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

Tomasz Buchert

advised by:
Lucas Nussbaum and Jens Gustedt

Experimentation

We all know how frustrating experimenting can be.

That's because experiments in distributed systems are:

- time-consuming
- difficult to do correctly
- complex and incomprehensible
- failure-prone

We have to properly **control** the experiments.

Experimentation tools

Many tools to manage experiments exist:

- Naive way (SSH + Bash + ...)
- Expo
- g5k-campaign
- OMF
- Plush
- ... among many others

They are based on different paradigms.

Our main goal

To improve the research, the experimentation framework has to:

- improve **descriptiveness** of the experiments
- give a **modular** way to build experiments
- handle **unexpected**, but **inevitable** errors
- ensure **scalability** of experiments
- ensure **reproducibility** (or at least **repeatability**)

In the end, we want to

improve experimentation on testbeds like Grid'5000 and PlanetLab.

Bottom-up vs top-down approach

The majority of tools mentioned use **bottom-up design**.

What about a **top-down** approach?

1. Start with high-level description of the experiment.
2. Implement low-level details.
3. Run the experiment.
4. Improve if necessary and reiterate.

There already exists an approach like this.

Business Process Management

Business Process Management is about:

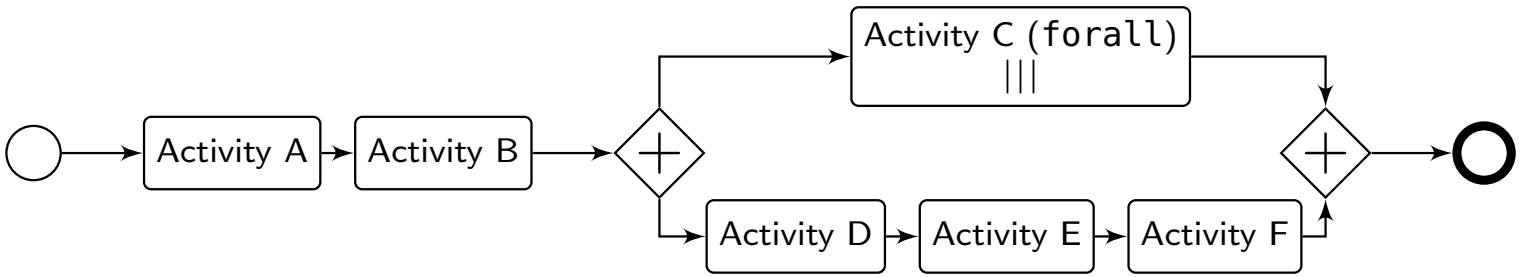
- **understanding** an organization
- modeling its processes as **workflows**
- **executing** processes and **monitoring** them
- **improving** organizational **activities**
- redesigning **processes** to make them:
  - cheaper
  - faster
  - less defective

The goal of my thesis

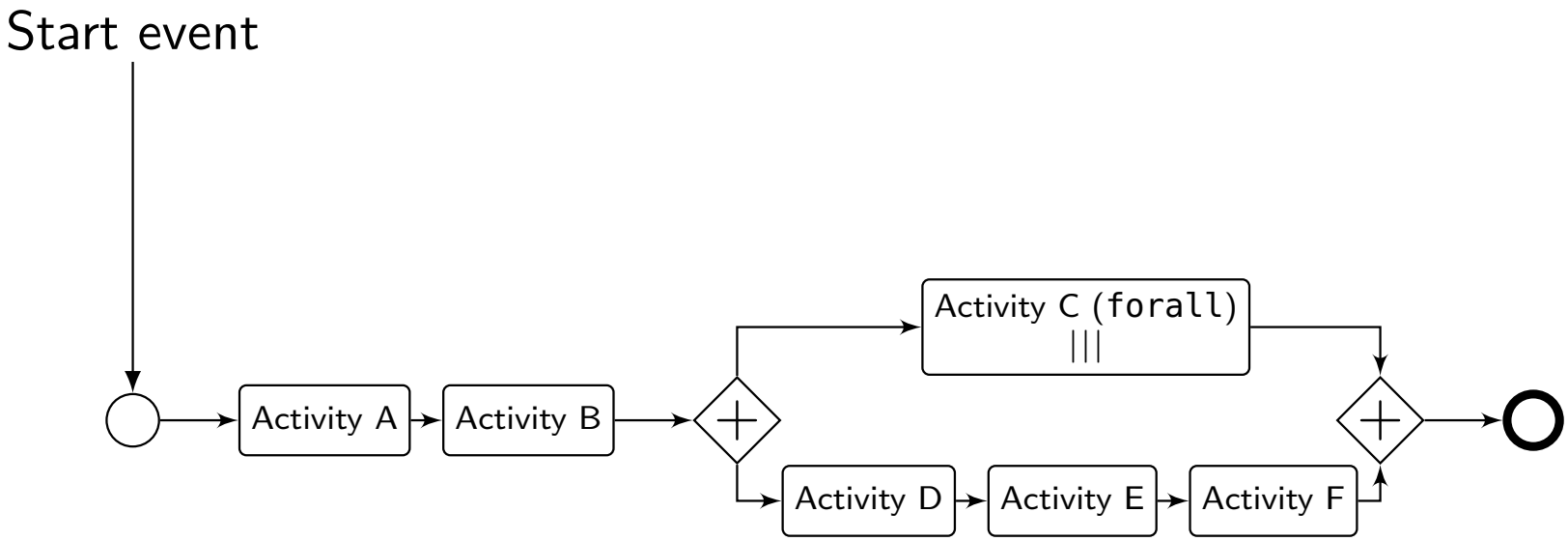
Can BPM improve the experimentation with distributed systems?

(and if the answer is "yes", then how?)

