open source ruby workflow and bpm engine

RubyKaigi2007, 19年 06月 10日

メトロー　ジョン

http://openwferu.rubyforge.org

何これ？

- what is workflow ?
  ワークフローってなに?

- what is OpenWFEru ?
  OpenWFEruってなに?

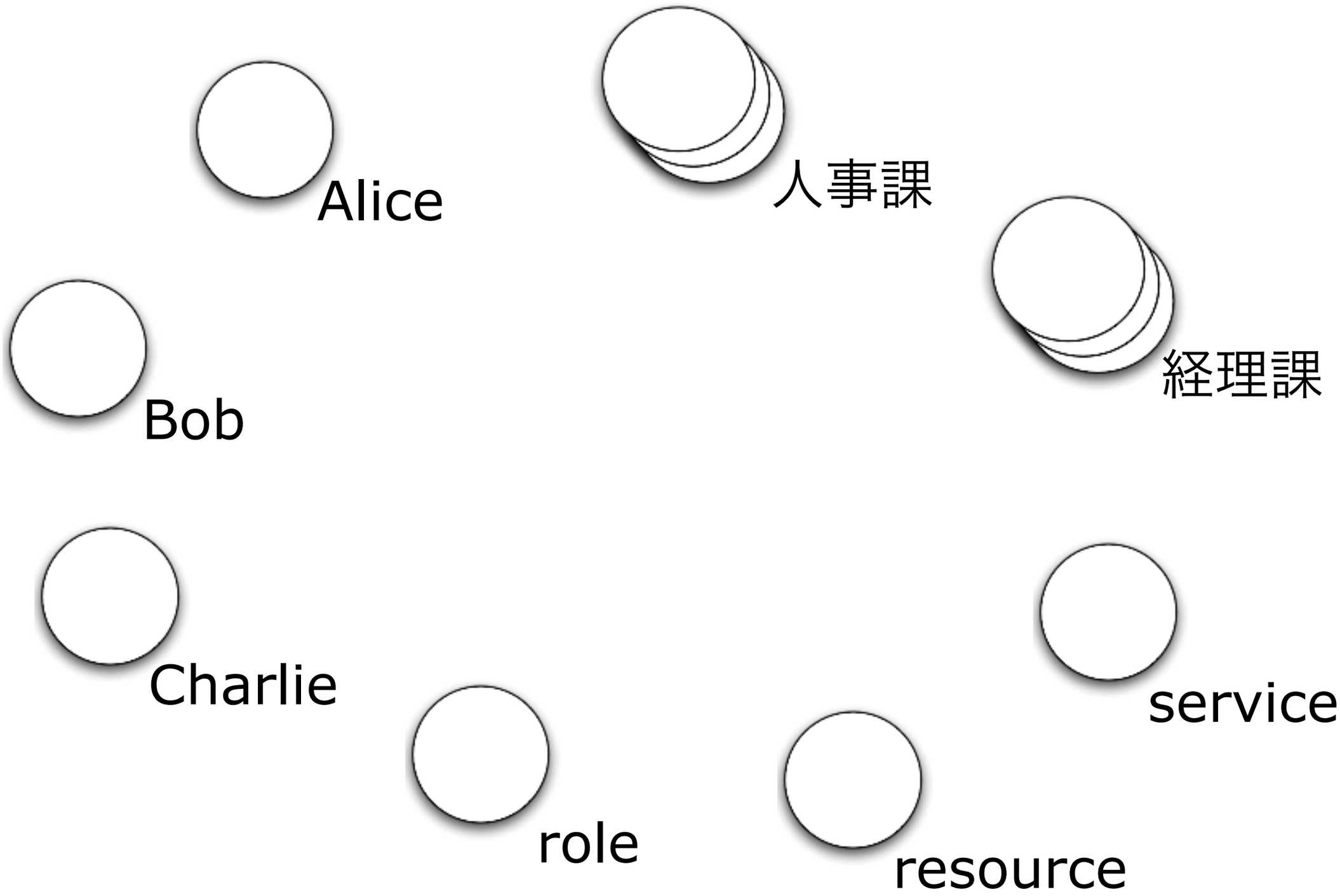
- what it could become
  と、その可能性

- an application written in Ruby
  Rubyで書かれたアプリケーション
- previously written in Java
  元々はJavaだったよ
- application ? solution ?
  アプリケーション? ソリューション?
- to which problem ?
  どっちに使うの？

