ESUG 2019: Concurrency

by Santiago Bragagnolo - Esug - 2019
santiago.bragagnolo@gmail.com
santiago.bragagnolo@inria.fr
skype:santiago.bragagnolo
@sbragagnolo

RMod

Who am I

(i have less hair now, same appetite)

- **2002 - 2012**
  - Software engineer/developer in the private sector
  - Teaching programming
- **2012 - 2019**
  - Research engineer @ Ecole de mines & INRIA.
- **2019 - ????**
  - Starting a PhD :)

RMod

Process

In computing, a **process** is an instance of a computer program that is being sequentially executed[1] by a computer system that has the ability to run several computer programs concurrently.

Merriam-Webster

2 **a** **(1)** **:** a natural phenomenon marked by gradual changes that lead toward a particular result

// the *process* of growth

(2) **:** a continuing natural or biological activity or function

// such life *processes* as breathing

b **:** a series of actions or operations conducing to an end

especially **:** a continuous operation or treatment especially in manufacture

Inria inventeurs du monde numérique 50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur

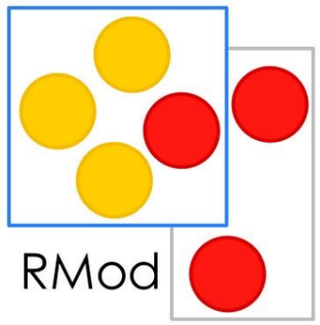
Processes: Living entities

Life cycle

- Born
- Grow
- Reproduce / Exchange
- Die

```
exchange := nil.
process := [
    self grow.
    exchange := #something.
    process die.
] beBorn
```

inventeurs du monde numérique

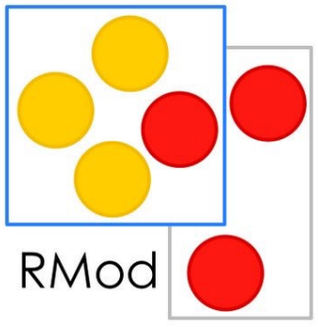


Processes in Pharo

Workspace

```
exchange := nil.
process := [
    'Business logic here!'.
    self inform: 'Hello from process: ', Processor activeProcess name.
    exchange := #somevalue.
    ] forkAt: Processor systemBackgroundPriority  named: #EsugExample.
```

Smalltalk script W +L



Processes: example of usage

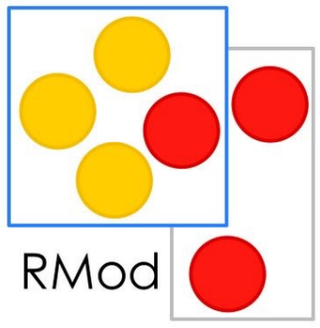
resetCompletionDelay

```
"Open the popup after 100ms and only after certain characters"
self stopCompletionDelay.
self isMenuOpen ifTrue: [ ^ self ].
editor atCompletionPosition ifFalse: [ ^ self ].

completionDelay := [
    (Delay forMilliseconds: NECPreferences popupAutomaticDelay) wait.
    UIManager default defer:  [
        editor atCompletionPosition ifTrue: [ self openMenu ]]
] fork.
```

Processes: example of usage

inventeurs du monde numérique

1867 › 2017 › 2067 Imaginons notre futur



Processes: example of usage

```
serveConnectionsOn: listeningSocket
    "We wait up to acceptWaitTimeout seconds for an incoming connection.
    If we get one we wrap it in a SocketStream and #executeRequestResponseLoopOn: on it"

    | stream socket |
    socket := listeningSocket waitForAcceptFor: self acceptWaitTimeout.
    socket ifNil: [ ^ self noteAcceptWaitTimedOut ].
    stream := self socketStreamOn: socket.
    [ [ [ self executeRequestResponseLoopOn: stream ]
        ensure: [ self logConnectionClosed: stream. self closeSocketStream: stream ] ]
            ifCurtailed: [ socket destroy ] ]
                forkAt: Processor lowIOPriority
                named: self workerProcessName
```

Processes: example of usage

inventeurs du monde numérique

1967 › 2017 ›› 2067 Imaginons notre futur

RMod

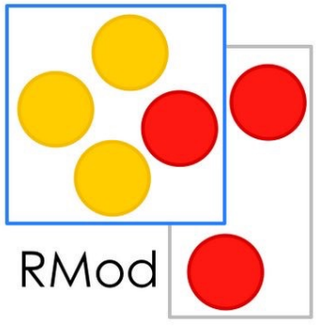
Processes: example of usage

Workspace

```
tickets := Stack new.
tickets add: 1.

buyingTicketProcess := [
	tickets isEmpty ifFalse: [
		100 milliSecond wait.
		self inform: 'Getting ticket number: ', tickets pop printString.
	] ifTrue: [
		self inform: 'No more tickets!'.
	].
].
user1 := buyingTicketProcess forkNamed: #User1Process.
user2 := buyingTicketProcess forkNamed: #User2Process.
```

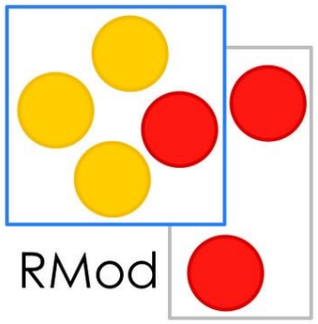
Smalltalk script W +L

Processes: example of usage



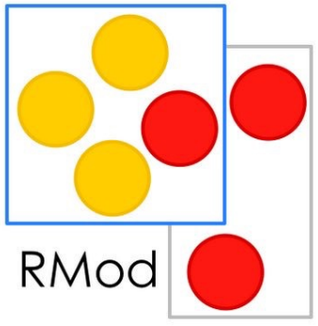
Manage process's life cycle is painful

- When to start a process?
- When to kill a process?
- How to keep a process alive?
- How to synchronise them?



What is TaskIt

- Task focused concurrency framework
- Open source (https://github.com/sbragagnolo/taskit)
- Used in projects where performance matters (PhaROS, Makros, Fog, etc)
- 6 years old



Why TaskIt

- Synchronise different tasks
- Unlock development perspectives
- Process lifecycle agnostic
- Process lifecycle fanatic

TaskIT blueprints

ALCATRAZ ISLAND
LIGHTHOUSE
1909
ALCATRAZ ISLAND
SAN FRANCISCO, CA 94123

DESTRUCTION ISLAND
LIGHTHOUSE
1892

CAPE HATTERAS LIGHTHOUSE
1870
[illegible] LIGHTHOUSE RD
[illegible]TON, NC 27920

LIGHTHOUSE
TOWER
(1873)

SPLIT ROCK LIGHTHOUSE
1910
3713 SPLIT ROCK LIGHTHOUSE RD
TWO HARBORS, MN 55616

ATTERAS LIGHTHOUSE
ks 1870 North Carolina

BODIE ISLAND LIGHTHOUSE
Nags Head • North Carolina
1872

ARCHITECT: RALPH RUSSELL TINKHAM
YEARS OF OPERATION: 1910-69
LAKE SUPERIOR ELEVATION: 602 FEET ABOVE SEA LEVEL
CLIFF HEIGHT: 130 FEET
COST: $75,000 FOR LAND AND BUILDINGS
OFFICIAL RANGE: 22 MILES
FLASHING SEQUENCE: ONCE EVERY 10 SECONDS (0.5 SECONDS EVERY 9.5 SECONDS)
LIGHT SOURCE: INCANDESCENT OIL-VAPOR (KEROSENE) LAMP, 1910-39; 1,000-WATT ELECTRIC BULB, 1940-69
TONE: SIREN BLAST, 1910-35; TYPE F DIAPHONE, 1936-61
YEARS OF OPERATION: 1910-61
SOUNDING SEQUENCE: 2 SECOND BLAST, 18 SECOND SILENCE
EFFECTIVE RANGE: 5 MILES

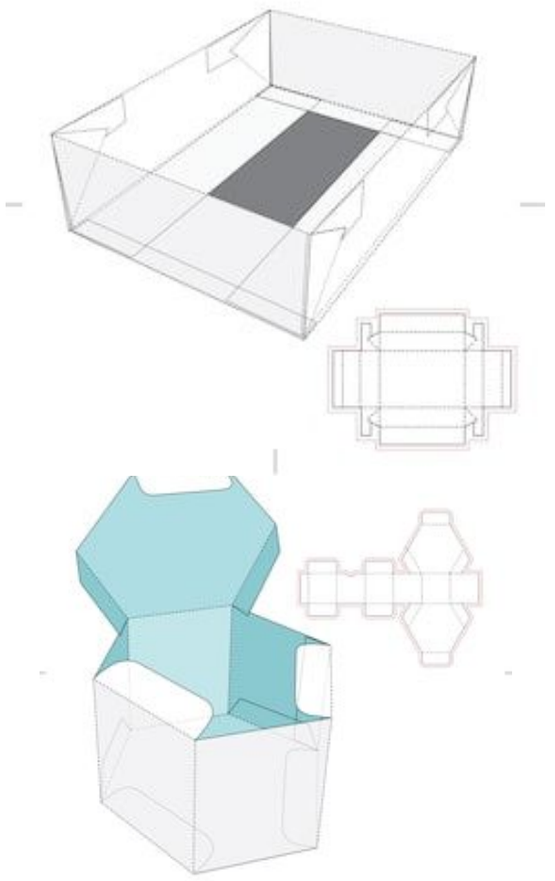
Living Room
Kitchen
Stores
EntranceRoom
Spring Tide +5·4 m
H.Water +3·0 m
L Water
Datum
Red
Sand

RMod

Tasks

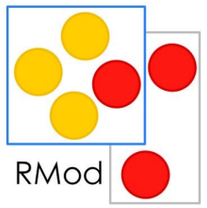
like programs, but smaller

- Objects
- Reusable computation units
- Process agnostic
- Built up from
  - Message send
  - Blocks



inventeurs du monde numérique

50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur



Task Examples

```
[ 'Happened' logCr ] schedule.
```

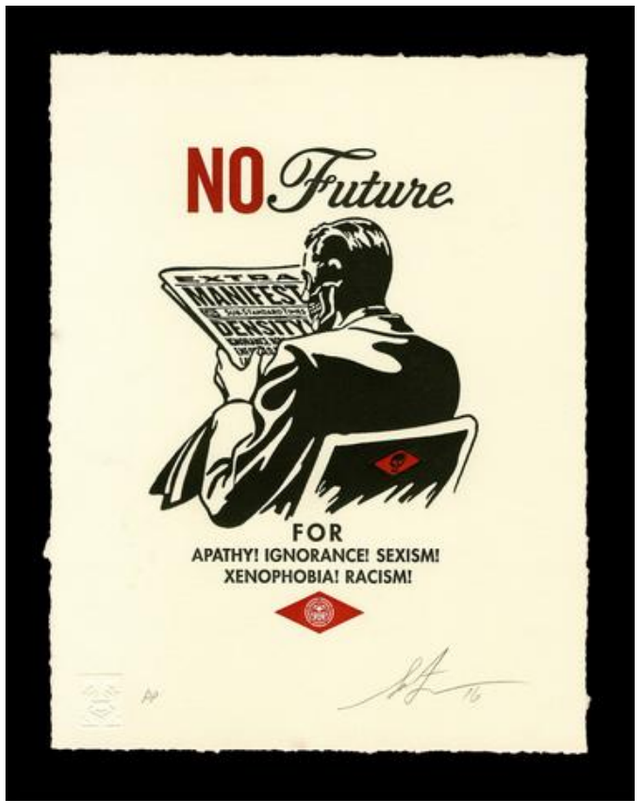
we do not care about when this task would be executed, not either it result

```
future := [ 2 + 2 ] future.
```

we do not care about when it will be executed, yet we do care about the result

Scheduled Task

- The task will be executed at some point
- Does not matter when
- No need of synchronisation



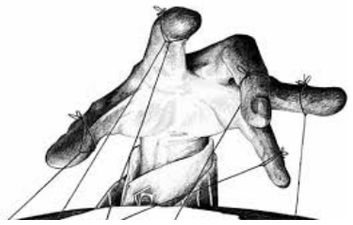
inventeurs du monde numérique

Scheduled Task

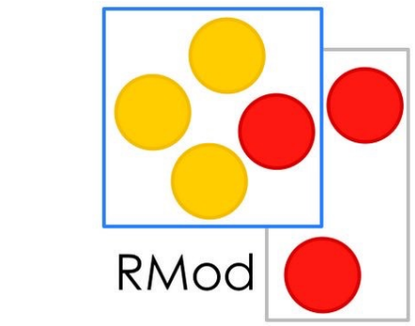
My call

```
client := Client new.
client id: UUID new.
[ self inform: 'save client: '.
client id asString ] schedule.
self save: client.
```

The Invisible hand of running strategy

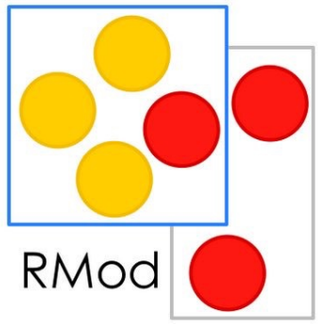


```
[ self inform: 'save client: '.
  client id asString ]
```



