Using business workflows to improve quality of experiments in distributed systems research

Tomasz Buchert — INRIA Nancy - Grand Est
Lucas Nussbaum — LORIA / Université de Lorraine

1 Problem: unreproducible experiments

Research in distributed systems suffers from many difficulties and technical issues that threaten the quality of results:

- **large-scale**, **complex** and **unreliable** communication networks,
- software and hardware **errors**,
- **non-isolated** experimental environment,
- bad **methodology** and lack of **scientific rigor**,
- **incomplete description** of the experiment.

With the ever-growing complexity of the computer systems, the problem becomes more evident.

2 Goal: make experiments reproducible

The experiment needs a high-level description that:

- is **abstract** and **descriptive**,
- is **modular** and composed of **reusable** parts,
- can be easily **maintained** and **improved**,
- uses **common patterns** instead of reinventing them,
- integrates **analysis** of experimental results,
- handles failures and timeouts **transparently**.

3 Approach: leveraging business workflows

Our method consists in harnessing methods of Business Process Management:

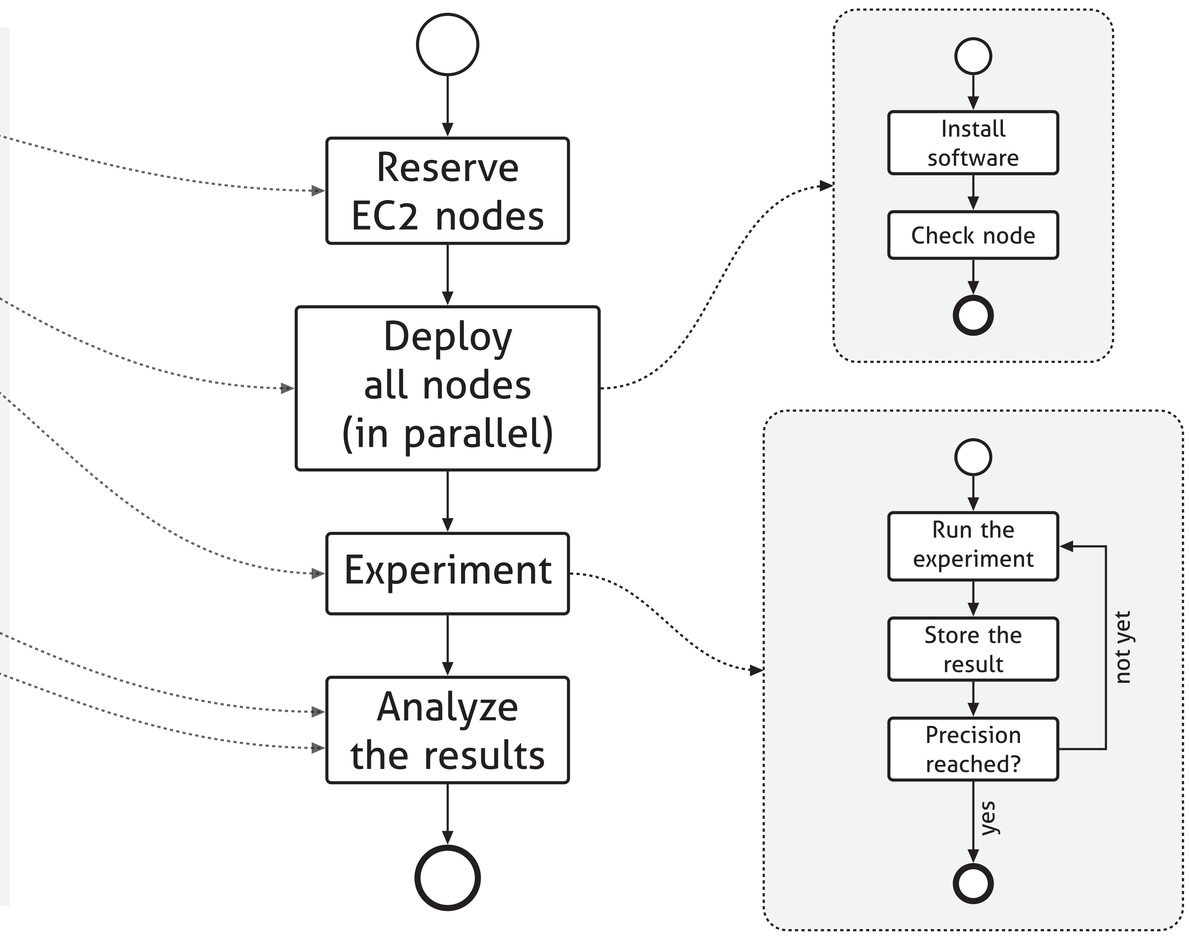
- the experiment is expressed using a **domain specific language** (DSL) using **experiment patterns** and patterns identified in the domain of BPM; the experiment description maps to **workflow representation**,
- the functional parts of the experiment, or **activities**, implement atomic actions that realize a concrete and well-defined goal (e.g., installation of a package),
- finally, an **experiment engine** executes the workflow and takes care of communication, data collection, monitoring, exceptional situations, etc.

4 Benefits: improved quality of the experiment

- Understanding:
  - better structure and **ease of understanding**,
  - unobvious **optimization** opportunities,
  - improved **maintainability** of the experiment.
- Modularity:
  - **reusable** components of the experiment,
  - well-defined and **loosely coupled** components,
  - **logical separation** between layers of software involved in the experiment.
- Monitoring:
  - the state of experiment constantly **monitored**,
  - **automatic recovery** from failures and exceptional situations,
  - timing information used to identify **bottlenecks** and other performance issues.
- Workflow patterns:
  - well-tested, **building blocks** of the experiment,
  - **parallel execution** of activities,
  - deadlocks avoided by **implicit synchronization**.

5 Example: description of an experiment involving Amazon EC2 instances

```
1  engine.process do
2      nodes = run :reserve_nodes    #
3      forall nodes do |n|    #
4          run :install_software, n
5          run :check_node, n
6      end
7      loop do    #
8          result = run :experiment, nodes
9          store :results, result
10         return_on (data :results).precision < 0.1
11     end
12     run :analyze_data    #
13 end
14
15 engine.activity :analyze_data do  #
16     results = data :results
17     value = results.average
18     delta = results.confidence
19     puts "The experiment outcome is #{value}."
20     puts "The confidence interval is #{delta}."
21     results.save "results.txt"
22 end
```



informatiques mathématiques
Loria Laboratoire lorrain de recherche en informatique et ses applications
UNIVERSITÉ DE LORRAINE