- RubyKaigi2007 enterprise track
  RubyKaigi2007 エンタープライズトラック

- enterprise, companies, organizations
  エンタープライズ、企業、組織

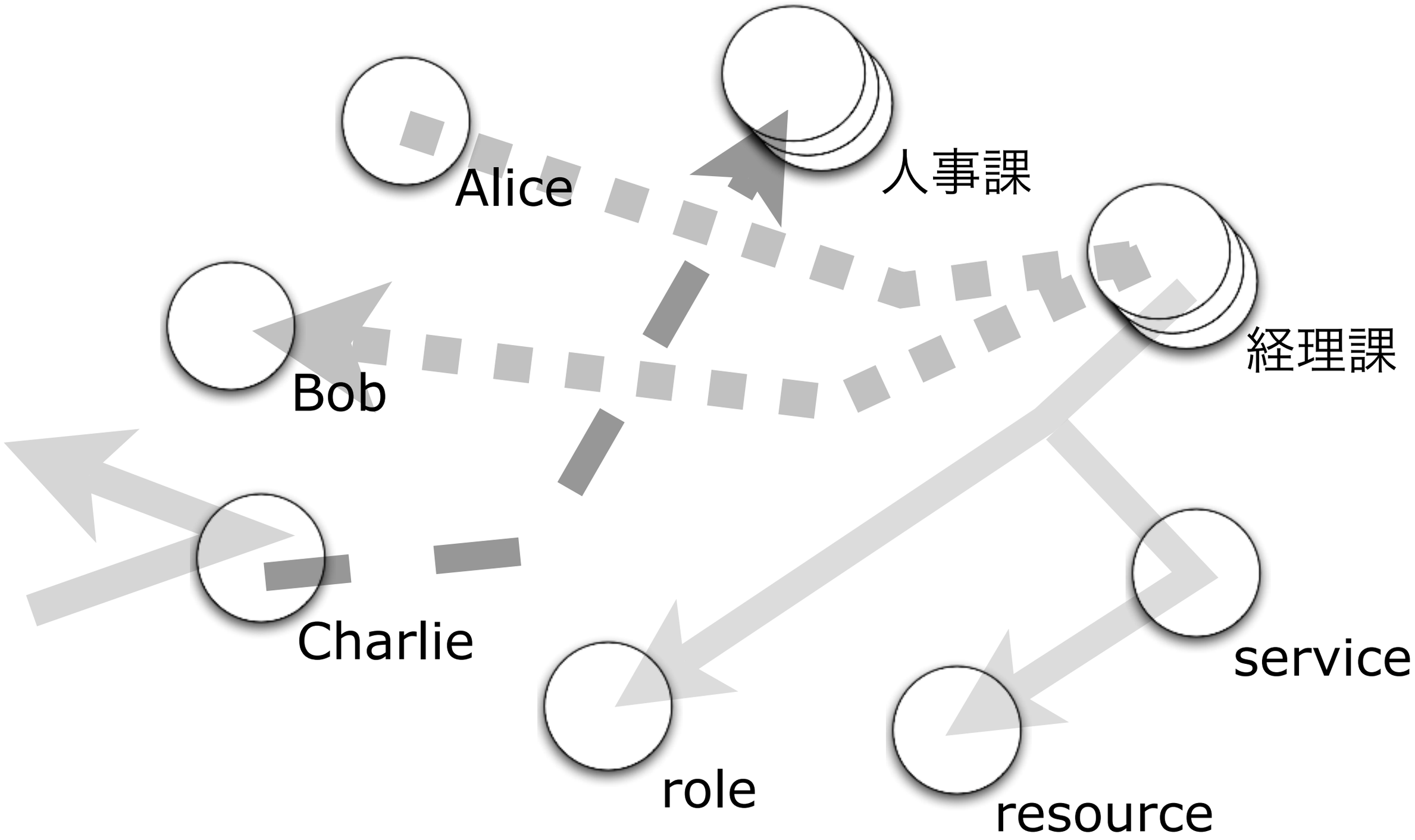
- organization 組織



- participants 参加者

- participants 参加者

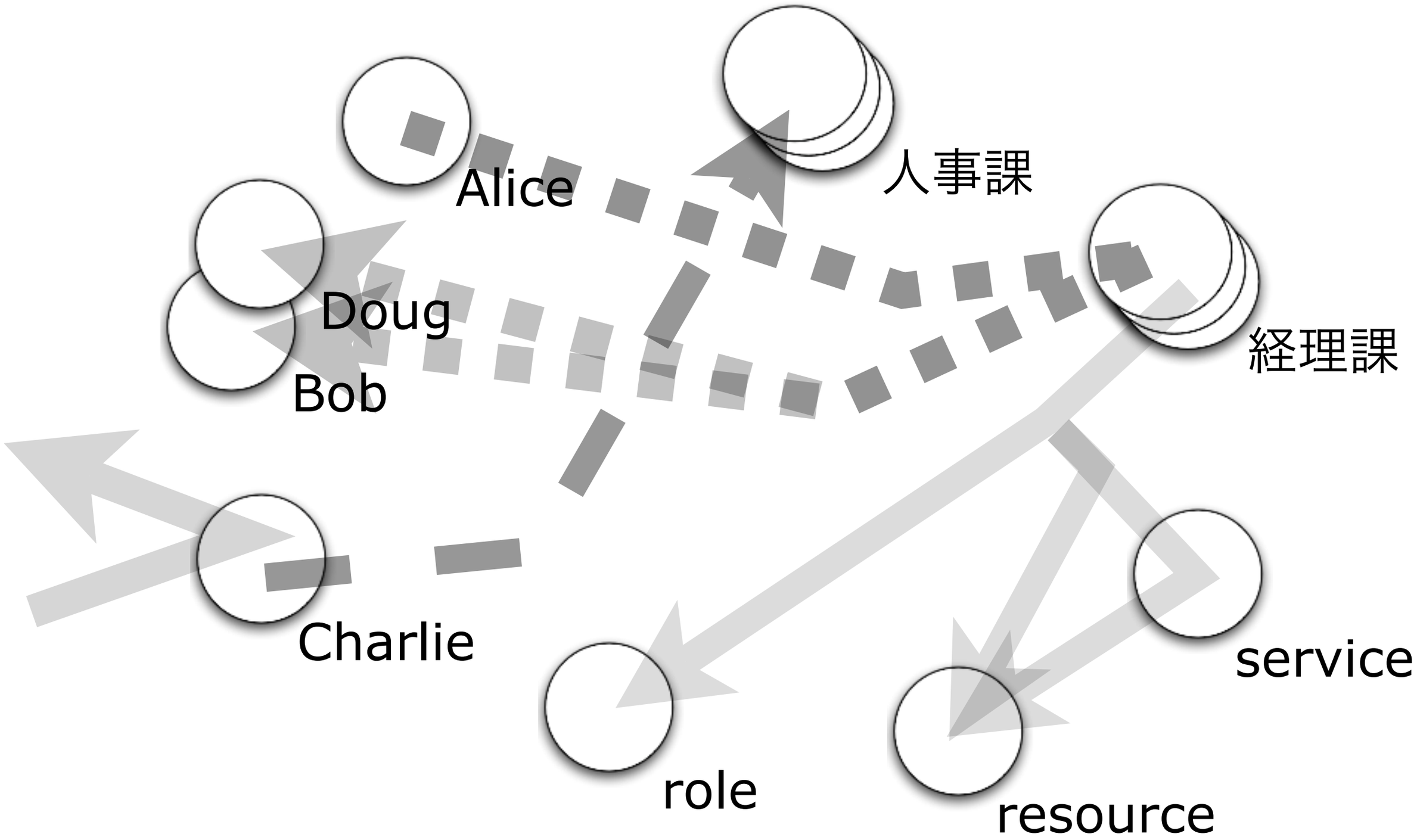
会社



- business processes ビジネスプロセス

- business processes ビジネスプロセス

会社



- changes 変化

変化

- changes
  変化

- the environment changed
  環境が 変化する

- the organization changed
  組織が 変化する

- quest for optimization / rentability
  効率化 / 改善を進める

- processes change sooner than participants do
  参加者よりもプロセスのほうが変わりやすい

- 1 organizational model
  組織モデルは1つでも

- for 1+ business processes
  ビジネスプロセスは1より多い

一級

- it’s easier to change something that is immediately available
  直接触れるものこそ、変えやすい

- that is a first-class citizen
  プロセスは一級市民

- participants and processes
  登場人物とプロセス

- participants ~ services
  登場人物はサービスとして目立ってるけど

- processes ~ usually embedded
  プロセスは大抵埋もれている

- where processes are 1st class citizens
  プロセスが一級市民扱いの世界といえば...
- operating system
- fork() signal() exit()
- plenty of processes running
  プロセスがたくさん走ってる
- participants as files / devices
  参加者はファイルやデバイス

る 

- an OS for business processes
  OpenWFEru はビジネスプロセスのOSだ!