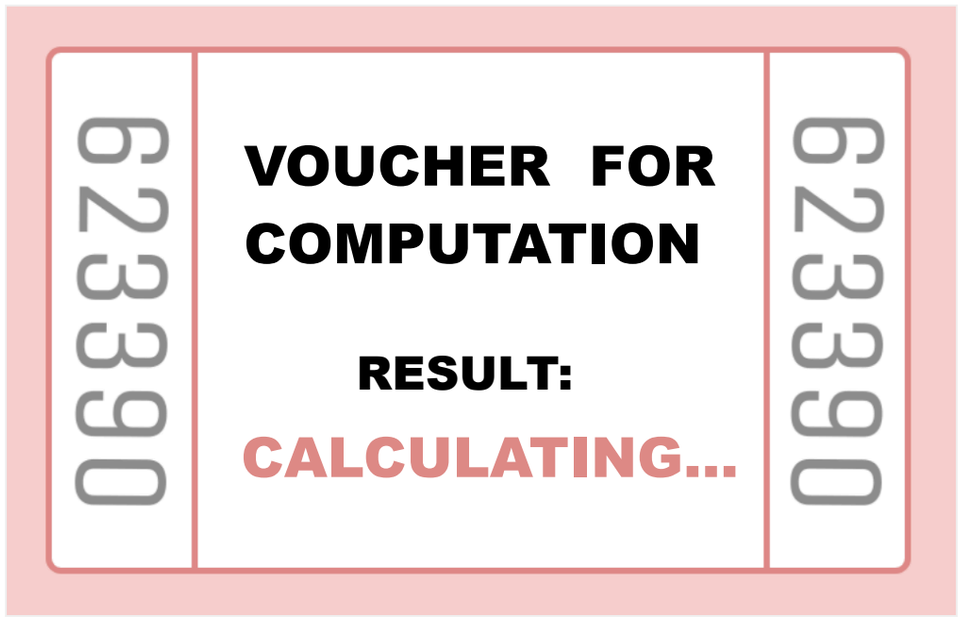
Futures

- Objects
- Represent the future of a computation
- Process agnostic



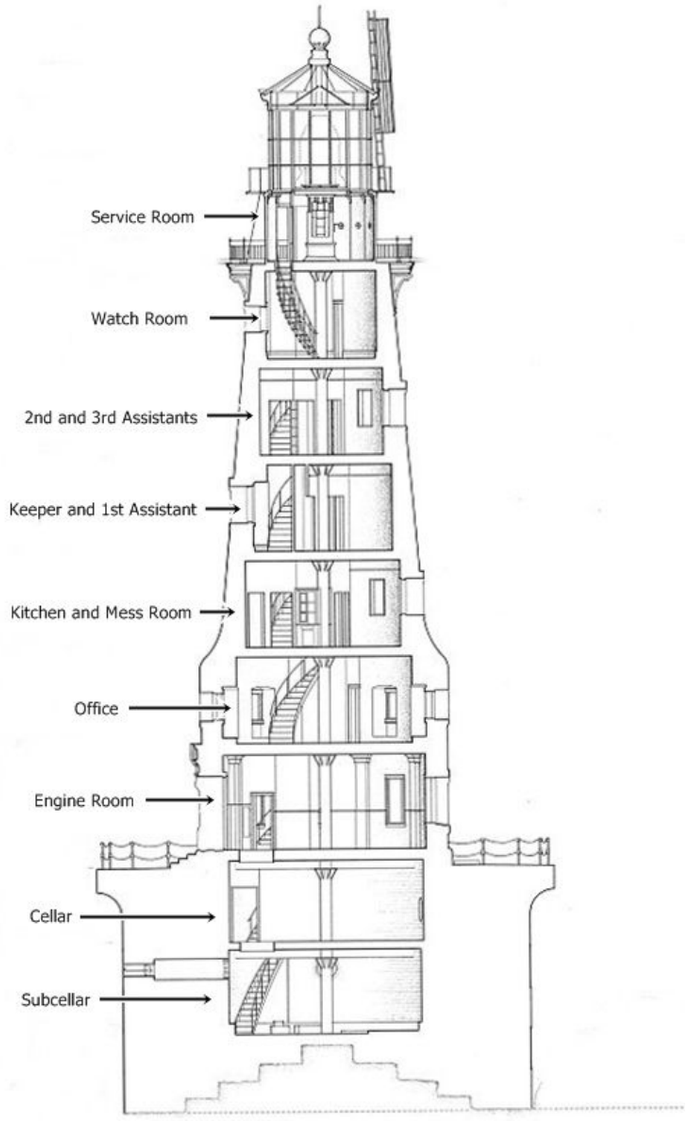
Futures

- As mean for getting the computed task result



Futures

- As mean of synchronisation
  - Synchronous
  - Asynchronous
  - Tasks combination



Synchronous

My call

```
future := [
stream write: data.
stream read
] future.
```

```
response := future synchronizeTimeout: 10 seconds.
```

```
self execute: line.
```

The Invisible hand
of running strategy

```
stream write: data.
v := stream read
future deploy: v
```

- **synchronous**
- asynchronous
- task combination

Synchronous

Playground

Page

```
stream := #file asFileReference  readStream.
future := [
    1 second wait.
    stream nextLine
] future.
self inform: (future synchronizeTimeout: 10 seconds).
self inform: 'After Synchro'
```

- **synchronous**
- asynchronous
- task combination

inventeurs du monde numérique

50 ANS 1967 ›› 2017 ›› 2067 Imaginons notre futur

Synchronous

Playground

Page

```
content :=   'Content'.
future := [
    content at: 10 put: $b.
] future.
future synchronizeTimeout: 10 seconds.
self inform: 'After Synchro'
```

- **synchronous**
- asynchronous
- task combination

inventeurs du monde numérique

1967 › 2017 ›› 2067 Imaginons notre futur

Asynchronous

My call

```
future := [
stream write: data.
stream read
] future.
future onSuccessDo:
[ : v |
    self execute: v.
]
```

The Invisible hand
of running strategy

```
stream write: data.
v := stream read
future deploy: v
```

VOUCHER FOR COMPUTATION
RESULT: CALCULATING...
623390 623390

```
self execute: v.
```

- synchronous
- **asynchronous**
- task combination

inventeurs du monde numérique
50 ANS
1967 › 2017 ›› 2067
Imaginons notre futur

RMod

Asynchronous

Playground

Page

```
stream := #file asFileReference  readStream.
future := [
    1 second wait.
    stream nextLine
] future.
future onSuccessDo: [ : v |  self inform: v.  ].
self inform: 'Before Synchro'
```

- synchronous
- **asynchronous**
- task combination

Inria inventeurs du monde numérique

50 ANS 1967 2017 2067 Imaginons notre futur

RMod

Asynchronous

Playground

Page

```
content := 'Content'.
future := [
    content at: 10 put: $b.
 ] future.
future onFailureDo:[: e | e debug ].
self inform:'Before Synchro'
```

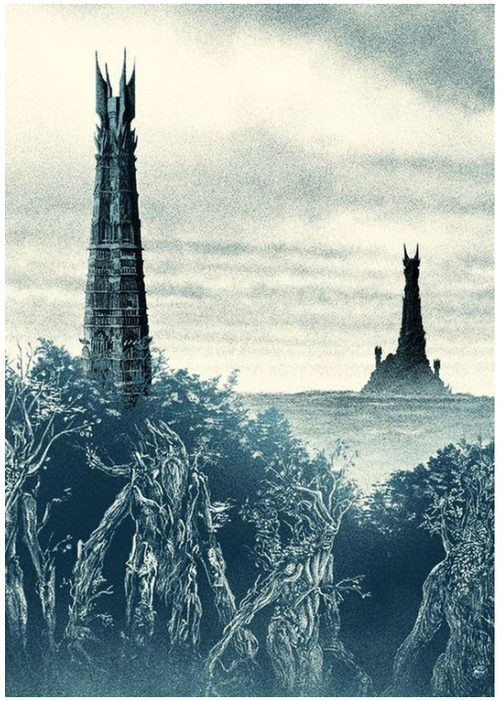
- synchronous
- **asynchronous**
- task combination

Inria inventeurs du monde numérique

50 ANS 1967 2017 2067 Imaginons notre futur

Task combination

- Reinforce sequence
- Transform results
- Trigger new processes



- synchronous
- asynchronous
- **task combination**

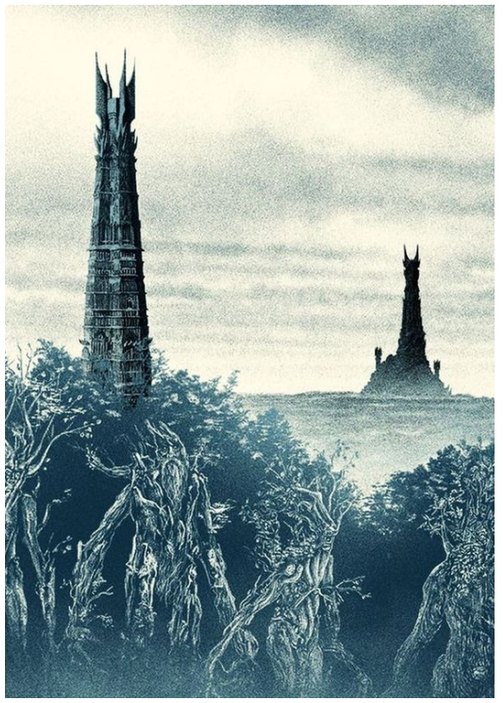
Collect

Run in sequence

Playground

Page

```
defaultMorphFuture := [ Morph new ] future.
redMorphFuture := defaultMorphFuture collect: [ : m | m color: Color red ].
redMorph := redMorphFuture synchronizeTimeout: 1 second.
redMorph openInWorld.
```



- synchronous
- asynchronous
- **task combination**

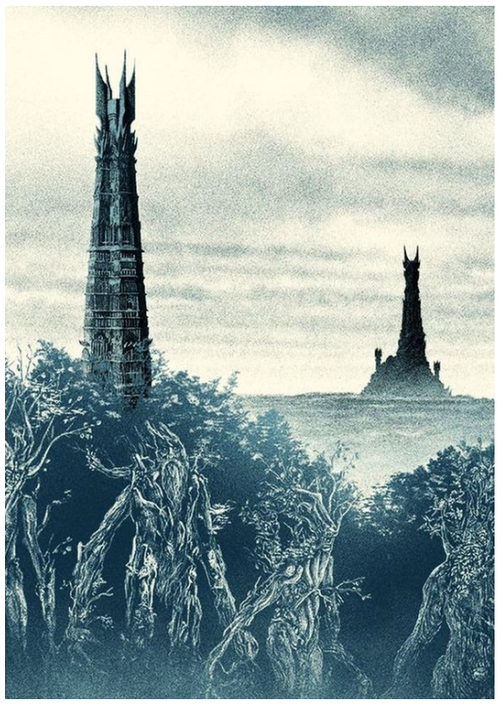
Zip

Run concurrently and join

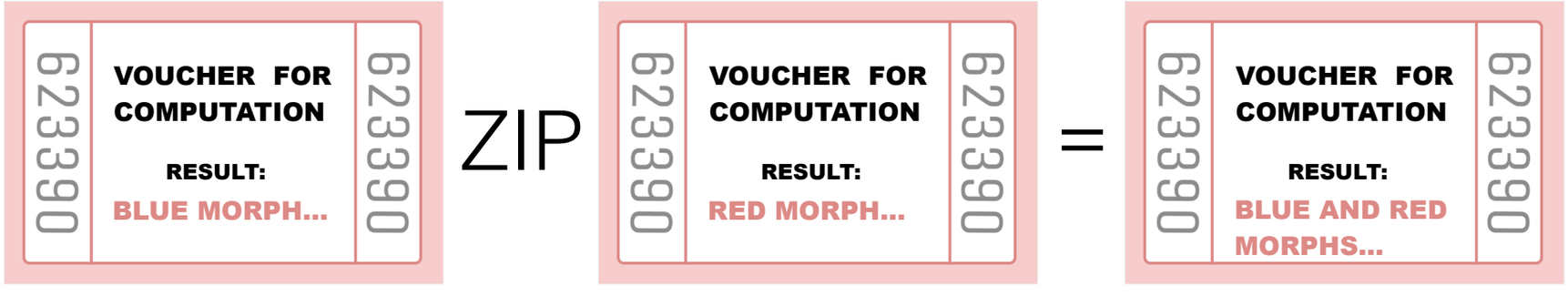
Playground

Page

```
aMorphFuture := [ Morph new ] future.
anOtherMorphFuture := [ Morph new color: Color red; position: 50@18 ; yourself ] future.
zippedMorphFuture := aMorphFuture zip:  anOtherMorphFuture.
morphs := (zippedMorphFuture synchronizeTimeout: 1 second ) .
morphs do: #openInWorld
```



