Pattern Discovery from Innovation Processes

Claudia Diamantini, Laura Genga, Domenico Potena, Emanuele Storti

DII, Universitá Politecnica delle Marche, Ancona, Italy

CTS 2013, San Diego, CA, May 20-24
Introduction

Innovation process as a goal-oriented set of collaborative activities in the context of an innovation project

Innovation vs. traditional organization processes:

- a special type of *spaghetti* process
- unstructured vs. structured
- more complex
- rapidly changing, dynamic
- more challenging to analyse

Motivation: need of solutions to obtain better understanding of collaborative practices in innovation
Related Work

**Process mining** techniques allow for extracting knowledge from event logs, i.e., traces of running processes [W.D.Aalst, 2011],[Rubin,2007]:

- **process discovery**: what is really going on?
- conformance check: are we doing what was agreed upon?
- process extension: how can we redesign the process?

Main limitation: no underlying schema to extract in innovation processes
Related Work

**Process mining** techniques allow for extracting knowledge from event logs, i.e. traces of running processes [W.D.Aalst, 2011],[Rubin,2007]:

- process discovery: what is really going on?
- **conformance check**: are we doing what was agreed upon?
- process extension: how can we redesign the process?

Main limitation: no underlying schema to extract in innovation processes
Related Work

**Process mining** techniques allow for extracting knowledge from event logs, i.e. traces of running processes [W.D.Aalst, 2011], [Rubin, 2007]:

- process discovery: what is really going on?
- conformance check: are we doing what was agreed upon?
- process extension: how can we redesign the process?

Main limitation: no underlying schema to extract in innovation processes
Related Work

**Process mining** techniques allow for extracting knowledge from event logs, i.e. traces of running processes [W.D.Aalst, 2011],[Rubin, 2007]:

- process discovery: what is really going on?
- conformance check: are we doing what was agreed upon?
- process extension: how can we redesign the process?

Main limitation: no underlying schema to extract in innovation processes
Approach

Discovery of frequent patterns from innovation processes

Goal: understanding common patterns for collaborative activities
- to recognize best/worst practices
- to investigate internal/external pitfalls
- to determine type and impact of collaboration activities
- to understand behaviour of actors involved
- to improve collaboration practices and interfaces
Approach

Hypotheses:

- collaborative activities on innovative project are logged and recorded by a **innovation management system**

- all the activities recorded in the logs are named according to a given **terminology**
  E.g., “skype call conf”, “physical meeting”, “meeting”, “idea suggestion”,...

- all the activities are categorized in **classes** according to a **taxonomy**
Methodology

Innovation patterns discovery

Data collection & preprocessing:
1. collection of data from logging systems, collaborative tools, emails
2. representation of traces into schemas

Pattern discovery
Methodology

Innovation patterns discovery

Data collection & preprocessing:
1. collection of data from logging systems, collaborative tools, emails
2. representation of traces into schemas

Pattern discovery
Methodology

Innovation patterns discovery

Data collection & preprocessing:
1. collection of data from logging systems, collaborative tools, emails
2. representation of traces into schemas

Pattern discovery
From traces to schemas

Main steps:

- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)
Main steps:

- **data cleaning and format adaptation**
- **each trace as a schema**
- **replacement of each activity with its class (class-level schema)**
- **preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)**
From traces to schemas

Main steps:
- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)
From traces to schemas

Main steps:

- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)
From traces to schemas

Main steps:
- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)
From traces to schemas

Main steps:
- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)  Innovation Pattern Discovery  CTS 2013
From traces to schemas

Main steps:
- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- **preprocessing** to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)  Innovation Pattern Discovery  CTS 2013  7 / 23
From traces to schemas

Main steps:
- data cleaning and format adaptation
- each trace as a schema
- replacement of each activity with its class (class-level schema)
- preprocessing to recognize structured representation for sets of schemas (parallel paths, cycles, ...)