- two 1st class citizens : participants and processes
  二種類の”一級市民”：参加者とプロセス

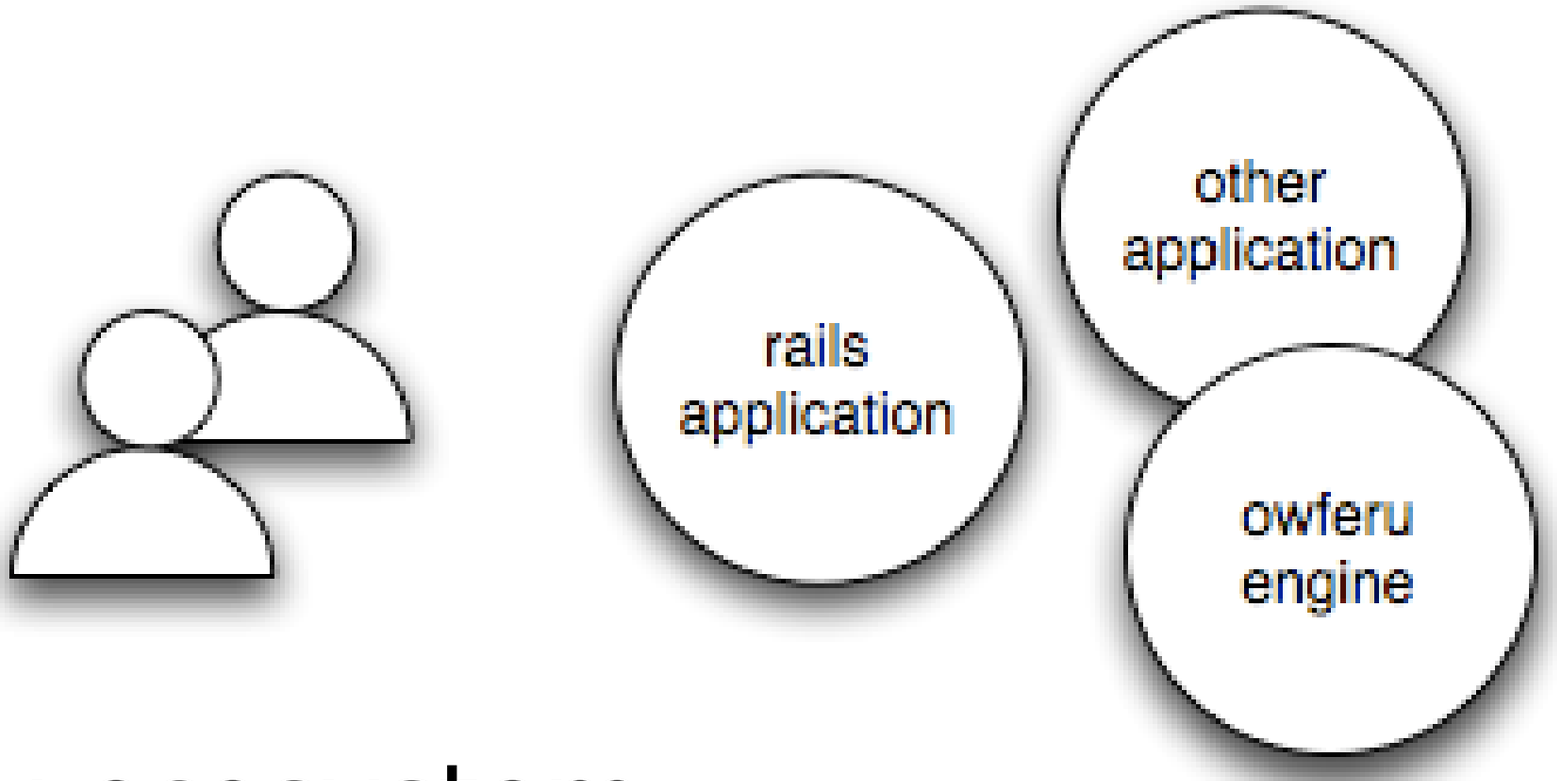
- closer to where things change
  変化により近い

- workflow engine
  ワークフローエンジンとは

- input：organizational model
  入力：組織モデル

- input：process definitions
  入力：プロセス定義

- output：run the processes
  出力：プロセスの実行



- Ruby ecosystem
- camping, ruport, ap4r, drb, ...
- Jruby =>
  the whole java circus (Java大サーカス)
- rest =>
  from .NET, python, perl, pnuts, ...

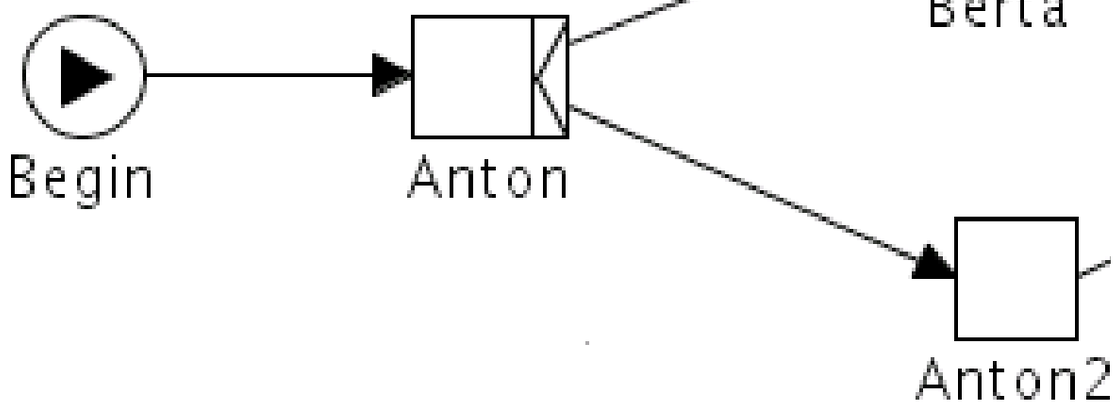
コード

```ruby
require 'rubygems'
require 'openwfe/def'
require 'openwfe/workitem'
require 'openwfe/engine/engine'

# instantiating an engine

engine = OpenWFE::Engine.new

# adding some participants

engine.register_participant :alice do |workitem|
    workitem.alice_comment = 'このお客さんは重要です'
end
engine.register_participant :bob do |workitem|
    workitem.attributes['bob_comment'] = 'そうだね'
end

process_definition = '''
    <process-definition name="customer_request_review" revision="1.5">
        <sequence>
            <participant ref="alice" />
            <participant ref="bob" />
        </sequence>
    </process-definition>
'''
# launching the process

li = OpenWFE::LaunchItem.new(process_definition)
li.customer_name = 'Sasada Koichi'

fei = engine.launch(li)
```