- synchronous
- asynchronous
- **task combination**



inventeurs du monde numérique

50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur

RMod

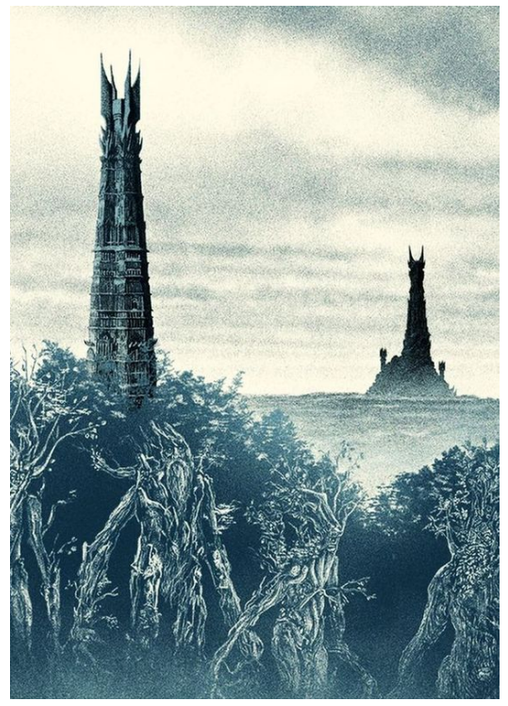
Fallback To

Run concurrently, responds conditionally

Playground

Page

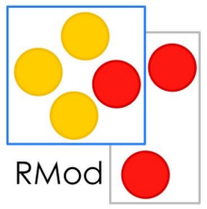
```
futureToFail := [ Error signal  ] future.
futureToFallback := [  'Here a fallback routine ' ] future.
futureThatDoNotFail := futureToFail fallbackTo: futureToFallback.
futureThatDoNotFail synchronizeTimeout: 1 second.
```



- synchronous
- asynchronous
- **task combination**

inventeurs du monde numérique

50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur

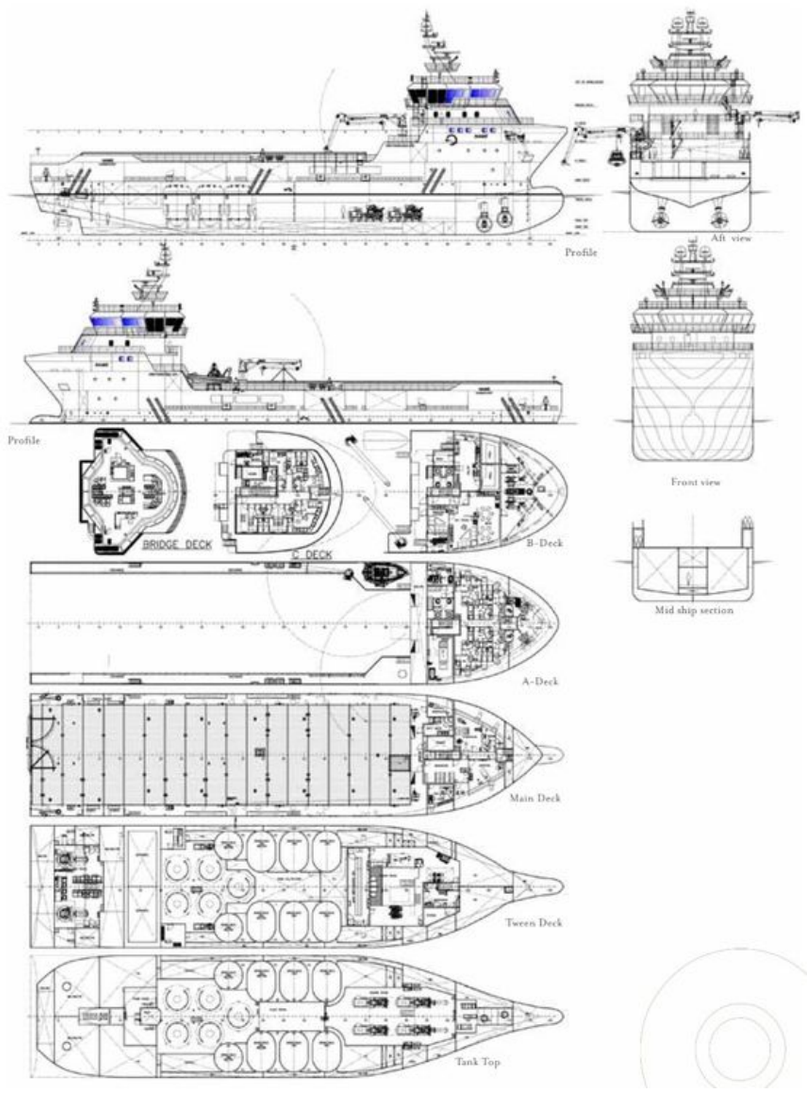

RMod

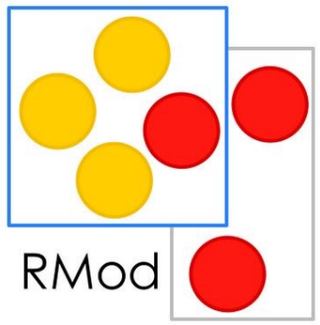
Runners


Inria
inventeurs du monde numérique
50 ANS
1967 › 2017 ›› 2067
Imaginons notre futur

Runners

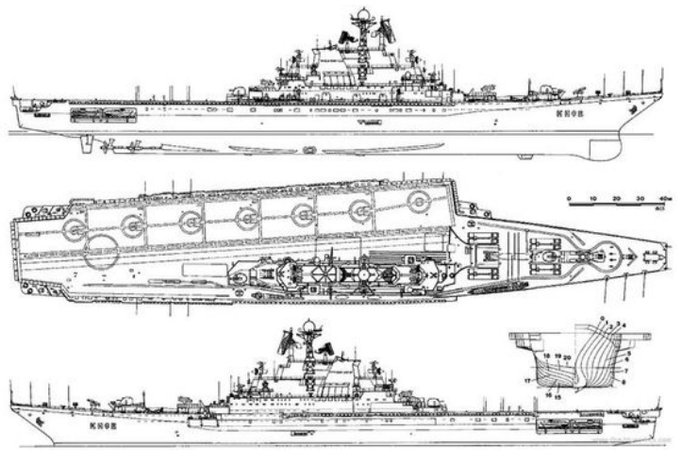
- Objects
- Represent the processing architecture
  1. How
  2. Where
  3. When

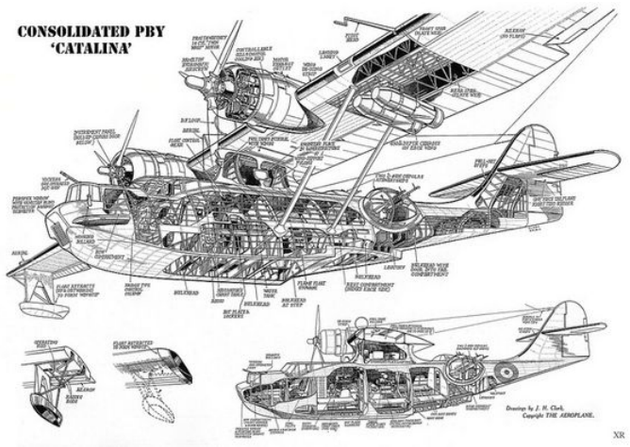


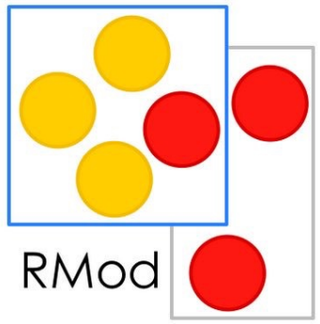


Runners

- Same process
- New process
- Worker
- Worker pool
- Service







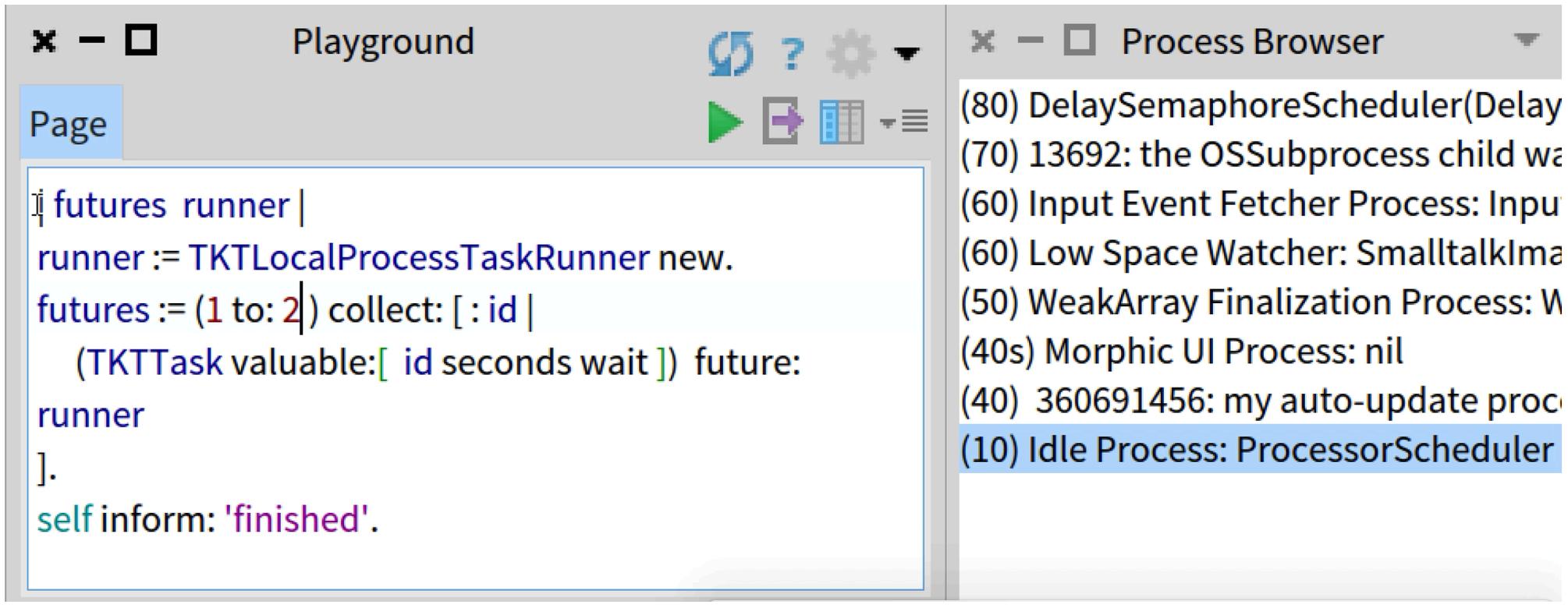
Same Process

- Simple to instantiate
- Non lifecycle control required
- Handy for debugging simple errors

```
| aFuture |

aFuture := (TKTTask valuable: [ " do something " ])
            future: TKTLocalProcessTaskRunner  new.
(TKTTask valuable: [ " do something " ])
            schedule: TKTLocalProcessTaskRunner new.
```

- **Same process**
- UI Runner
- New process
- Worker
- Worker pool
- Service

Same Process

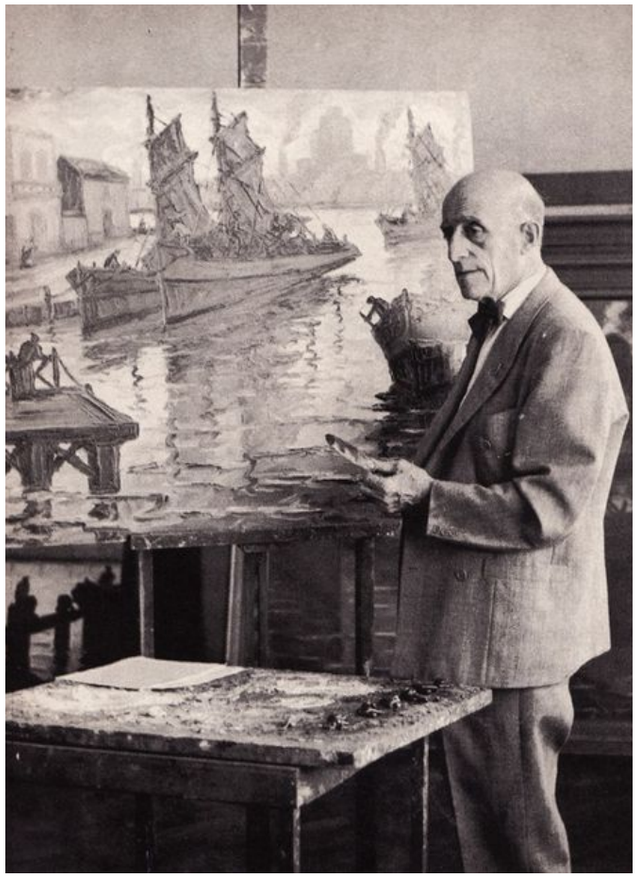


- **Same process**
- UI Runner
- New process
- Worker
- Worker pool
- Service

UI Runner

- Simple to instantiate
- Non lifecycle control required
- Handy for UI tasks

```
| aFuture  |

aFuture := (TKTTask valuable: [ " do something " ])
            future: TKTUIProcessTaskRunner new.
(TKTTask valuable: [ " do something " ])
            schedule: TKTUIProcessTaskRunner new.
```