Emanuele Storti (UNIVPM)
Innovation Pattern Discovery
CTS 2013 7 / 23
Innovation patterns discovery

Data collection & preprocessing:
1. collection of data from logging systems, collaborative tools, emails
2. representation of traces into schemas

Pattern discovery:
1. translation of schemas into graphs
2. usage of graph-mining techniques to extract patterns
From schemas to graphs

Several alternative models of representation (A, B, C), with different detail levels [Diamantini, 2012].

C model:
- highest level of compactness
- implicit representation of operators
- SPLIT-XOR to model cycles
- arcs are labeled to preserve information about source/target nodes
Methodology

Innovation patterns discovery

Data collection & preprocessing:
1. collection of data from logging systems, collaborative tools, emails
2. representation of traces into schemas

Pattern discovery:
1. translation of schemas into graphs
2. usage of graph-mining techniques to extract patterns
Graph clustering

SUBDUE [Joyner, 2000]

- graph-based hierarchical clustering algorithm
- suited for discrete-valued and structured data
- searches for substructures (i.e., subgraphs) that best compress the input graph, according to Minimum Description Length

- Input: a set of graphs
- Output: the set of discovered subgraphs (hierarchically arranged in a lattice)
Graph clustering (cont’d)

1. searches for the best SUB (i.e., that minimizes the Description Length of the graph)

2. compresses the graph by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a lattice.
Graph clustering (cont’d)

1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)

2. **compresses the graph** by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a **lattice**
Graph clustering (cont’d)

1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)

2. **compresses the graph** by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a **lattice**.
Graph clustering (cont’d)

1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)

2. **compresses the graph** by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a **lattice**
Graph clustering (cont’d)

1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)
2. **compresses the graph** by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a **lattice**.
Graph clustering (cont’d)

1. **searches for the best SUB** (i.e., that minimizes the Description Length of the graph)
2. **compresses the graph** by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a **lattice**
Graph clustering (cont’d)

1. searches for the best SUB (i.e., that minimizes the Description Length of the graph)
2. compresses the graph by using the best SUB

SUBs may be defined in terms of previous SUBs. Iterations of this basic step results in a lattice
Traditional cluster **evaluation** is not applicable to hierarchic domains

Desirable features of a good clustering:

- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

**New indexes**

- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
Traditional cluster **evaluation** is not applicable to hierarchic domains

Desirable features of a good clustering:

- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

**New indexes**

- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
Graph clustering (cont’d)

Traditional cluster evaluation is not applicable to hierarchic domains
Desirable features of a good clustering:

- few and big clusters (large coverage, good generality)
- minimal/no overlap (better defined concepts)

New indexes

- **completeness**: % of original nodes/edges still present in the final lattice
- **representativeness** (of a SUB): % of input graphs holding the SUB at least once
Experimentation

Synthetic datasets from an automatic process generator:

- **taxonomy** of 51 collaborative activities arranged in classes:
  E.g., callForIdeas, meeting, voting, ideaRefinement

- **3 templates representing typical innovation scenarios:**
  (I) team collaborating with external partners for the setup of an innovation project,
  (II) ideas crowd-sourcing and collaborative voting for selection,
  (III) client-push innovation model
Example: Template 1

- scenario: setup of an innovation project where a team collaborates with external partners
- defined at class-level
- various points of variability
Experimental results

Settings:
- 2 experiments (instance/class)
- 200 instance-level processes ($\approx 2000$ tasks)
- 200 class-level processes
- translation into graphs
- SUBDUE execution
- lattice generation
Experimental results

Fragment of generated lattice with discovered patterns
Experimental results

Fragment of generated lattice with discovered patterns
Experimental results

Nodes = 1973, Edges = 2123, 
Avg nodes/graph = 9.89
Beam range = 1:20

<table>
<thead>
<tr>
<th></th>
<th>instance-level</th>
<th>class-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBs (num)</td>
<td>610</td>
<td>101</td>
</tr>
<tr>
<td>top SUBs</td>
<td>18.20%</td>
<td>16.83%</td>
</tr>
<tr>
<td>Completeness</td>
<td>88.50%</td>
<td>98.34%</td>
</tr>
<tr>
<td>Representativeness (first 10 top SUBs)</td>
<td>61.5%</td>
<td>95.5%</td>
</tr>
<tr>
<td>Execution time (s)</td>
<td>244</td>
<td>4</td>
</tr>
</tbody>
</table>