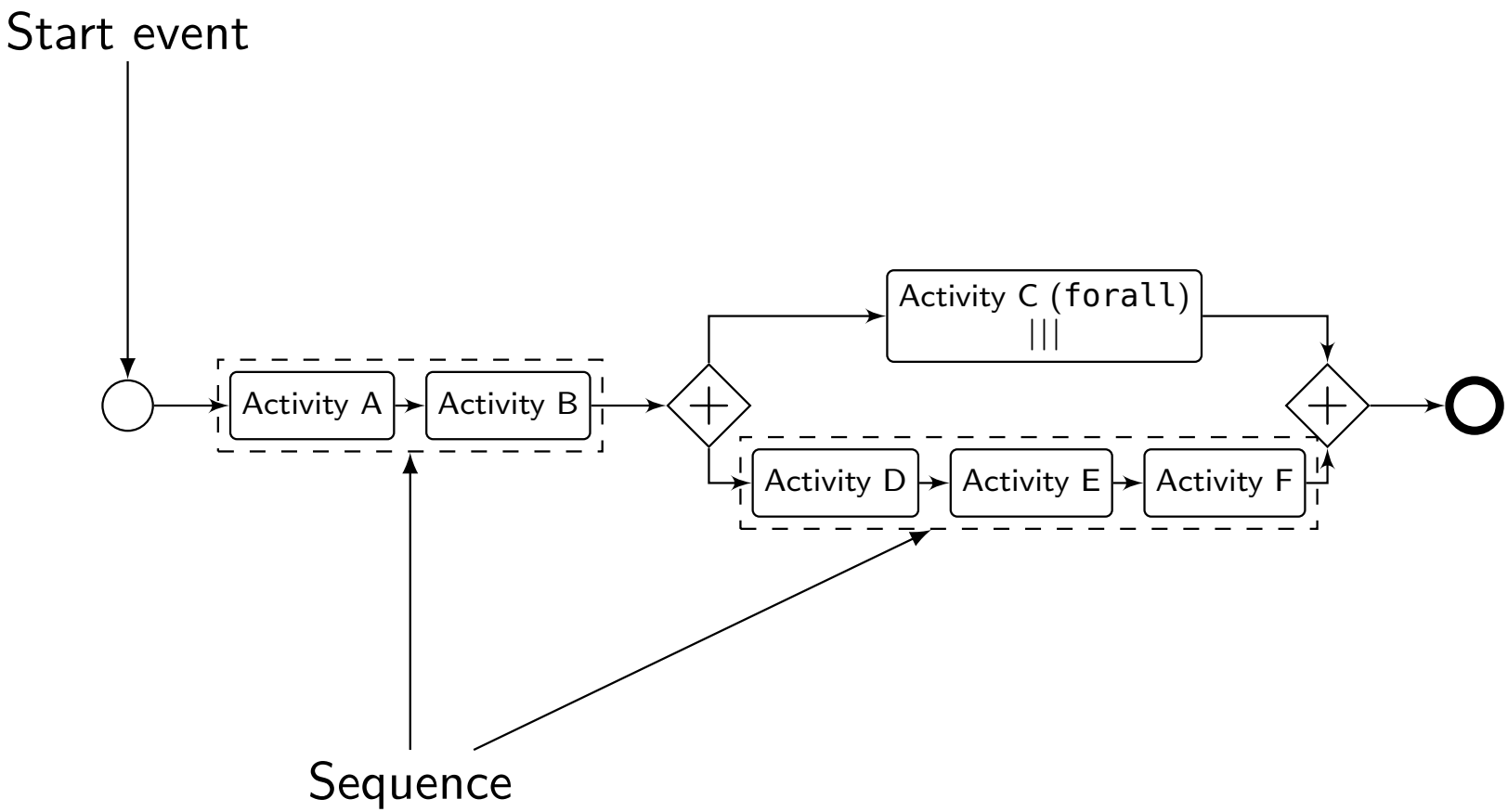
Crash course in workflows



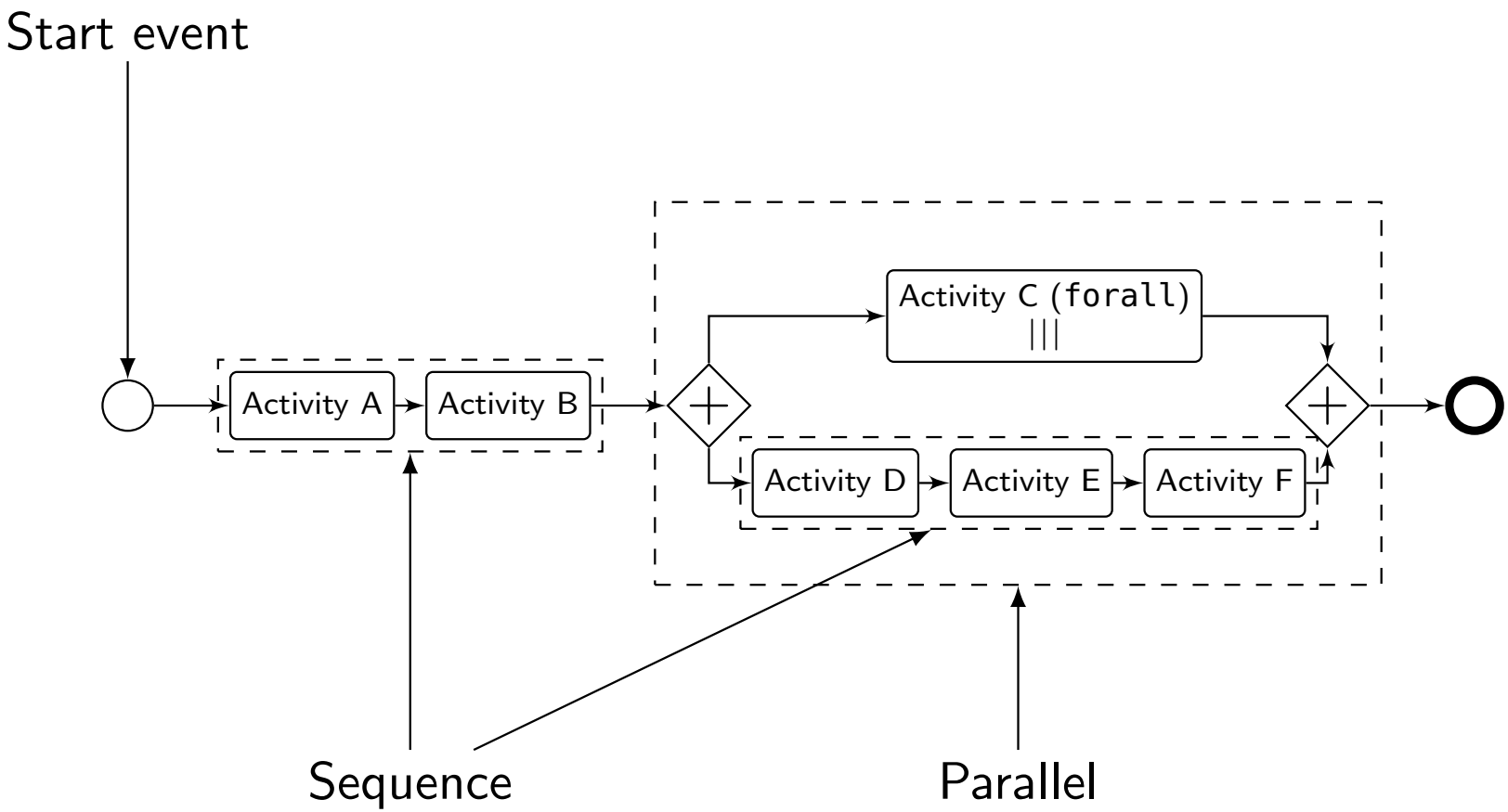
Crash course in workflows



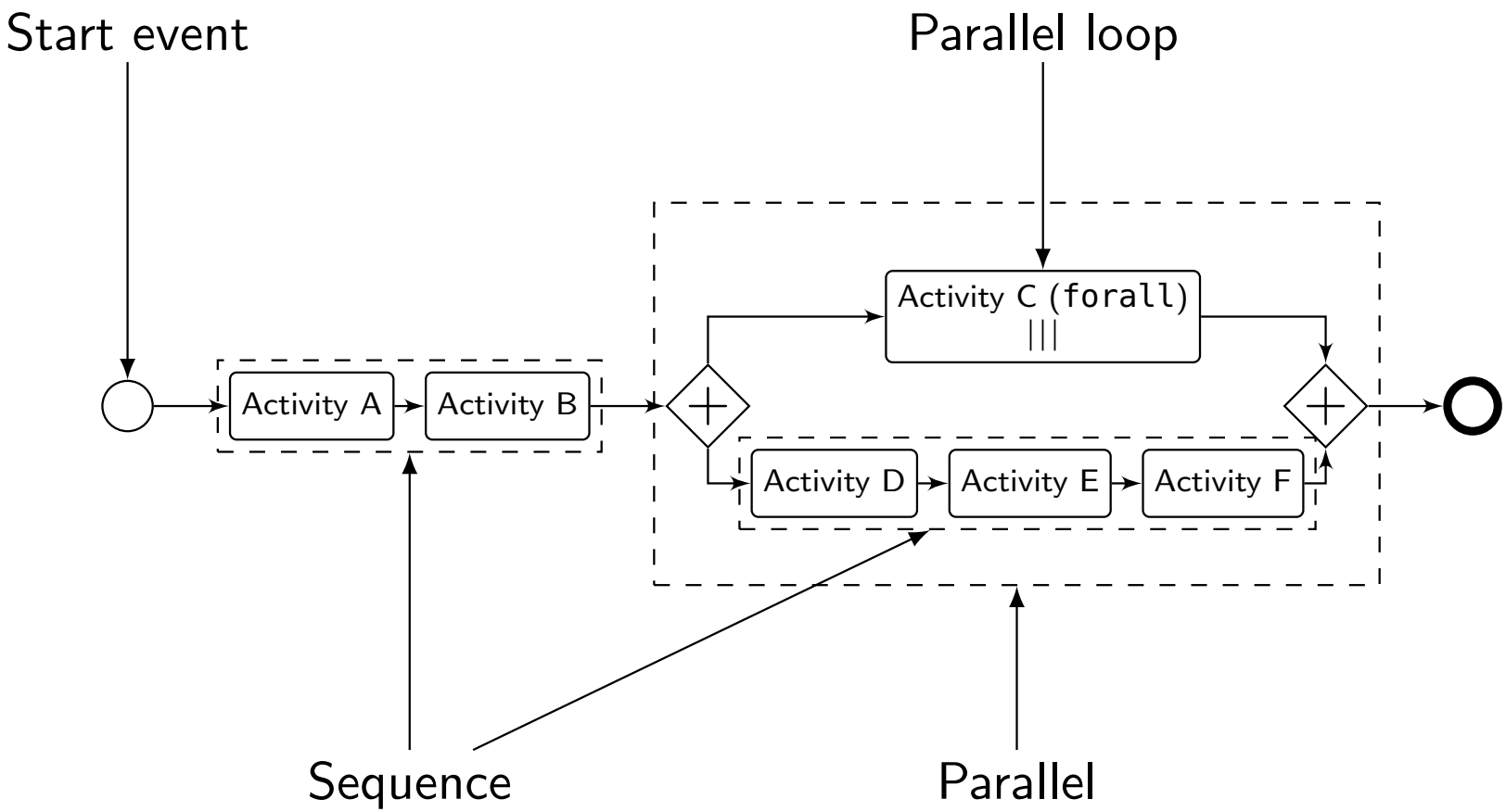
Crash course in workflows



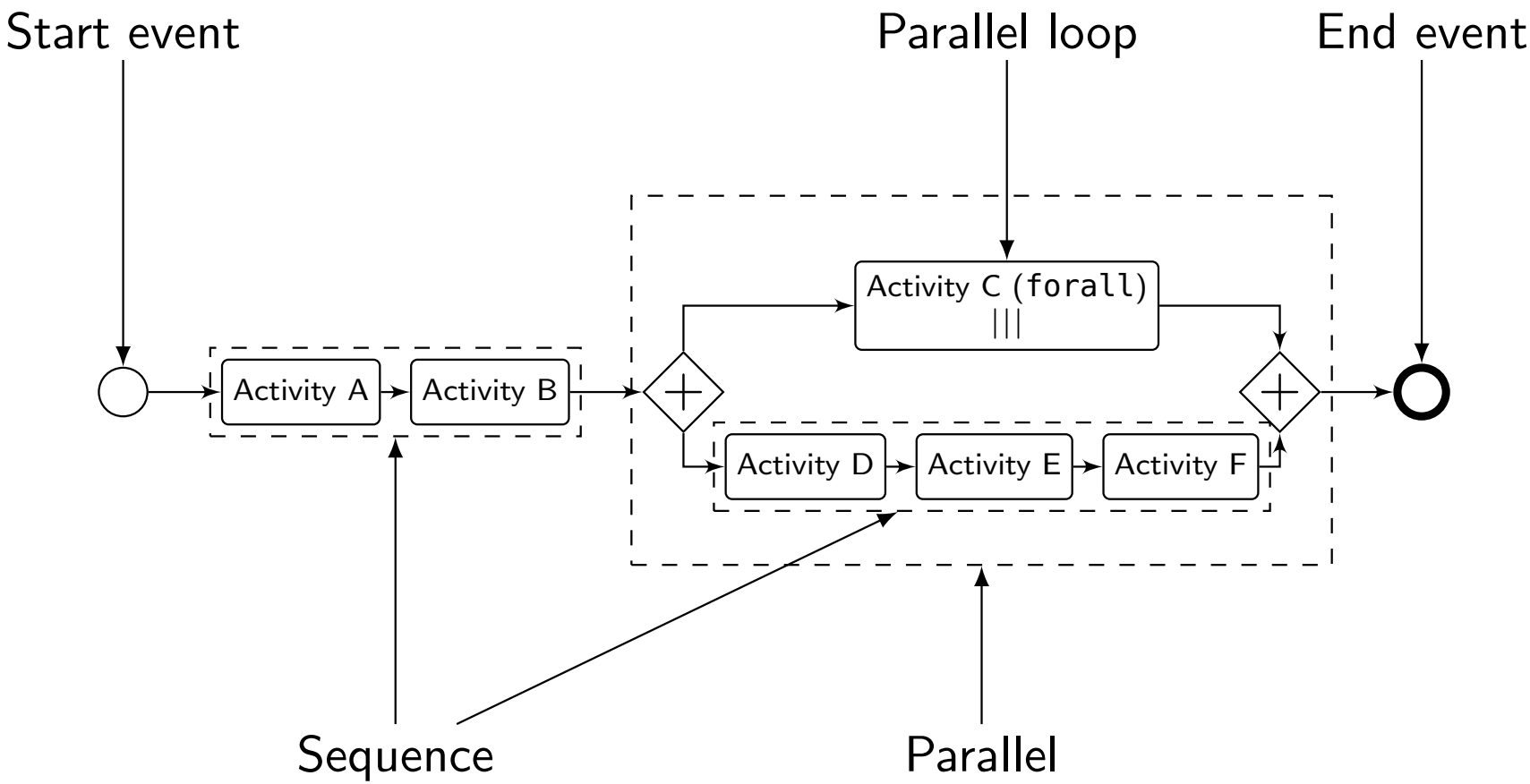
Crash course in workflows



Crash course in workflows



Crash course in workflows



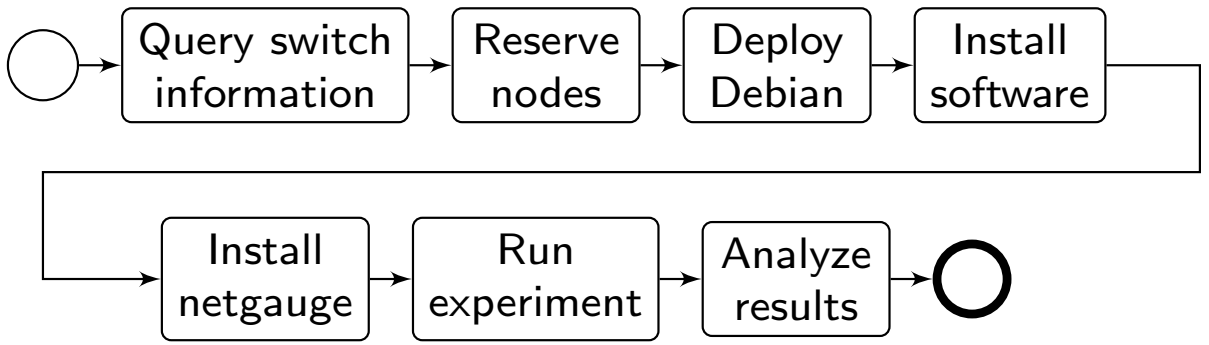
Example of an experiment

Measure the *effective bisection bandwidth* of a switch.

1. Get names of all nodes connected to the switch.
2. Reserve the nodes.
3. Deploy Debian OS.
4. Install necessary software.
5. Compile and install *netgauge*.
6. Run the experiment.