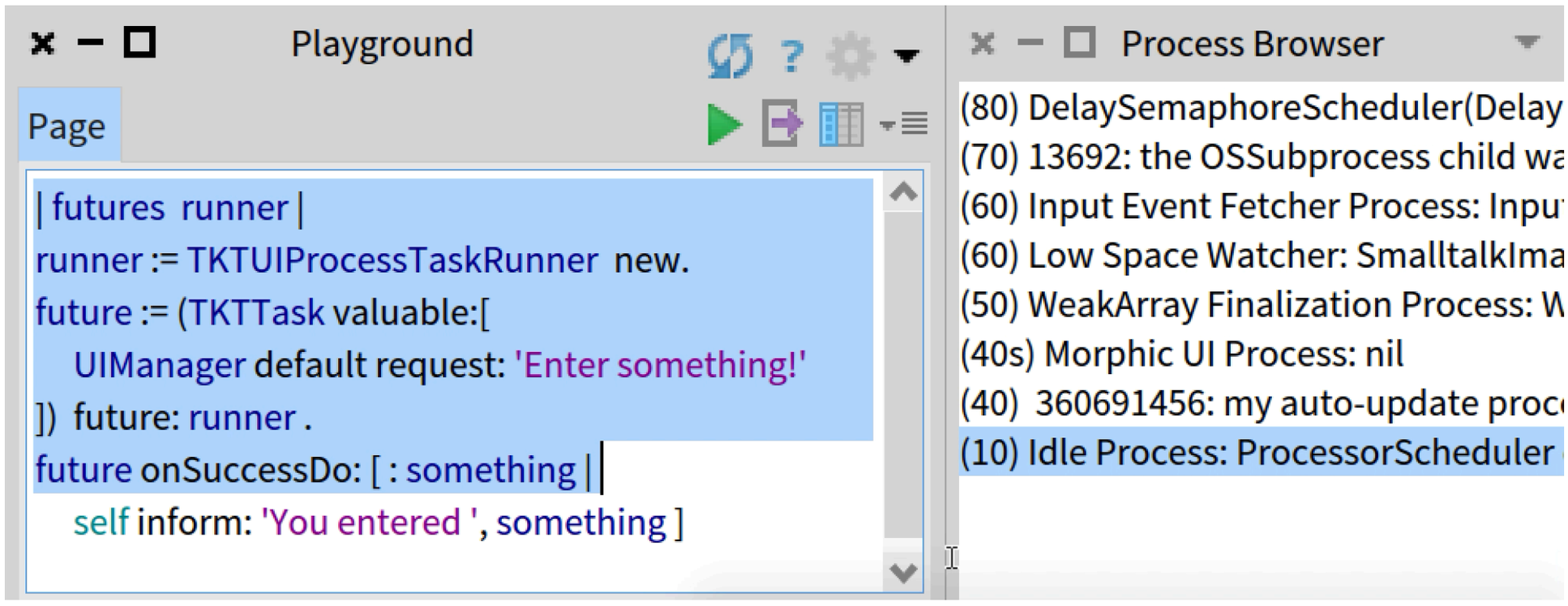


- Same process
- **UI Runner**
- New process
- Worker
- Worker pool
- Service

inventeurs du monde numérique

1967 › 2017 ›› 2067 Imaginons notre futur

UI Runner



- Same process
- **UI Runner**
- New process
- Worker
- Worker pool
- Service

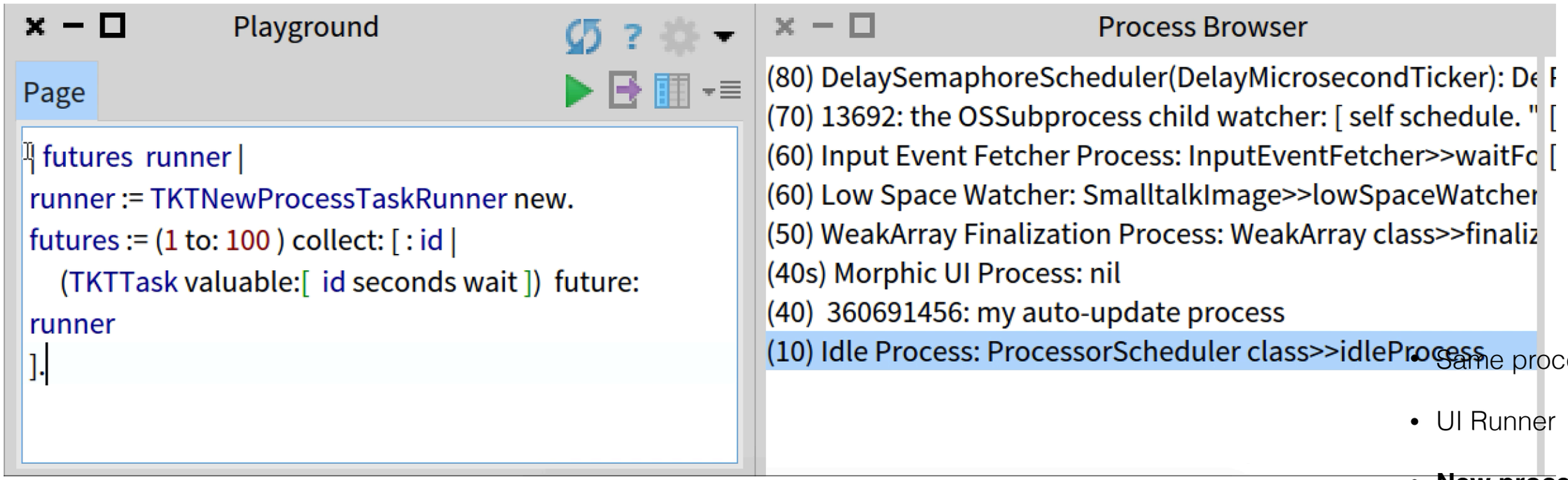
New-Process

- Simple to instantiate
- Lifecycle managed automatically: The process dies after the execution of the task
- Handy for executing tasks at the moment

```
| aFuture |

aFuture := (TKTTask valuable: [ " do something " ])
            future: TKTNewProcessTaskRunner new.
(TKTTask valuable: [ " do something " ])
            schedule: TKTNewProcessTaskRunner new.
```

- Same process
- UI Runner
- **New process**
- Worker
- Worker pool
- Service

New-Process



- Same process
- UI Runner
- **New process**
- Worker
- Worker pool
- Service

inventeurs du monde numérique

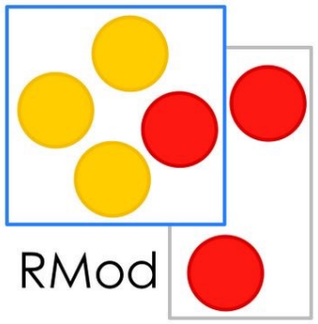
50 ANS 1967 2017 2067 Imaginons notre futur

Worker

- Instantiation requires to hold the worker reference
- Lifecycle managed by garbage collection & Watch dog
- Handy for reusing the same process

```
| aFuture worker |
worker := TKTWorker new.
worker queue: AtomicSharedQueue new.
worker start.

aFuture := (TKTTask valuable: [ " do something " ])
            future: worker.
(TKTTask valuable: [ " do something " ])
            schedule: worker.
```

- Same process
- UI Runner
- New process
- **Worker**
- Worker pool
- Service



Worker

Playground

Page

```
|worker future |
worker := TKTWorker new.
worker queue: AtomicSharedQueue new.
worker start.
future := worker future: (TKTTask valuable:[ 'Here a really complex task ']).
worker schedule: [ self inform: 'It was not magic! :) ' ].
future synchronizeTimeout: 1 second.
```

- Same process
- UI Runner
- New process
- **Worker**
- Worker pool
- Service

Worker-Pool

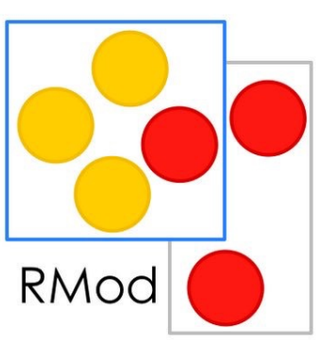
- Instantiation requires to hold the worker reference
- Lifecycle managed by garbage collection & Watch dog
- Handy for reusing the same process and control the system's load

```
| aFuture  pool |

pool := TKTCommonQueueWorkerPool new.
pool poolMaxSize: 4. " default value "

aFuture := (TKTTask valuable: [ " do something " ])
            future: pool.
(TKTTask valuable: [ " do something " ])
            schedule: pool.
```

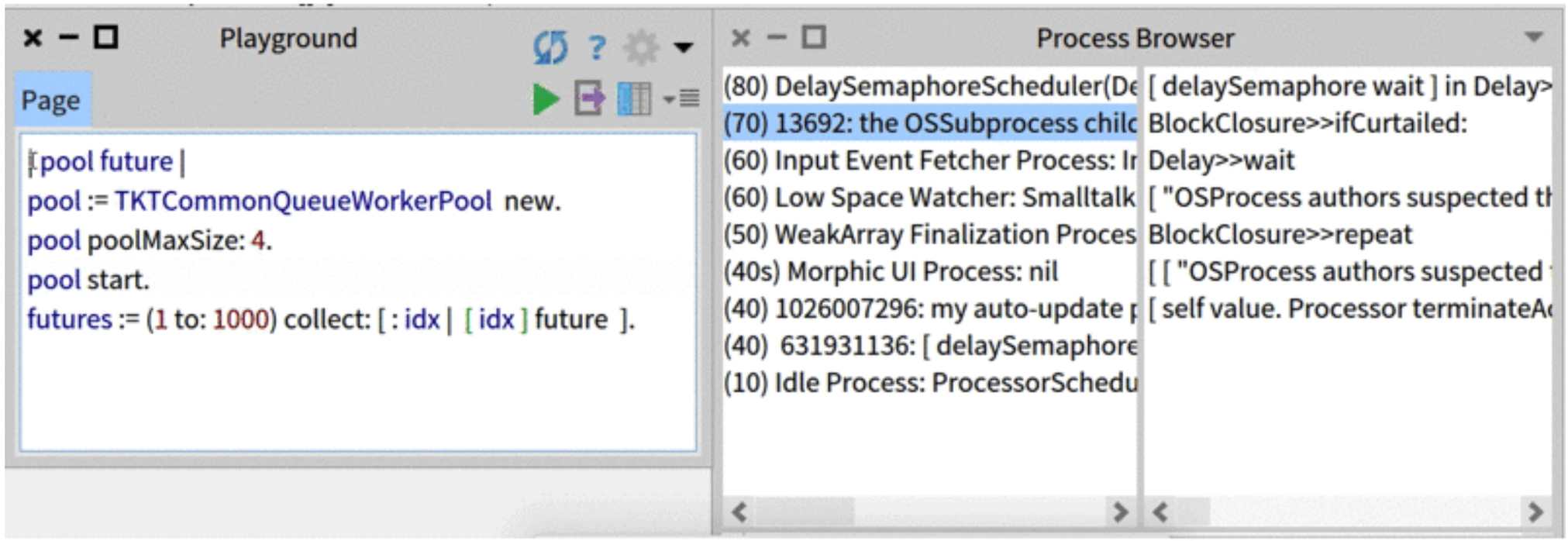
- Same process
- UI Runner
- New process
- Worker
- **Worker pool**
- Service

inventeurs du monde numérique

50 ANS 1967 › 2017 › 2067 Imaginons notre futur



Worker-Pool



- Same process
- UI Runner
- New process
- Worker
- **Worker pool**
- Service

inventeurs du monde numérique

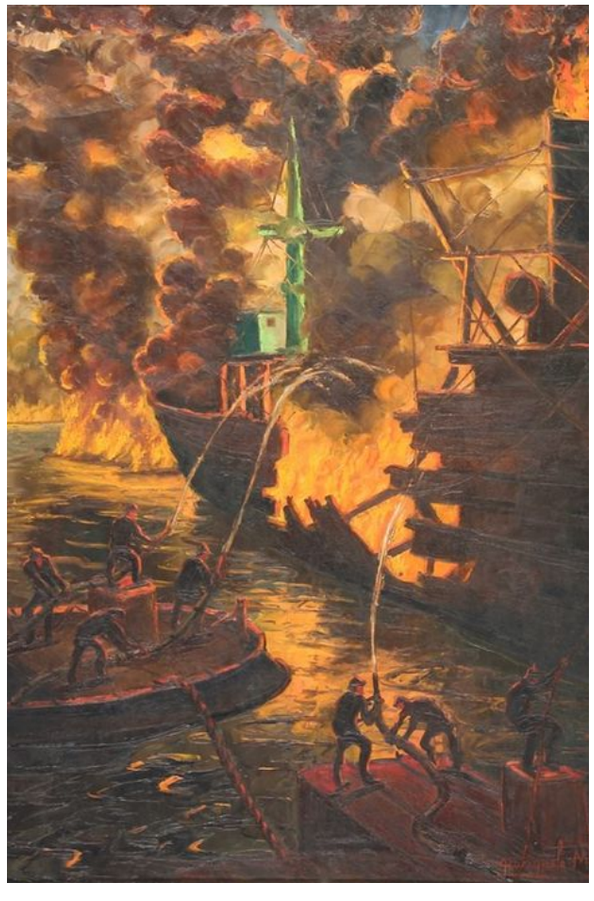
Service

- Instantiation requires to set the task before starting the service, and also requires an unique name