Considerations:
- Instance-level: lower representativeness, higher execution time
- Class-level: compact representation, better patterns
Experimental results

Nodes = 1973, Edges = 2123,
Avg nodes/graph = 9.89
Beam range = 1:20

<table>
<thead>
<tr>
<th></th>
<th>instance-level</th>
<th>class-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBs (num)</td>
<td>610</td>
<td>101</td>
</tr>
<tr>
<td>top SUBs</td>
<td>18.20%</td>
<td>16.83%</td>
</tr>
<tr>
<td>Completeness</td>
<td>88.50%</td>
<td>98.34%</td>
</tr>
<tr>
<td>Representativeness (first 10 top SUBs)</td>
<td>61.5%</td>
<td>95.5%</td>
</tr>
<tr>
<td>Execution time (s)</td>
<td>244</td>
<td>4</td>
</tr>
</tbody>
</table>

Considerations:
- Instance-level: lower representativeness, higher execution time
- Class-level: compact representation, better patterns
Conclusion

Methodology for the discovery of significant and recurrent patterns in innovation processes:

- procedure for translation of traces into schemas and graphs
- application of hierarchical clustering technique SUBDUE
- evaluations on synthetic dataset of innovation processes

Considerations

- patterns vs. schema discovery
- usage of class-instance taxonomy of activities to cluster class-level schemas
- similar substructures can be discovered by SUBDUE
Conclusion

Applications:

- recognition of **frequent patterns** (common/best/worst practices, pitfalls, ...)
- organization of a **process repository** (by indexing processes through substructures)
- **enterprise integration**: finding similarities (differences, overlaps, complementariness) in innovation practices among companies

Future work:

- extending experimentations, especially with real innovation processes (BIVEE project)
- investigating different clustering algorithms
- semantic annotation of tasks
Acknowledgments

This work has been partly funded by the EC through the ICT Project BIVEE: Business Innovation and Virtual Enterprise Environment (FoF-ICT-2011.7.3-285746)
Pattern Discovery from Innovation Processes

Claudia Diamantini, Laura Genga, Domenico Potena, Emanuele Storti

DII, Università Politecnica delle Marche, Ancona, Italy

CTS 2013, San Diego, CA, May 20-24
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. Search the best SUB
   - \textit{Init}: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top \( n \) SUBs
   - \textit{End}: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

**Search the best SUB**
- *Init*: set of SUBs consisting of all uniquely labeled vertices
- extend each SUB in all possible ways by a single edge and a vertex
- order SUBs according to MDL principle
- keep only the top $n$ SUBs
- *End*: exhaustion of search space or user constraints
SUBDUE

Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. Search the best SUB
   - \textit{Init}: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top \( n \) SUBs
   - \textit{End}: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. Search the best SUB
   - **Init**: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - **End**: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints
SUBDUE

Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. Search the best SUB
   - Init: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - End: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints

2. **Compress the graph with the best SUB**
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints

2. **Compress the graph with the best SUB**
Subdue uses a variant of beam search (heuristic)
Best SUB minimizes the Description Length of the graph

1. **Search the best SUB**
   - *Init*: set of SUBs consisting of all uniquely labeled vertices
   - extend each SUB in all possible ways by a single edge and a vertex
   - order SUBs according to MDL principle
   - keep only the top $n$ SUBs
   - *End*: exhaustion of search space or user constraints

2. **Compress the graph with the best SUB**

Subsequent SUBs may be defined in terms of previously defined SUBs

Iterations of this basic step results in a lattice