ズーム

```
<process-definition name="customer_request_review" revision="1.5">
    <sequence>
        <participant ref="alice" />
        <participant ref="bob" />
    </sequence>
</process-definition>

<process-definition name="customer_reque
    <sequence>
        <concurrence>
            <participant ref="alice" />
            <participant ref="bob" />
        </concurrence>
        <participant ref="charly" />
    </sequence>
</process-definition>
```

- easy to diff
  diffできるよ




```
<?xml version="1.0" encoding="UTF-8"?>
<specificationSet xmlns="http://www.citi.qut.edu.au/yawl" />
  <specification uri="test.ywl">
    <metaData />
    <schema xmlns="http://www.w3.org/2001/XMLSchema" />
    <decomposition id="YAWL_test_net" isRootNet="true" xsi:type="NetFactsType">
      <processControlElements>
        <inputCondition id="0_InputCondition">
          <flowsInto>
            <nextElementRef id="2_Anton" />
          </flowsInto>
        </inputCondition>
        <task id="2_Anton">
          <flowsInto>
            <nextElementRef id="3_Anton2" />
          </flowsInto>
          <flowsInto>
            <nextElementRef id="5_Berta" />
          </flowsInto>
          <join code="xor" />
          <split code="and" />
          <decomposesTo id="Anton" />
        </task>
        <task id="5_Berta">
          <flowsInto>
            <nextElementRef id="4_Cesar" />
          </flowsInto>
          <join code="xor" />
          <split code="and" />
          <decomposesTo id="Berta" />
        </task>
        <task id="3_Anton2">
          <flowsInto>
            <nextElementRef id="4_Cesar" />
          </flowsInto>
          <join code="xor" />
          <split code="and" />
          <decomposesTo id="Anton2" />
        </task>
        <task id="4_Cesar">
          <flowsInto>
            <nextElementRef id="1_OutputCondition" />
          </flowsInto>
          <join code="and" />
          <split code="and" />
          <decomposesTo id="Cesar" />
        </task>
        <outputCondition id="1_OutputCondition" />
      </processControlElements>
    </decomposition>
    <decomposition id="Berta" xsi:type="WebServiceGatewayFactsType" />
    <decomposition id="Anton" xsi:type="WebServiceGatewayFactsType" />
    <decomposition id="Anton2" xsi:type="WebServiceGatewayFactsType" />
    <decomposition id="Cesar" xsi:type="WebServiceGatewayFactsType" />
  </specification>
</specificationSet>
```

- concise
  すっきり
- high level concepts
  ハイレベルなコンセプト

ルビー

```
process_definition = '''
    <process-definition name="customer_request_review" revision="1.5">
        <sequence>
            <participant ref="alice" />
            <participant ref="bob" />
        </sequence>
    </process-definition>
'''
```