- Lifecycle managed by the user by start/stop/restart

- Handy for providing services/daemons

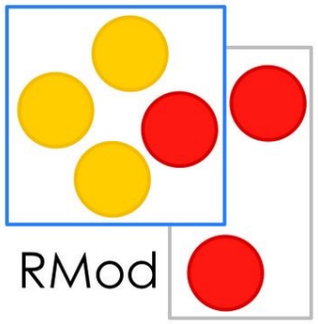
```
| service  |

service := TKTParameterizableService new.
service stepDelay: 500 milliSeconds.
service name: 'Unique-Service-Name'.
service step: [ self inform: ' Tick ' ].
service start.
```



- Same process
- UI Runner
- New process
- Worker
- Worker pool
- **Service**

inventeurs du monde numérique

50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur



Service

Playground

Page

```
service := TKTParameterizableService new.
service step: [ self inform: 'Stepping' ].
service stepDelay: 1000 milliSeconds.
service name: 'UniqueNameForService'.
service start.

(TKTConfiguration serviceManager
findServiceNamed: 'UniqueNameForService') stop.
```

Process Br

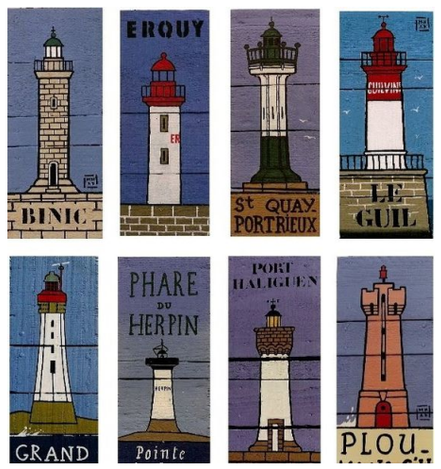
```
(80) DelaySemaphoreSchedule
(60) Input Event Fetcher Proces
(60) Low Space Watcher: Smallt
(50) WeakArray Finalization Pro
(40)  360691456: my auto-upda
(40)  65476352: [ delaySemaph
(40s) Morphic UI Process: nil
(10) Idle Process: ProcessorSch
```

- Same process
- UI Runner
- New process
- Worker
- Worker pool
- **Service**

Appendix 1:
Extensions

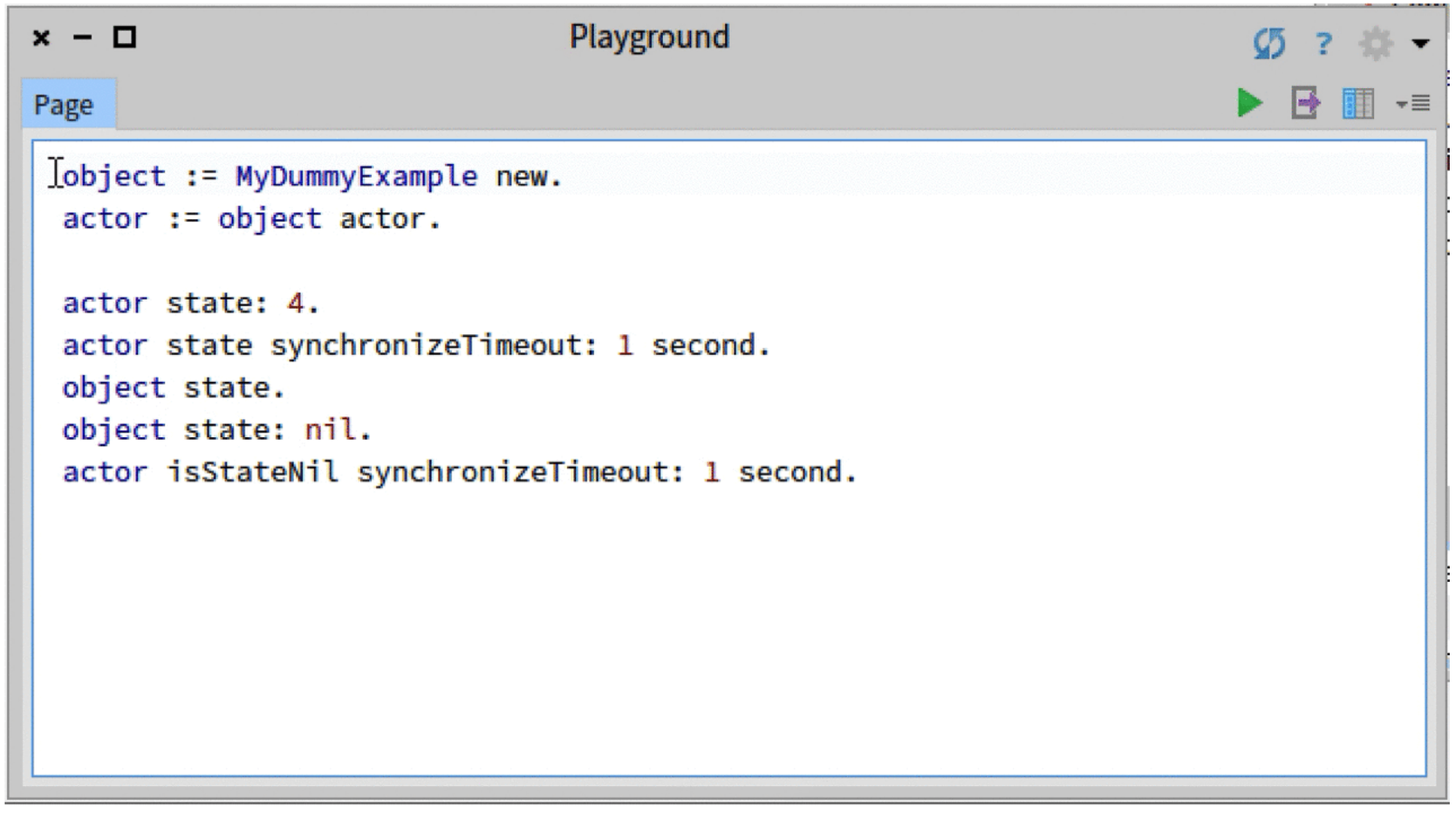
TaskIt Extensions: ActIt

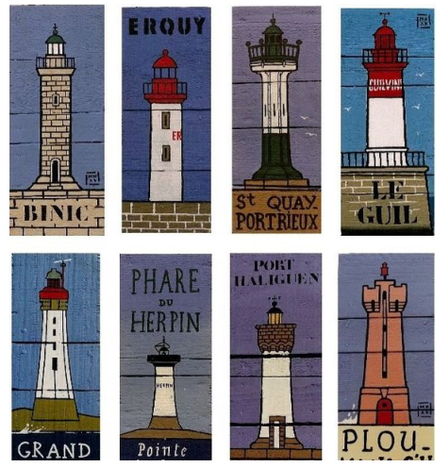
- Provides an ActTalk inspired implementation
- Provides processing flavours
  - Worker
  - UI
  - Same process



RMod

TaskIt Extensions: ActIt





Inria inventeurs du monde numérique

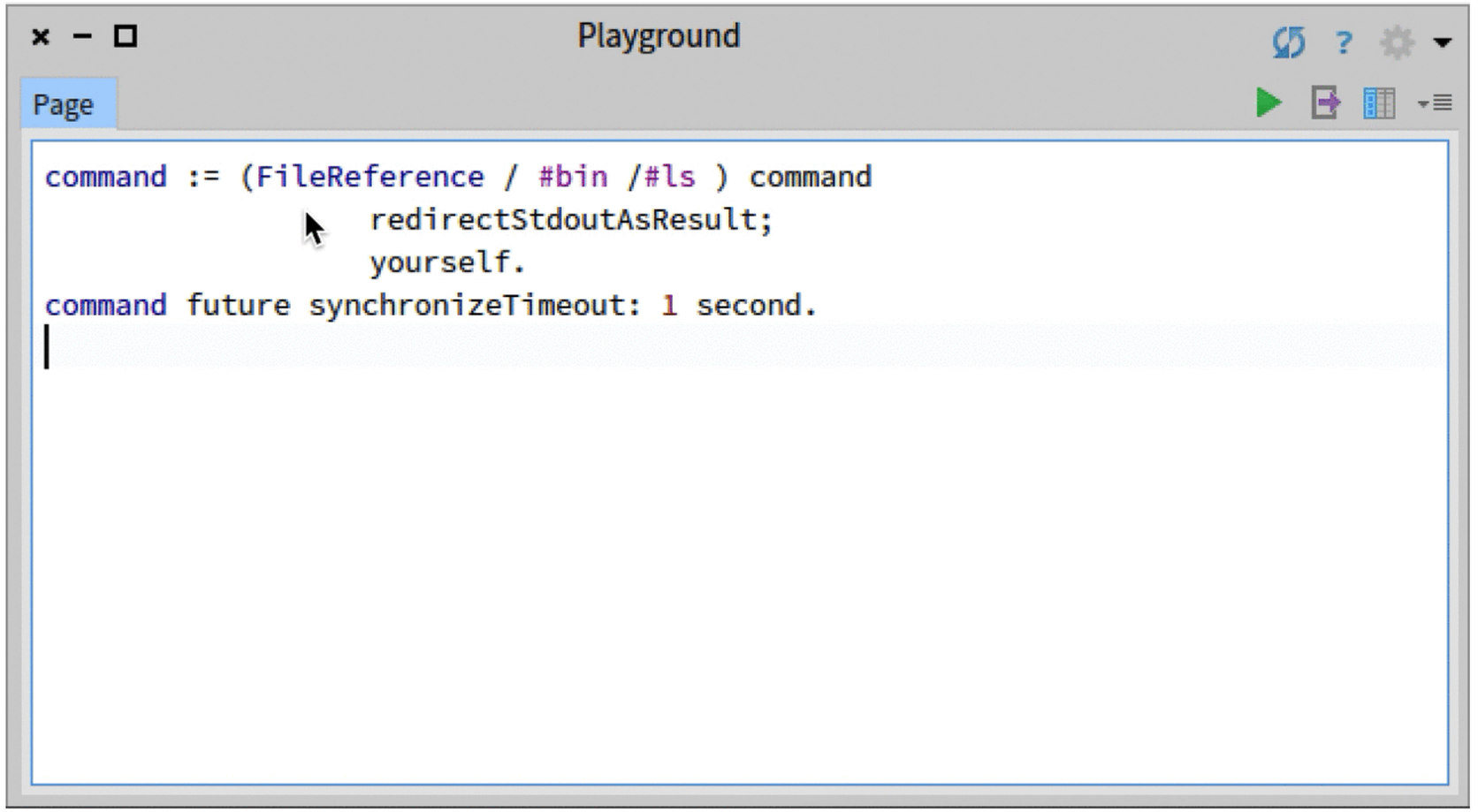
50 ANS 1967 >> 2017 >> 2067 Imaginons notre futur

TaskIt Extensions: Shell

- Provides a new kind of task
- Is based on OS-Subprocess
- Allows to transform standard output into results

RMod

TaskIt Extensions: Shell

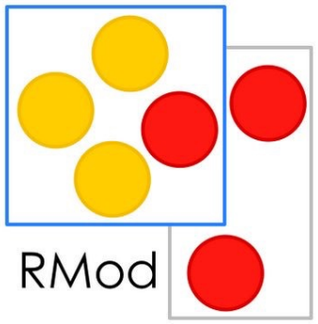


Inria
inventeurs du monde numérique
50 ANS
1967 › 2017 ›› 2067
Imaginons notre futur