7. Analyze results.

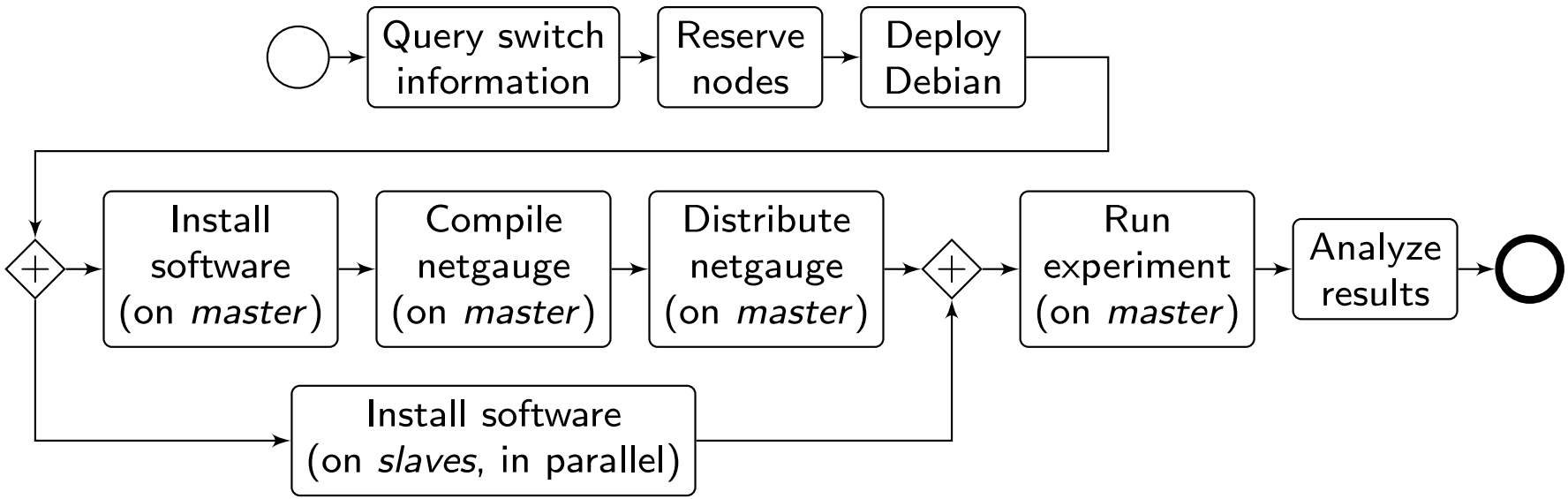
An experiment workflow

We start with a following workflow:



An experiment workflow

... after few adjustments we arrive at:



Results

My current work is represented by 2 contributions:

- analysis of requirements for an experimentation engine
- development of an experimentation engine (XPFlow)

Publication 1

Leveraging business workflows in distributed systems research for the orchestration of reproducible and scalable experiments

In this article we defined goals and requirements for an experiment engine and positioned our approach by comparing it to existing ones.