- XML or Ruby
- XML <-> Ruby
- Ruby for ‘macros’

```
class CustomerRequestReview0 < OpenWFE::Proc
    sequence do
        concurrence do
            participant :ref => :alice
            participant :ref => :bob
        end
        participant :ref => :charly
    end
end

class CustomerRequestReview0 < OpenWFE::ProcessDefinition
    sequence do
        concurrence :count => 2 do
            [ :alice, :bob, :doug ].each do |p|
                participant :ref => p
            end
        end
        participant :ref => :charly
    end
end
```

- participant (登場人物)
- process-definition (プロセス定義)
- concurrence, sequence (並列と直列)
- timeout, sleep, wait
- redo, undo, cancel
- listen
- cancel-process

参加者

```
# adding some participants

engine.register_participant :alice do |workitem|
    workitem.alice_comment = 'このお客さんは重要です'
end
engine.register_participant :bob do |workitem|
    workitem.attributes['bob_comment'] = 'そうだね'
end

require 'openwfe/participants/enoparticipants'

p = OpenWFE::MailParticipant.new(:from_address => "bpm.system@acme.com") do |workitem|
    s = "Subject: news from the BPM system\n\n"
    s << "dear #{workitem.customer_name}\n"
    s << "..."
    s
end
engine.register_participant(:charly, p)

require 'openwfe/participants/sqsparticipants'

engine.register_participant :denshiro, OpenWFE::SqsParticipant.new("WorkflowQueue")

require 'openwfe/worklist/storeparticipant'

engine.register_participant "^store_.*", OpenWFE::YamlParticipant

require 'your/participants'

engine.register_participant "^role_.*", YourLib::ActiveWhatever.new
```

- blocks
- mail
- amazon SQS
- yaml file
- 色々...

```
--- !ruby/object:OpenWFE::InFlowWorkItem
attributes:
  customer_address: gion 1-6-2
  customer_name: sasaki kojirou
  params: {}

  ___map_type: smap
  __result__: gion 1-6-2
flow_expression_id: !ruby/object:OpenWFE::FlowExpressionId
  engine_id: engine
  expression_id: 0.0.2
  expression_name: participant
  initial_engine_id: engine
  owfe_version: 0.9.11
  workflow_definition_name: Test
  workflow_definition_revision: "0"
  workflow_definition_url: field:__definition
  workflow_instance_id: 20070603-jiyurajaro
last_modified:
participant_name: sales_team
```

- payload
- identifier
- participant name

- flexible
- neutral

- definitions, participants and the engine
  定義と参加者、そしてエンジン

- long running processes
  長命なプロセスは、

- may runner longer than the engine
  エンジン自体よりも長生きなので

- persistence
  永続化

  - stop/restart
  - migrations
  - fixes

永続化

```
require 'rubygems'
require 'openwfe/engine/file_persisted_engine'
require 'openwfe/expool/history'
require 'openwfe/expool/journal'

engine = OpenWFE::CachedFilePersistedEngine.new


engine.init_service("history", FileHistory)
engine.init_service("journal", Journal)

engine.application_context[:keep_journals] = true
```

```
# execution is persistence and kept
# in journals (execution logs)

engine.get_journal.replay 'work/journal/20070603-gejikogoza.journal', 0


engine.register_participant :role_reviewer do |workitem|
    puts "received a workitem with #{workitem.attributes.length} attributes"
    workitem.attributes['reviewed'] = true
end

engine.get_journal.replay_at_last_error '20070603-jekijogoza'
```

- workflow engine
- 0.9.11
- yaml / fs persistence
- old OpenWFE REST interface (2003)
- journaling
- ...
- increasing usage (GeoBPMS, job offers, ...)

将来

- 0.9.12 should work 100% on JRuby
- web process definition designer
- ActiveRecord persistence
- monitoring tools
- simulation tools
- other process definition languages
- more distribution
- more documentation (日本語 ?)
- ...

終わり

OPEN WFE-る

- john mettraux　　メトロー　ジョン
- smtp : john at openwfe dot org
- blog : http://jmettraux.openwfe.org
- source : http://openwferu.rubyforge.org