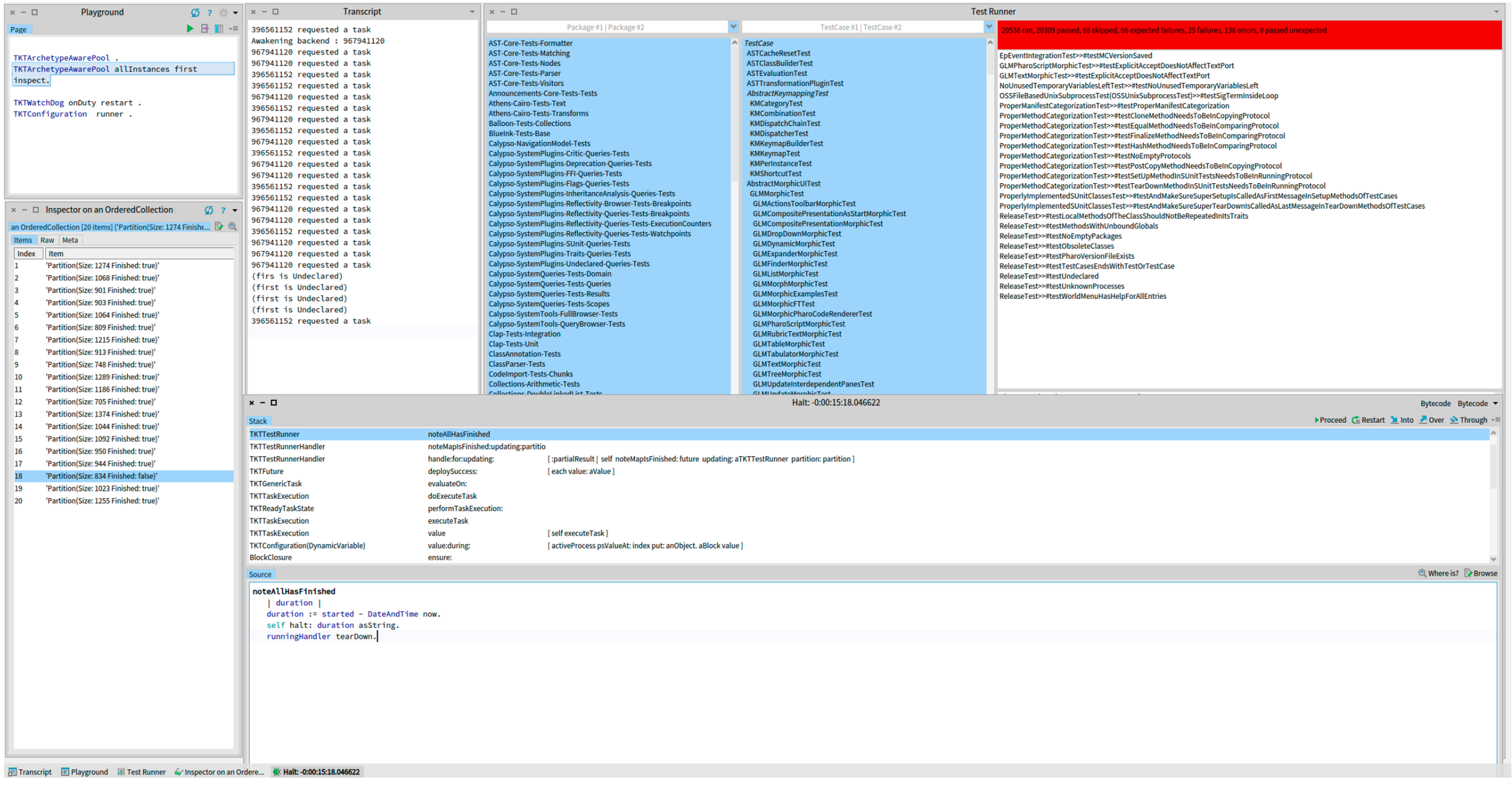
TaskIt Extensions: ForkIt

- Master / slave architecture
- Reuse most of the task it and task it shell architecture
- Alpha state, but improving fast

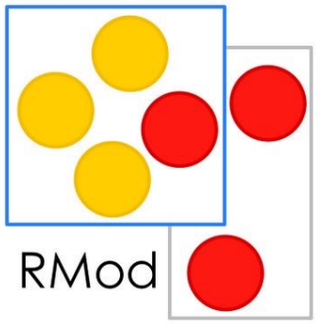
inventeurs du monde numérique


RMod

ForkIt


Playground
Page
TKTArchetypeAwarePool .
TKTArchetypeAwarePool allInstances first
inspect.
TKTWatchDog onDuty restart .
TKTConfiguration runner .
Transcript
396561152 requested a task
Awakening backend : 967941120
967941120 requested a task
967941120 requested a task
396561152 requested a task
396561152 requested a task
967941120 requested a task
396561152 requested a task
967941120 requested a task
396561152 requested a task
967941120 requested a task
396561152 requested a task
967941120 requested a task
967941120 requested a task
396561152 requested a task
967941120 requested a task
967941120 requested a task
396561152 requested a task
967941120 requested a task
967941120 requested a task
396561152 requested a task
(firs is Undeclared)
(first is Undeclared)
(first is Undeclared)
(first is Undeclared)
396561152 requested a task
Test Runner
Package #1 | Package #2
TestCase #1 | TestCase #2
20538 ran, 20309 passed, 68 skipped, 66 expected failures, 25 failures, 138 errors, 0 passed unexpected
AST-Core-Tests-Formatter
AST-Core-Tests-Matching
AST-Core-Tests-Nodes
AST-Core-Tests-Parser
AST-Core-Tests-Visitors
Announcements-Core-Tests-Tests
Athens-Cairo-Tests-Text
Athens-Cairo-Tests-Transforms
Balloon-Tests-Collections
BlueInk-Tests-Base
Calypso-NavigationModel-Tests
Calypso-SystemPlugins-Critic-Queries-Tests
Calypso-SystemPlugins-Deprecation-Queries-Tests
Calypso-SystemPlugins-FFI-Queries-Tests
Calypso-SystemPlugins-Flags-Queries-Tests
Calypso-SystemPlugins-InheritanceAnalysis-Queries-Tests
Calypso-SystemPlugins-Reflectivity-Browser-Tests-Breakpoints
Calypso-SystemPlugins-Reflectivity-Queries-Tests-Breakpoints
Calypso-SystemPlugins-Reflectivity-Queries-Tests-ExecutionCounters
Calypso-SystemPlugins-Reflectivity-Queries-Tests-Watchpoints
Calypso-SystemPlugins-SUnit-Queries-Tests
Calypso-SystemPlugins-Traits-Queries-Tests
Calypso-SystemPlugins-Undeclared-Queries-Tests
Calypso-SystemQueries-Tests-Domain
Calypso-SystemQueries-Tests-Queries
Calypso-SystemQueries-Tests-Results
Calypso-SystemQueries-Tests-Scopes
Calypso-SystemTools-FullBrowser-Tests
Calypso-SystemTools-QueryBrowser-Tests
Clap-Tests-Integration
Clap-Tests-Unit
ClassAnnotation-Tests
ClassParser-Tests
CodeImport-Tests-Chunks
Collections-Arithmetic-Tests
TestCase
ASTCacheResetTest
ASTClassBuilderTest
ASTEvaluationTest
ASTTransformationPluginTest
AbstractKeymappingTest
KMCategoryTest
KMCombinationTest
KMDispatchChainTest
KMDispatcherTest
KMKeymapBuilderTest
KMKeymapTest
KMPerInstanceTest
KMShortcutTest
AbstractMorphicUITest
GLMMorphicTest
GLMActionsToolbarMorphicTest
GLMCompositePresentationAsStartMorphicTest
GLMCompositePresentationMorphicTest
GLMDropDownMorphicTest
GLMDynamicMorphicTest
GLMExpanderMorphicTest
GLMFinderMorphicTest
GLMListMorphicTest
GLMMorphMorphicTest
GLMMorphicExamplesTest
GLMMorphicFTTest
GLMMorphicPharoCodeRendererTest
GLMPharoScriptMorphicTest
GLMRubricTextMorphicTest
GLMTableMorphicTest
GLMTabulatorMorphicTest
GLMTextMorphicTest
GLMTreeMorphicTest
GLMUpdateInterdependentPanesTest
EpEventIntegrationTest>>#testMCVersionSaved
GLMPharoScriptMorphicTest>>#testExplicitAcceptDoesNotAffectTextPort
GLMTextMorphicTest>>#testExplicitAcceptDoesNotAffectTextPort
NoUnusedTemporaryVariablesLeftTest>>#testNoUnusedTemporaryVariablesLeft
OSSFileBasedUnixSubprocessTest(OSSUnixSubprocessTest)>>#testSigTermInsideLoop
ProperManifestCategorizationTest>>#testProperManifestCategorization
ProperMethodCategorizationTest>>#testCloneMethodNeedsToBeInCopyingProtocol
ProperMethodCategorizationTest>>#testEqualMethodNeedsToBeInComparingProtocol
ProperMethodCategorizationTest>>#testFinalizeMethodNeedsToBeInComparingProtocol
ProperMethodCategorizationTest>>#testHashMethodNeedsToBeInComparingProtocol
ProperMethodCategorizationTest>>#testNoEmptyProtocols
ProperMethodCategorizationTest>>#testPostCopyMethodNeedsToBeInCopyingProtocol
ProperMethodCategorizationTest>>#testSetUpMethodInSUnitTestsNeedsToBeInRunningProtocol
ProperMethodCategorizationTest>>#testTearDownMethodInSUnitTestsNeedsToBeInRunningProtocol
ProperlyImplementedSUnitClassesTest>>#testAndMakeSureSuperSetupIsCalledAsFirstMessageInSetupMethodsOfTestCases
ProperlyImplementedSUnitClassesTest>>#testAndMakeSureSuperTearDownIsCalledAsLastMessageInTearDownMethodsOfTestCases
ReleaseTest>>#testLocalMethodsOfTheClassShouldNotBeRepeatedInItsTraits
ReleaseTest>>#testMethodsWithUnboundGlobals
ReleaseTest>>#testNoEmptyPackages
ReleaseTest>>#testObsoleteClasses
ReleaseTest>>#testPharoVersionFileExists
ReleaseTest>>#testTestCasesEndsWithTestOrTestCase
ReleaseTest>>#testUndeclared
ReleaseTest>>#testUnknownProcesses
ReleaseTest>>#testWorldMenuHasHelpForAllEntries
Inspector on an OrderedCollection
an OrderedCollection [20 items] ('Partition(Size: 1274 Finishe...
Items
Raw
Meta
Index
Item
1 'Partition(Size: 1274 Finished: true)'
2 'Partition(Size: 1068 Finished: true)'
3 'Partition(Size: 901 Finished: true)'
4 'Partition(Size: 903 Finished: true)'
5 'Partition(Size: 1064 Finished: true)'
6 'Partition(Size: 809 Finished: true)'
7 'Partition(Size: 1215 Finished: true)'
8 'Partition(Size: 913 Finished: true)'
9 'Partition(Size: 748 Finished: true)'
10 'Partition(Size: 1289 Finished: true)'
11 'Partition(Size: 1186 Finished: true)'
12 'Partition(Size: 705 Finished: true)'
13 'Partition(Size: 1374 Finished: true)'
14 'Partition(Size: 1044 Finished: true)'
15 'Partition(Size: 1092 Finished: true)'
16 'Partition(Size: 950 Finished: true)'
17 'Partition(Size: 944 Finished: true)'
18 'Partition(Size: 834 Finished: false)'
19 'Partition(Size: 1023 Finished: true)'
20 'Partition(Size: 1255 Finished: true)'
Halt: -0:00:15:18.046622
Bytecode
Stack
Proceed
Restart
Into
Over
Through
TKTTestRunner noteAllHasFinished
TKTTestRunnerHandler noteMapIsFinished:updating:partitio
TKTTestRunnerHandler handle:for:updating: [:partialResult | self noteMapIsFinished: future updating: aTKTTestRunner partition: partition]
TKTFuture deploySuccess: [each value: aValue]
TKTGenericTask evaluateOn:
TKTTaskExecution doExecuteTask
TKTReadyTaskState performTaskExecution:
TKTTaskExecution executeTask
TKTTaskExecution value [self executeTask]
TKTConfiguration(DynamicVariable) value:during: [activeProcess psValueAt: index put: anObject. aBlock value]
BlockClosure ensure:
Source
Where is?
Browse
noteAllHasFinished
| duration |
duration := started - DateAndTime now.
self halt: duration asString.
runningHandler tearDown.
Transcript
Playground
Test Runner
Inspector on an Ordere...
Halt: -0:00:15:18.046622


Inria
inventeurs du monde numérique
50 ANS
1967 » 2017 » 2067
Imaginons notre futur



TaskIt Extensions: ForkIt

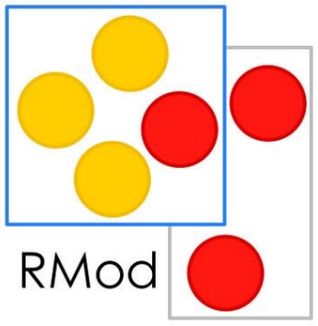
- Provides an extension for building images
- Provides a new runner: Remote Worker

TaskIt Extensions: ForkIt

- Working on adapting to the industrial standards

- Process communication Message queue (RabbitMq)

- Building process (Puppet/ Vagrant/Others / not yet decided)



Thanks :)

- Synchronise different tasks by using powerful and highly tested **futures**
- Delegate the lifecycle control to specialised **runners**, according with your domain
- Control the load of your image by using **pools** of processes
- Boost your productivity in concurrency by using a **mature** library used for user interaction and robotic communication
- https://github.com/sbragagnolo/taskit

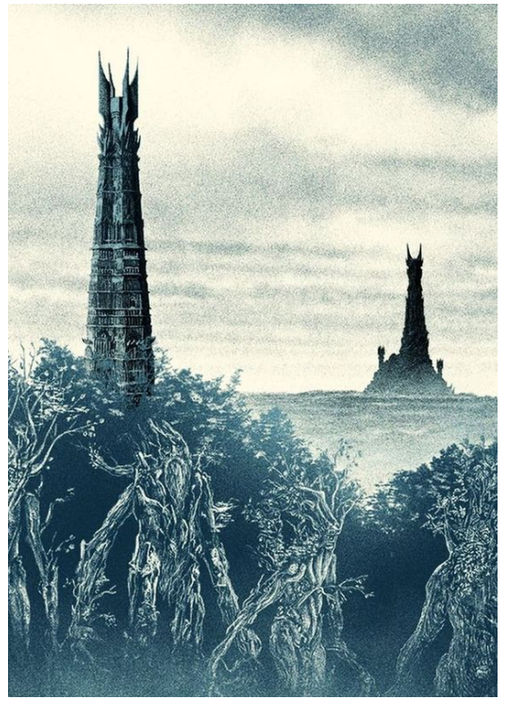
Appendix 2: All the combinators

Combinations: Collect

Playground

Page

```
| aFuture |
aFuture := [ 2 + 3 ] future.
(aFuture collect: [ :number | number factorial ])
    onSuccessDo: [ :result | self inform: result asString ].
```



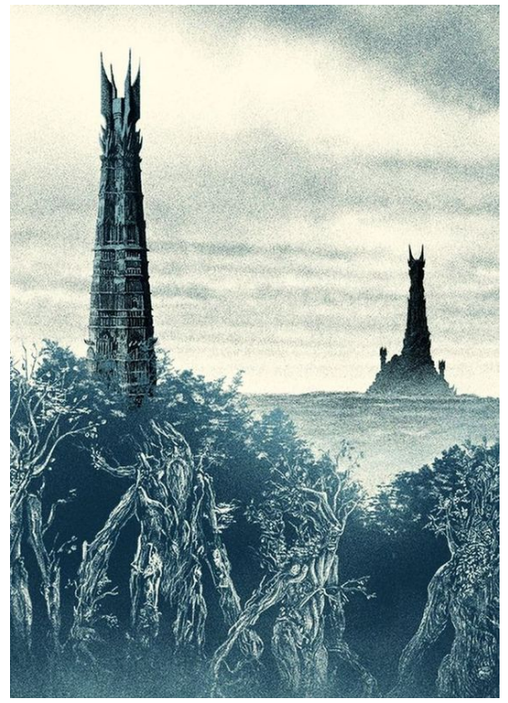
- synchronous
- asynchronous
- **task combination**

Combinations: Select

Playground

Page

```
future := [ 2 + 3 ] future.
(future select: [ :number | number even ])
    onSuccessDo: [ :result | self inform:  result asString ];
    onFailureDo: [ :error | self inform: error asString  ].
```

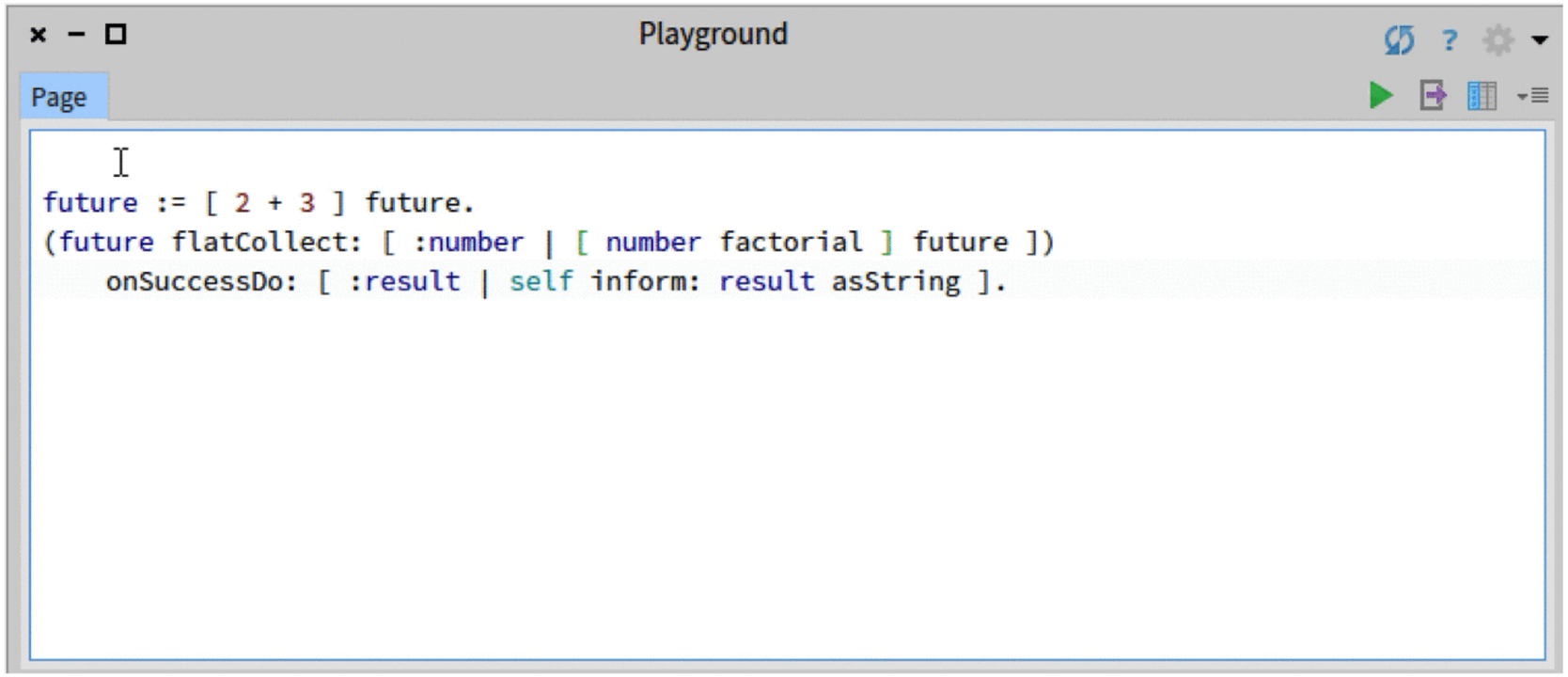


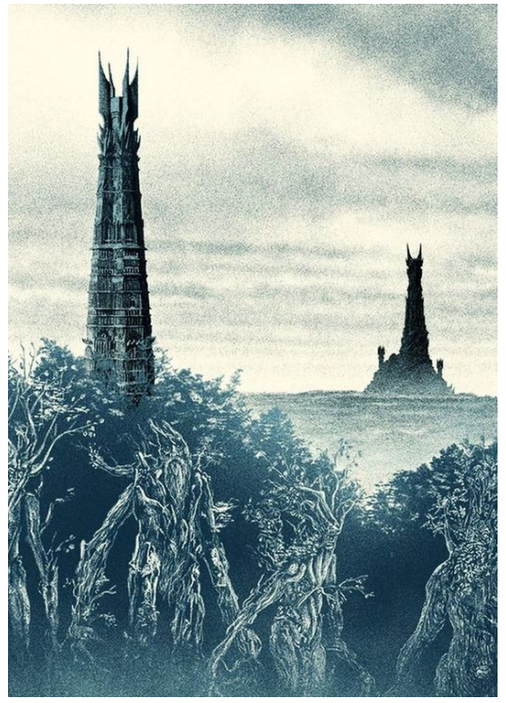
- synchronous
- asynchronous
- **task combination**

inventeurs du monde numérique

50 ANS 1967 » 2017 » 2067 Imaginons notre futur

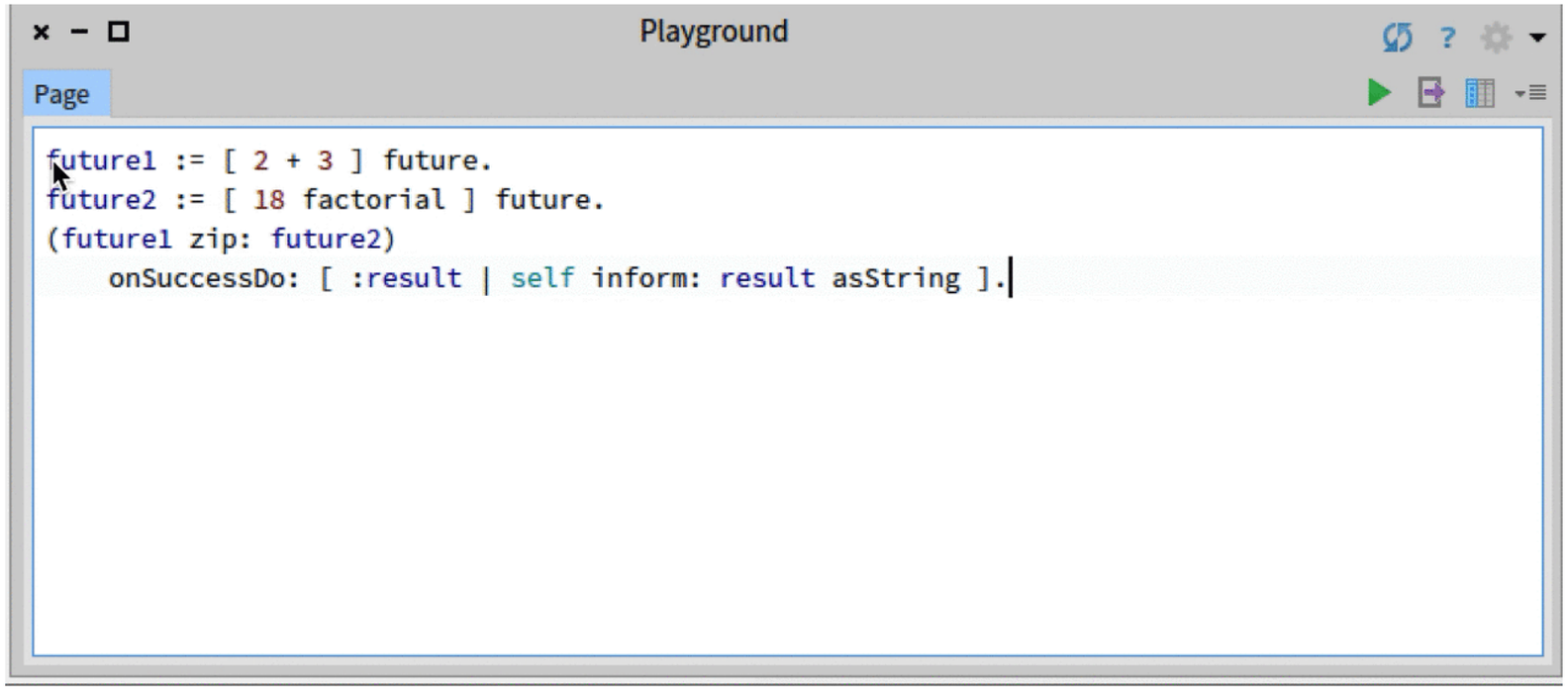
Combinations: Flat Collect

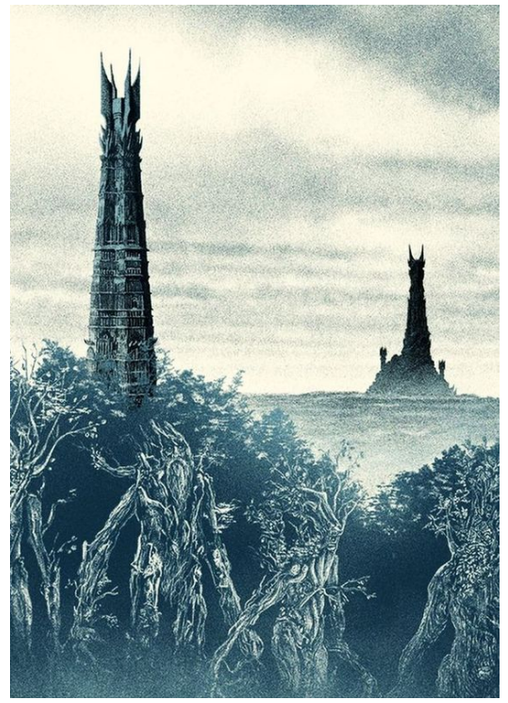




- synchronous
- asynchronous
- **task combination**

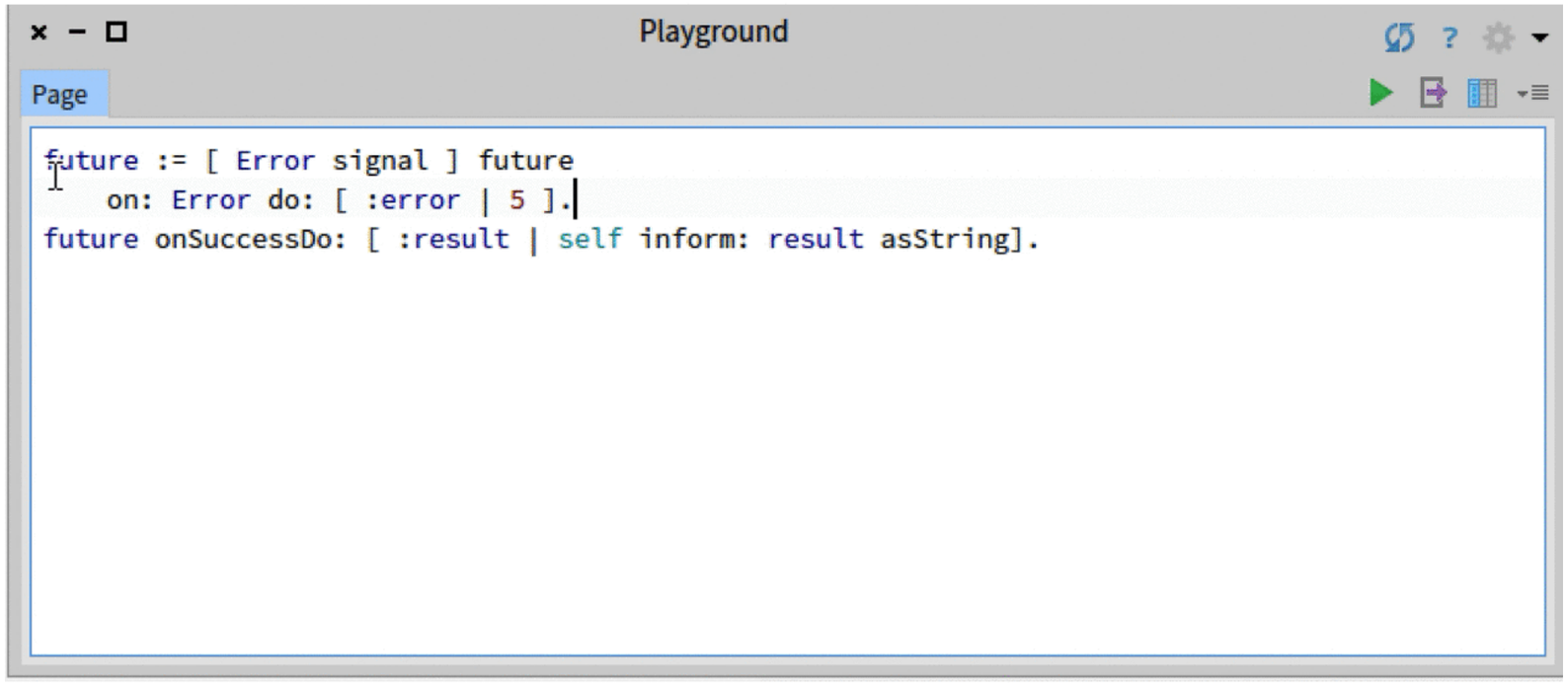
Combinations: Zip

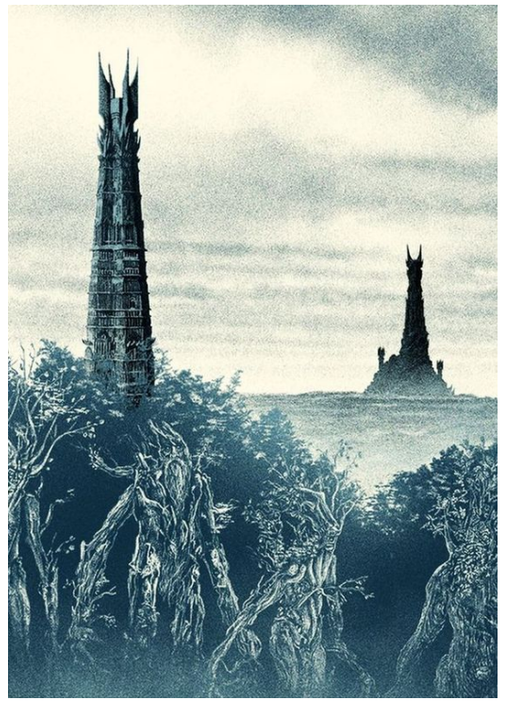




- synchronous
- asynchronous
- **task combination**

Combinations: On-Do





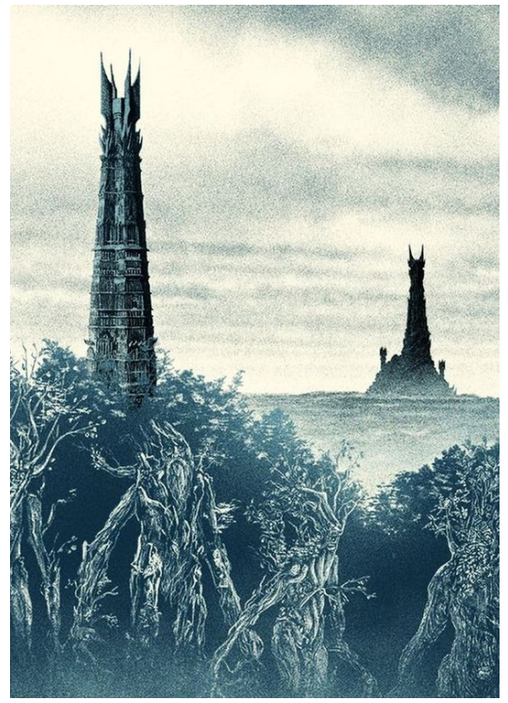
- synchronous
- asynchronous
- **task combination**

Combinations: Fallback To

Playground

Page

```
failFuture := [ Error signal ] future.
successFuture := [ 1 + 1 ] future.
(failFuture fallbackTo: successFuture)
    onSuccessDo: [ :result |self inform: result asString ].
```



- synchronous
- asynchronous
- **task combination**

inventeurs du monde numérique

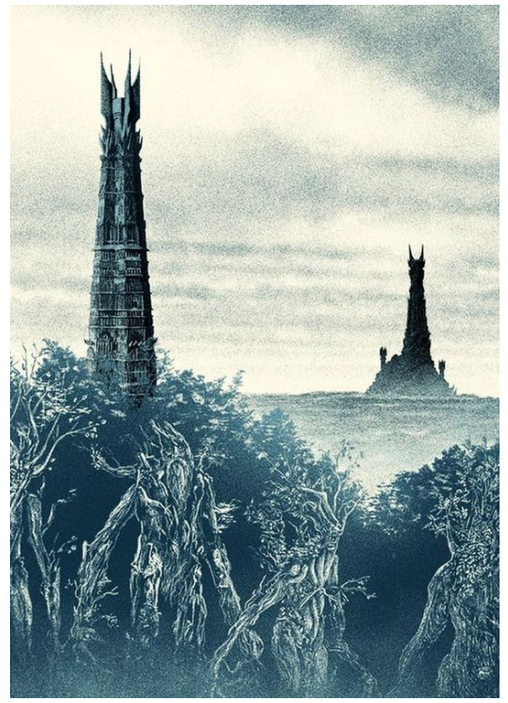
50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur

Combinations: First complete

Playground

Page

```
failFuture := [ 2 second wait. 20 ] future.
successFuture := [ 1 second wait. 1 + 1 ] future.
(failFuture firstCompleteOf: successFuture)
    onSuccessDo: [ :result | self inform: result asString ];
    onFailureDo: [ :error | self inform: error asString  ].
```



- synchronous
- asynchronous
- **task combination**

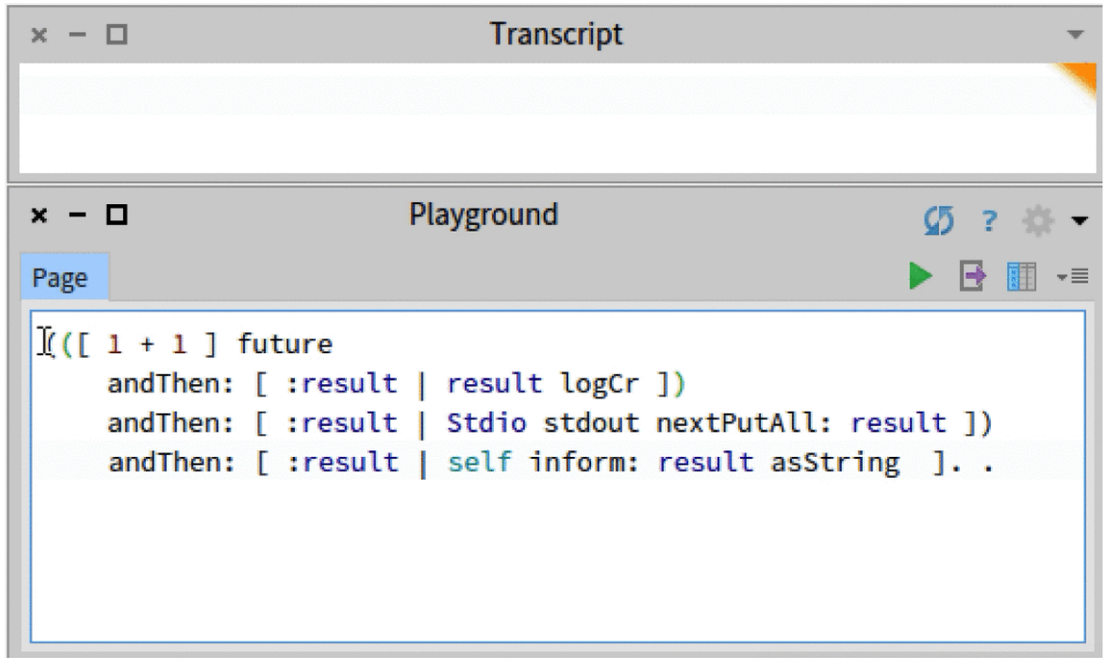
inventeurs du monde numérique

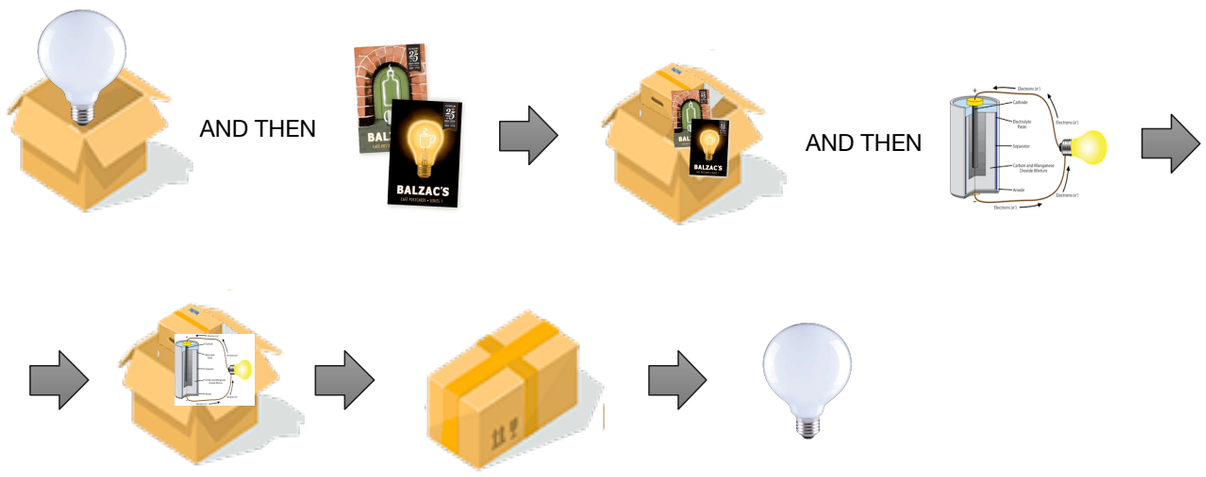
50 ANS 1967 » 2017 » 2067 Imaginons notre futur

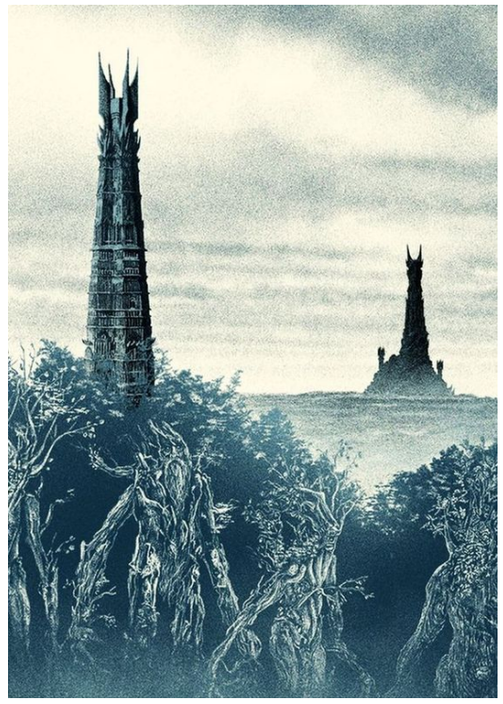
RMod

And then

Run in sequence







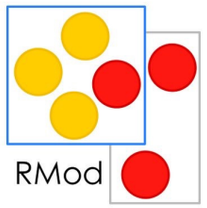
- synchronous
- asynchronous
- **task combination**

inventeurs du monde numérique

50 ANS 1967 › 2017 ›› 2067 Imaginons notre futur

Concurrence

- From old french “concurrencé”
  - Co-occurrence (Happening simultaneously)
  - Competition

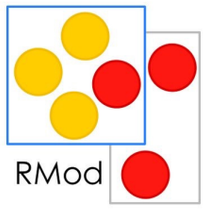


Concurrence (CS)

Multiple computations happening at the same time, in the same system

or

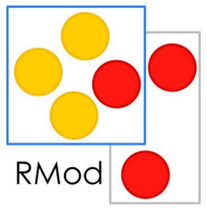
Ability of different parts or units of a program, algorithm, or problem to be executed out-of-order or in partial order, without affecting the final outcome.



Concurrence (CS)

Why should we?

- Not blocking the user
- Enhancing the resources usage
  - Doing things in background (or while the CPU is idle)
  - Managing many time-consuming operations simultaneously (I/O)



Concurrency

- Sharing resources
- Maximising the overall performance, in detriment of the particular or individual performance