We showed that BPM can indeed help.

Publication 1 (cont.)

Design

Descriptiveness
Modularity
Reusability
Maintainability
Support for common patterns

Execution

Snapshotting
Error handling
Integration with lower-level tools
Human interaction

Monitoring

Monitoring
Instrumentation
Data analysis

Publication 2

Orchestration d'expériences à l'aide de processus métier

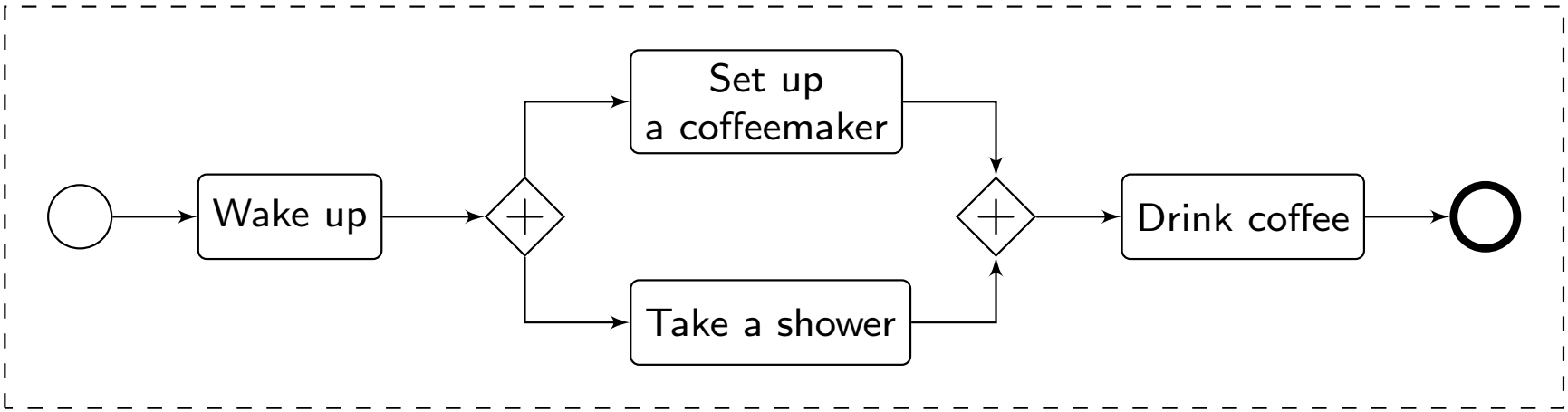
In this article we validated our approach by:

- describing our early implementation
- presenting our language to describe experiments
- testing the new approach with an MPI experiment

Main concepts

There are 2 main concepts in our approach:

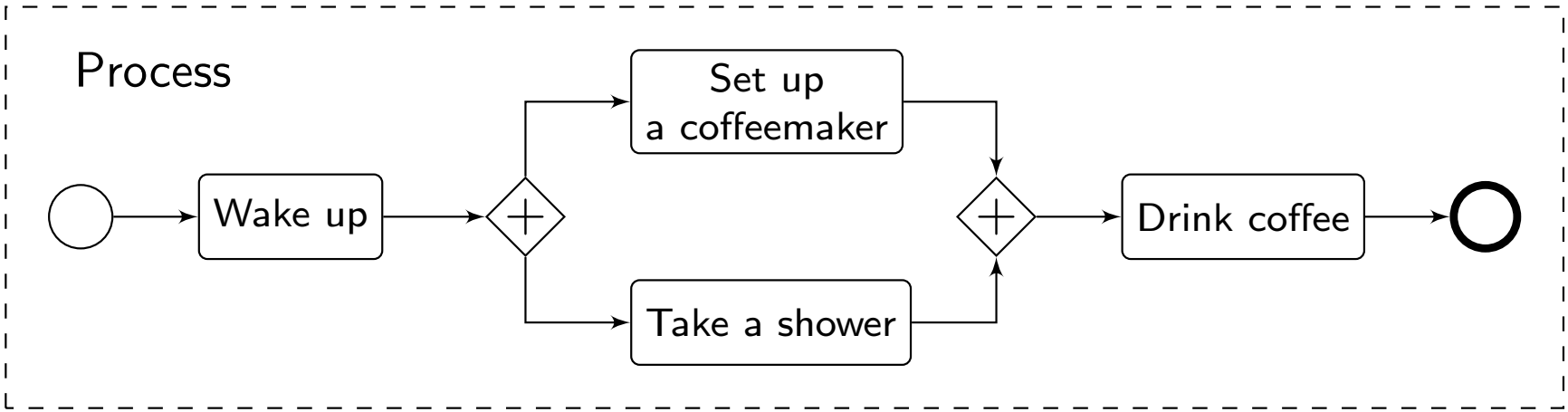
- Processes – high-level description of an experiment:
  - workflows written in a DSL
  - orchestrate other processes and activities
- Activities – low-level building blocks of experiments:
  - do real hard work
  - written in a standard programming language



Main concepts

There are 2 main concepts in our approach:

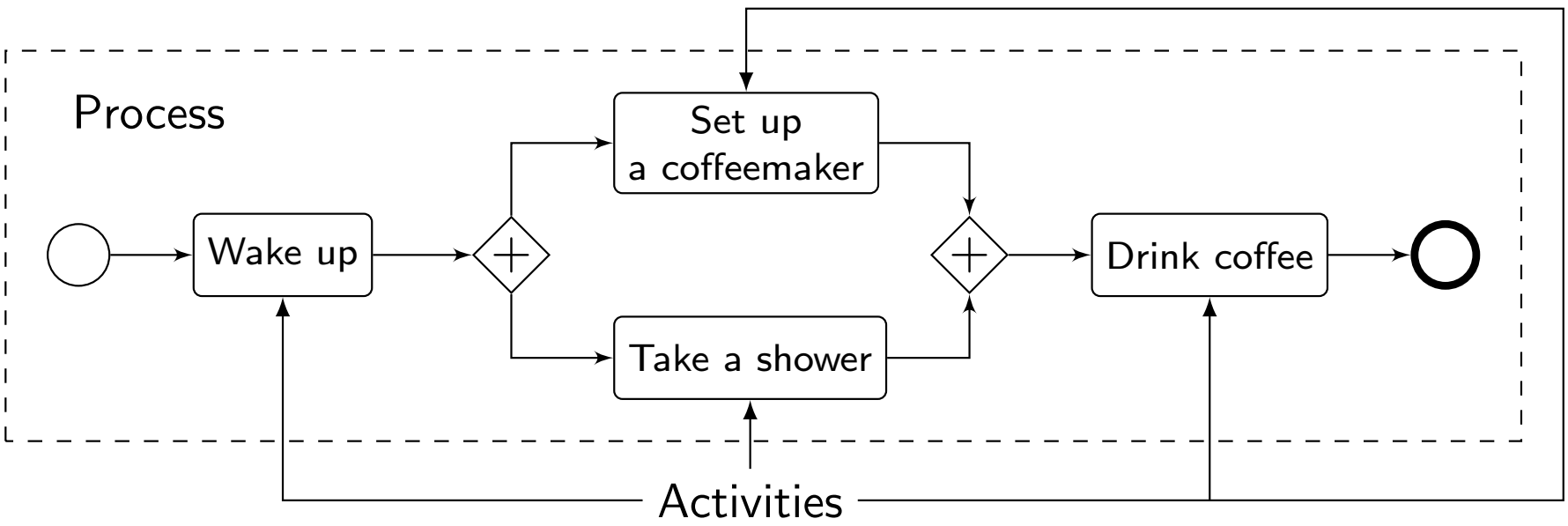
- Processes – high-level description of an experiment:
  - workflows written in a DSL
  - orchestrate other processes and activities
- Activities – low-level building blocks of experiments:
  - do real hard work
  - written in a standard programming language



Main concepts

There are 2 main concepts in our approach:

- Processes – high-level description of an experiment:
  - workflows written in a DSL
  - orchestrate other processes and activities
- Activities – low-level building blocks of experiments:
  - do real hard work
  - written in a standard programming language



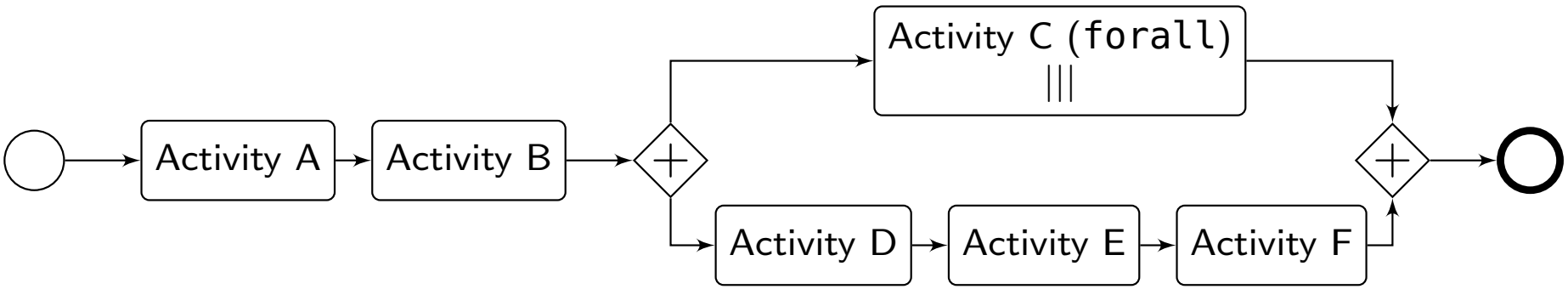
Domain-specific language for processes

The DSL for processes features different **workflow patterns**:

- running activities and other processes (`run`),
- running activities in order or in parallel (`sequence`, `parallel`),
- conditional expressions (`if`, `switch`)
- running sequential and parallel loops (`loop`, `foreach`, `forall`),
- error handling (`try`, `checkpoint`).

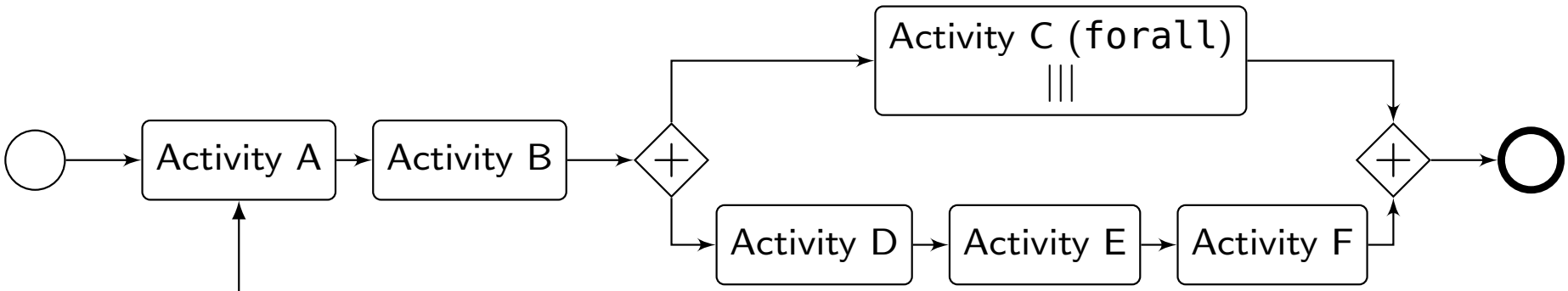
Some of them are taken directly from BPM.

DSL example



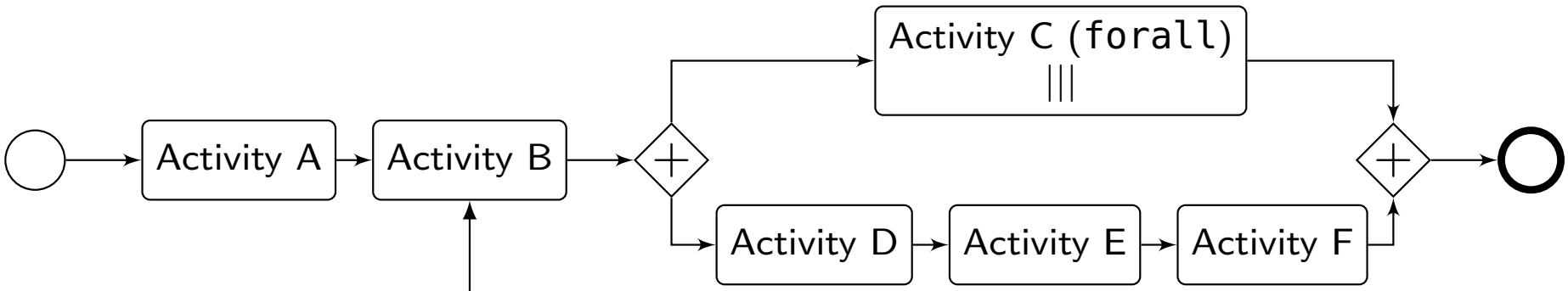

```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

DSL example



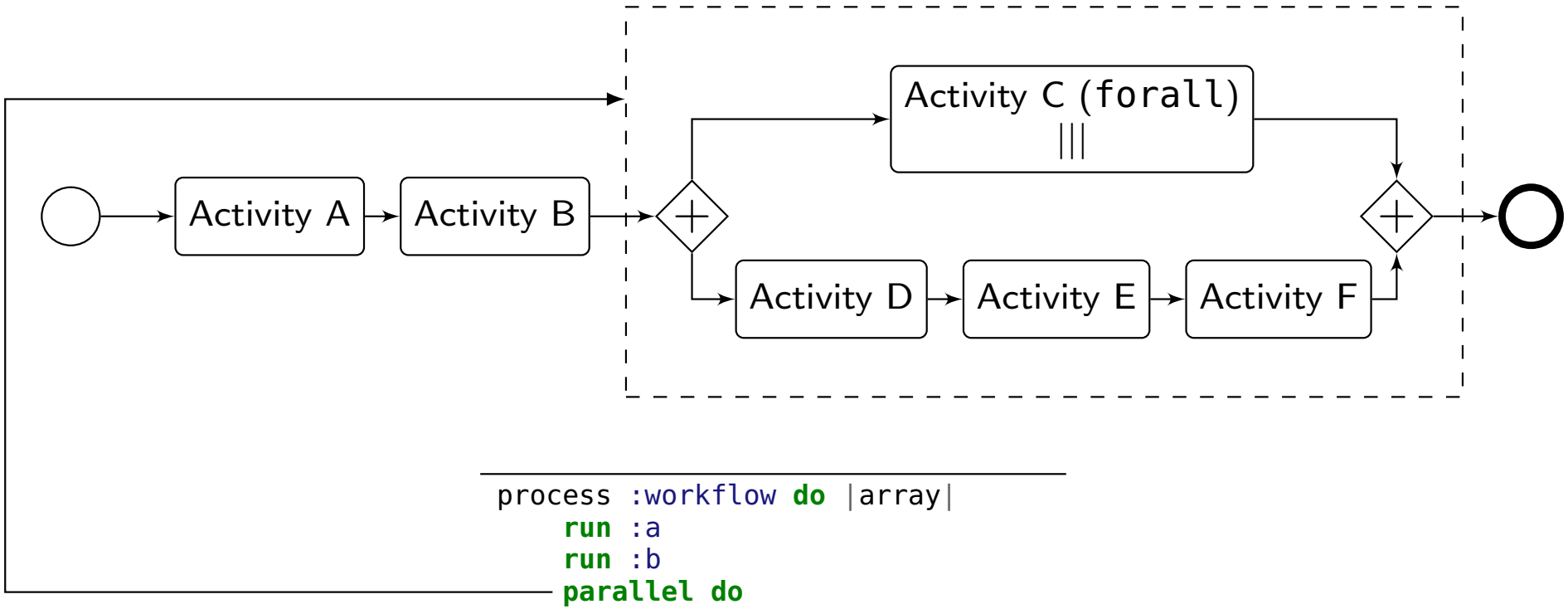

```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

DSL example



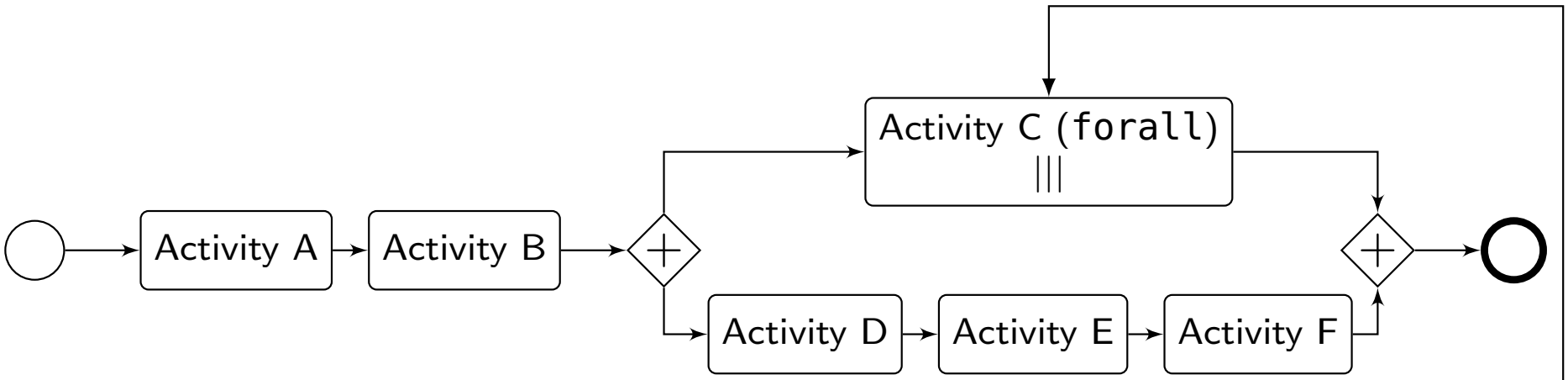

```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

DSL example



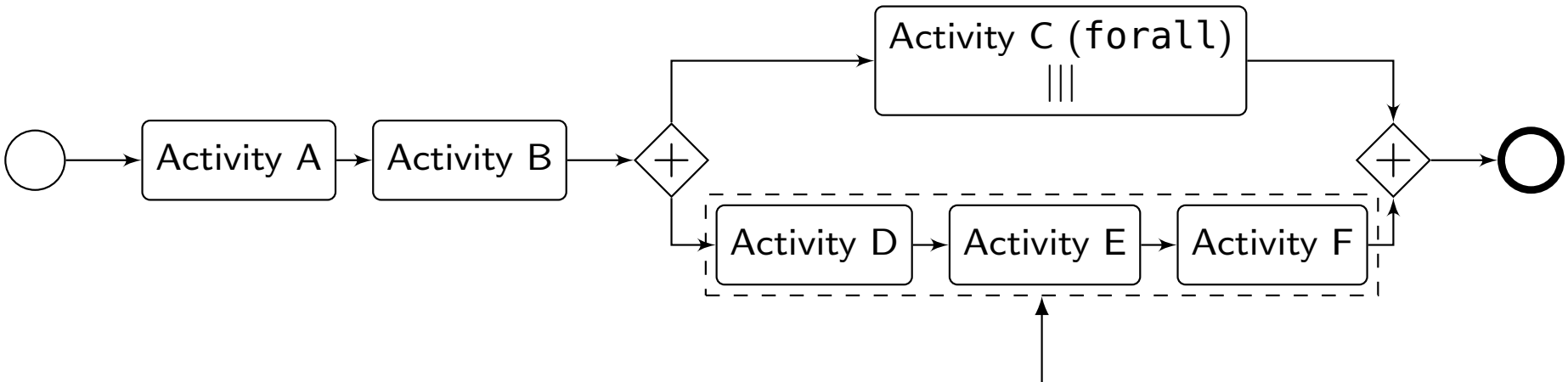

```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

DSL example



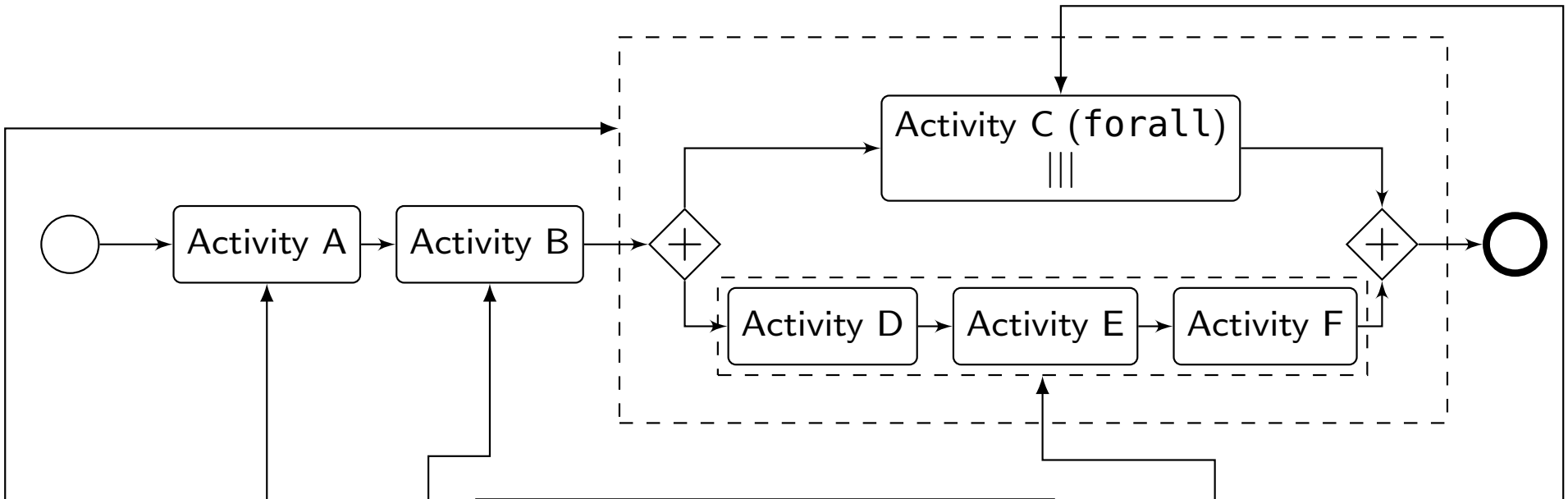

```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

DSL example




```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

DSL example




```
process :workflow do |array|
    run :a
    run :b
    parallel do
        forall array do |x|
            run :c, x
        end
        sequence do
            run :d
            run :e
            run :f
        end
    end
end
```

Error handling

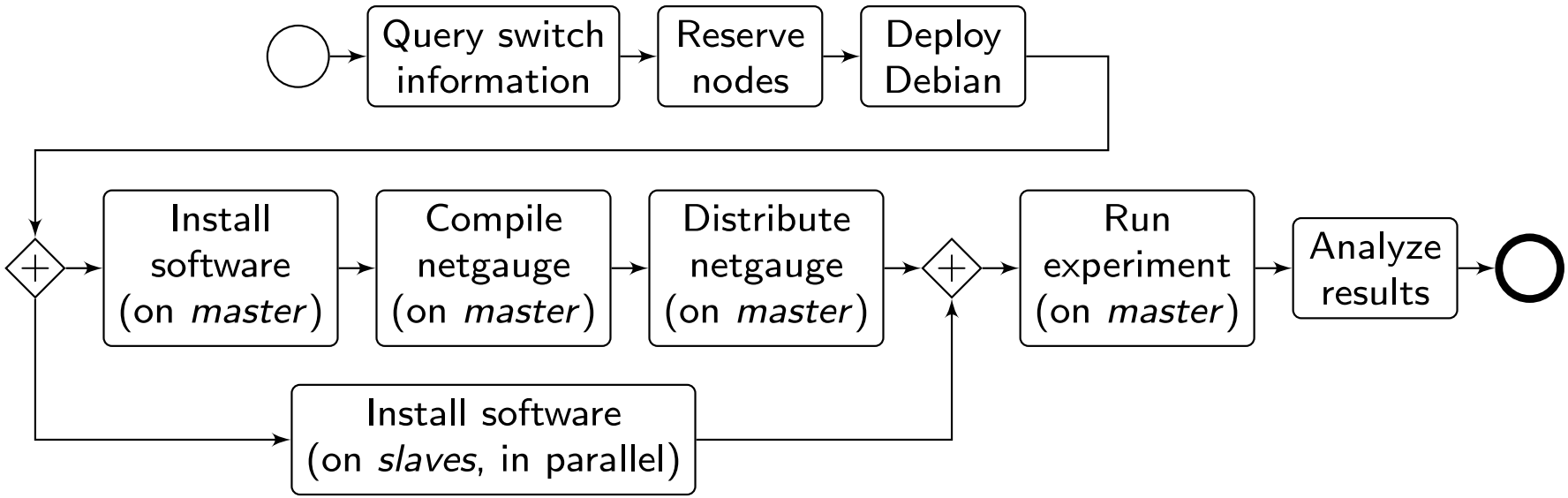
XPFlow gives some means to cope with failures:

- snapshotting:
  - saves a state of an experiment for future use
  - shortens a development's cycle
- retry policy:
  - retries a failed subprocess execution
  - improves reliability

```
process :snapshotting do
    run :long_deployment
    checkpoint :d
    run :experiment
end
```

```
process :retrying do
    try :retry => 5 do
        run :tricky_activity
    end
end
```

An experiment (once again)



An experiment workflow - DSL representation

```
process :exp do |site, switch|
    s = run g5k.switch, site, switch
    ns = run g5k.nodes, s
    r = run g5k.reserve_nodes,
        :nodes => ns, :time => '2h',
        :site => site, :type => :deploy
    master = (first_of ns)
    rest = (tail_of ns)
    run g5k.deploy,
        r, :env => 'squeeze-x64-nfs'
    checkpoint :deployed
    parallel :retry => true do
        forall rest do |slave|
            run :install_pkgs, slave
        end
        sequence do
            run :install_pkgs, master
            run :build_netgauge, master
            run :dist_netgauge,
                master, rest
        end
    end
    checkpoint :prepared
    output = run :netgauge, master, ns
    checkpoint :finished
    run :analysis, output, switch
end
```

An experiment workflow - DSL representation

```
process :exp do |site, switch|
    s = run g5k.switch, site, switch
    ns = run g5k.nodes, s
    r = run g5k.reserve_nodes,
        :nodes => ns, :time => '2h',
        :site => site, :type => :deploy
    master = (first_of ns)
    rest = (tail_of ns)
    run g5k.deploy,
        r, :env => 'squeeze-x64-nfs'
    checkpoint :deployed
    parallel :retry => true do
        forall rest do |slave|
            run :install_pkgs, slave
        end
        sequence do
            run :install_pkgs, master
            run :build_netgauge, master
            run :dist_netgauge,
                master, rest
        end
    end
    checkpoint :prepared
    output = run :netgauge, master, ns
    checkpoint :finished
    run :analysis, output, switch
end
```

Activity :install_pkgs

```
activity :install_pkgs do|node|
    log 'Installing packages on ', node
    run 'g5k.bash', node do
        aptget :update
        aptget :upgrade
        aptget :purge, 'mx'
    end
end
```

An experiment workflow - DSL representation

```
process :exp do |site, switch|
    s = run g5k.switch, site, switch
    ns = run g5k.nodes, s
    r = run g5k.reserve_nodes,
        :nodes => ns, :time => '2h',
        :site => site, :type => :deploy
    master = (first_of ns)
    rest = (tail_of ns)
    run g5k.deploy,
        r, :env => 'squeeze-x64-nfs'
    checkpoint :deployed
    parallel :retry => true do
        forall rest do |slave|
            run :install_pkgs, slave
        end
        sequence do
            run :install_pkgs, master
            run :build_netgauge, master
            run :dist_netgauge,
                master, rest
        end
    end
    checkpoint :prepared
    output = run :netgauge, master, ns
    checkpoint :finished
    run :analysis, output, switch
end
```

Activity :build_netgauge

```
activity :build_netgauge do |master|
    log "Building netgauge on #{master}"
    run 'g5k.copy', NETGAUGE, master, '~'
    run 'g5k.bash', master do
        build_tarball NETGAUGE, PATH
    end
    log "Build finished."
end
```

An experiment workflow - DSL representation

```
process :exp do |site, switch|
    s = run g5k.switch, site, switch
    ns = run g5k.nodes, s
    r = run g5k.reserve_nodes,
        :nodes => ns, :time => '2h',
        :site => site, :type => :deploy
    master = (first_of ns)
    rest = (tail_of ns)
    run g5k.deploy,
        r, :env => 'squeeze-x64-nfs'
    checkpoint :deployed
    parallel :retry => true do
        forall rest do |slave|
            run :install_pkgs, slave
        end
        sequence do
            run :install_pkgs, master
            run :build_netgauge, master
            run :dist_netgauge,
                master, rest
        end
    end
    checkpoint :prepared
    output = run :netgauge, master, ns
    checkpoint :finished
    run :analysis, output, switch
end
```

Activity :dist_netgauge

```
activity :dist_netgauge do |m, s|
    master, slaves = m, s
    run 'g5k.dist_keys', master, slaves
    run 'g5k.bash', master do
        distribute BINARY,
            DEST, 'localhost', slaves
    end
end
```

An experiment workflow - DSL representation

```
process :exp do |site, switch|
    s = run g5k.switch, site, switch
    ns = run g5k.nodes, s
    r = run g5k.reserve_nodes,
        :nodes => ns, :time => '2h',
        :site => site, :type => :deploy
    master = (first_of ns)
    rest = (tail_of ns)
    run g5k.deploy,
        r, :env => 'squeeze-x64-nfs'
    checkpoint :deployed
    parallel :retry => true do
        forall rest do |slave|
            run :install_pkgs, slave
        end
        sequence do
            run :install_pkgs, master
            run :build_netgauge, master
            run :dist_netgauge,
                master, rest
        end
    end
    checkpoint :prepared
    output = run :netgauge, master, ns
    checkpoint :finished
    run :analysis, output, switch
end
```

Activity :netgauge

```
activity :netgauge do |master, nodes|
    log "Running experiment..."
    out = run 'g5k.bash', master do
        cd PATH
        mpirun nodes, "./netgauge"
    end
    log "Experiment done."
end
```

Conclusions

In this talk I presented my work:

- my research topic and the main goals
- 2 (or 3) publications (+ poster)
- implementation of our approach (XPFlow)

Current activities:

- validating the approach in large-scale or cloud scenarios and with other testbeds (e.g., PlanetLab)
- more detailed analysis of related works (survey paper)
- extending the idea to provide reproducibility, provenance, design of experiments, data analysis, validation, better monitoring, etc.
- work on lower-level services for efficient and scalable experiments

More: `http://www.loria.fr/~buchert/`

Thank you for your attention. Questions?