolog [12].

The adaptation works by synthesizing a linear process $h$ (constituted by a sequence of actions) which can recover the situation.

The different concurrently running branches are all interrupted both during the planning stage and during the execution of the recovery procedure.
Other works on run-time adaptation /1

- In [71], the authors proposed a technique (based on Situation Calculus, ConGolog [72] and regression planning) for adapting processes without having to stop the concurrently running branches.
- When an exogenous action breaks one of the concurrently running branches, only the branch involved in the exception has to be blocked. The recovery plan is computed in concurrency with the remaining part of the process to be executed.
- Once the recovery plan has been synthesized, its execution will involve the branch affected by the deviation.

**Strong assumption:**
the technique works if and only if the concurrently running branches are all independent.
Other works on run-time adaptation /2

- In [34], the authors propose a technique based on Continuous Planning algorithms for adapting processes without having to stop directly any task in the process.
- The continuous planner works with a partial-order planning algorithm.
- The technique works under the assumption that if some exception arises (and it is reflected in a discrepancy between the two realities), it means that some task preconditions do not hold, by preventing the task execution.
- The continuous planner search for a recovery plan in concurrency with the execution of the main process.
- The changes in the two realities are directly reflected into the plan under construction.
The Continuous Planning Algorithm [34]

\[ \varphi(s+1) = \text{Physical Reality at situation } s+1 \]

\[ \psi(s+1) = \text{Expected Reality at situation } s+1 \]

If a task ends its execution then
1. stop the planner execution
2. take the partial plan built so far
3. update the two realities
4. remove conflicts and make the partial plan appropriate with the new initial state and goal
5. the planner can resume its execution by starting with the revised partial plan

During the planning stage, the main process can carry on with its execution.

The plan under construction has to be synchronized with the current process execution.
Artful processes

What are artful processes?

- **Artful processes** [26]
  - informal processes typically carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.)
  - “knowledge workers” [63]
- Knowledge workers create artful processes “on the fly”
- Though artful processes are frequently repeated, they are not exactly reproducible, even by their originators, nor can they be easily shared.
On the visualization of processes

The declarative model

• Rather than using a **procedural language** for expressing the allowed sequence of activities, it is based on the description of workflows through the usage of **constraints**

  • the idea is that every task can be performed, except the ones which do not respect such constraints
  • this technique fits with processes that are highly flexible and subject to changes, such as artful processes

The notation here is based on [57,33] (*DecSerFlow, Declare*)
On the visualization of processes

Imperative vS declarative

Imperative models work better in presence of a partial specification of the process scheme

Declarative models work better in presence of a partial specification of the process scheme
Memento!
Declare Worklist

Knowledge-intensive Processes: An Overview of Contemporary Approaches
On the visualization of processes
An example of DecSerFlow [57] notation

No, it is not the initial action

You could even start from here

You might want to run a legal trace like this:

- \( \langle a_3, a_3, a_3, a_2, a_2, a_3, a_4, a_5, a_6, a_7, a_6, a_5 \rangle \)

What we want to state here is that such a notation is probably not quite intuitive
Dynamic Condition-Response Graphs (DCR Graphs)

The runtime state graphical notation [25]
On the visualization of processes

Introducing the new local view: the rationale

4th quadrant:
impllying or legal tasks, before

The analyzed task is
put on the origin

No matter when,
"u" must be done
if you want to do "t"

A bold box means:
zero or one time

A dashed box means:
zero, one or more times

The asterisk stands for
any other activity

Implication (tasks)

2nd quadrant:
impllyed or legal tasks, after

Time (steps)
On the visualization of constraints

The static local view: some examples

ChainResponse(r, t)

AlternateResponse(s, t)

CoExistence(t, u), NotSuccession(t, q)
On the representation of processes
The static global view

Basic

Extended
A GUI sketch

Local and global views together
On the representation of constraints

Dynamic view
Declarative workflow discovery techniques

State of the art

- Chesani et al. [4]
  - describes the usage of inductive logic programming techniques to mine models expressed as a SCIFF theory. SCIFF theory is thus translated into the ConDec notation.

- Bellodi et al. [73]
  - adopts a probabilistic approach on the learned theory.

- Maggi et al. [33]
  - is based on the translation of Declare constraints into automata, where traces are replayed on.

- Di Ciccio, Mecella [74,17]
  - is a two-step algorithm:
    1. It builds a knowledge base of its own
    2. Responses to specific queries on the knowledge base establish whether constraints hold
References

[1-68]
Please refer to the paper:


