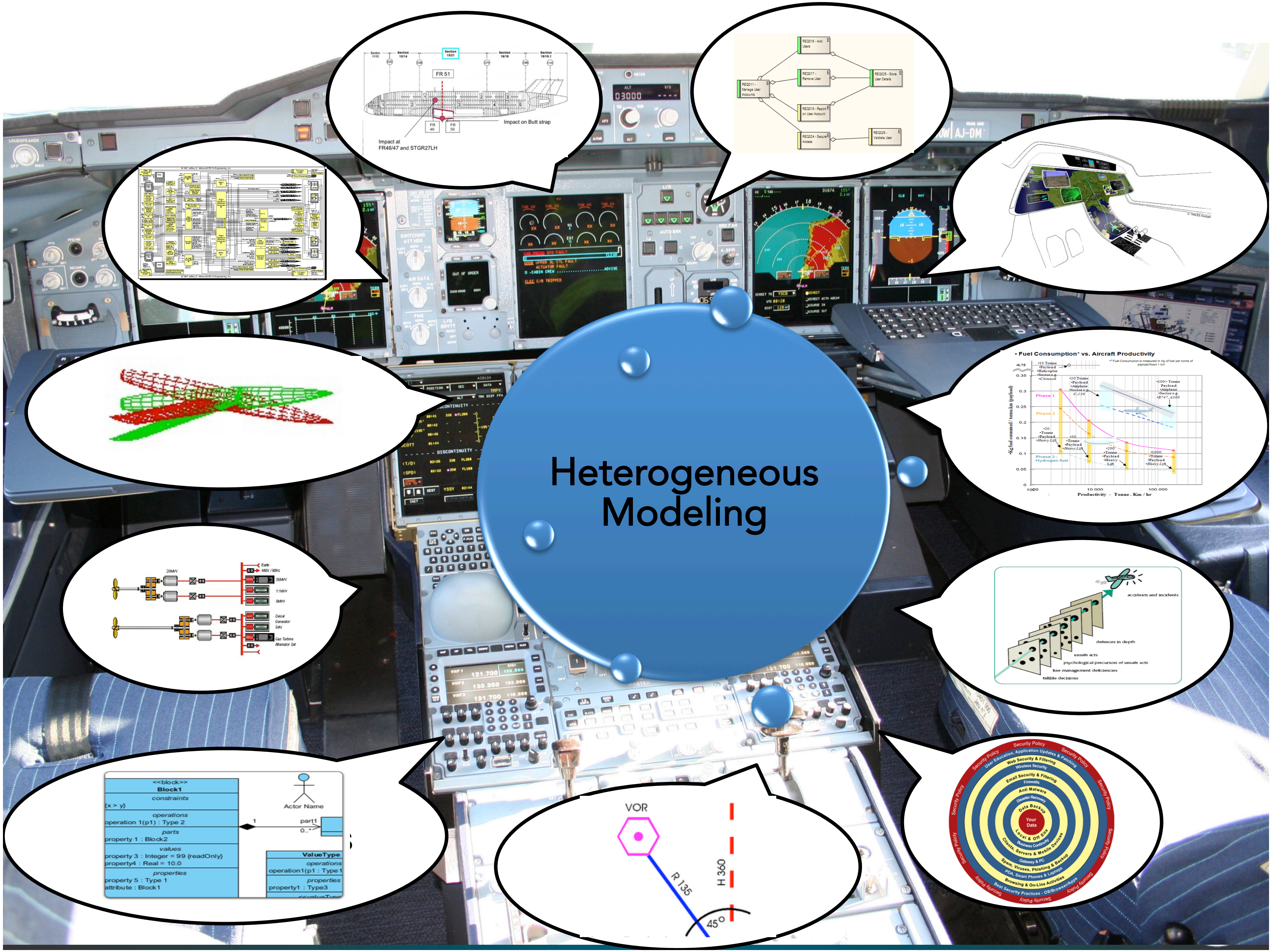
Schedule your micro-services on Docker Swarm with a Sirius-based workflow designer

Olivier Barais, Benoit Combemale,
Cédric Brun, Johann Bourcier,
David Bromberg

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTEMES ALÉATOIRES

Heterogeneous and Distributed
Services for the Future Computing Continuum

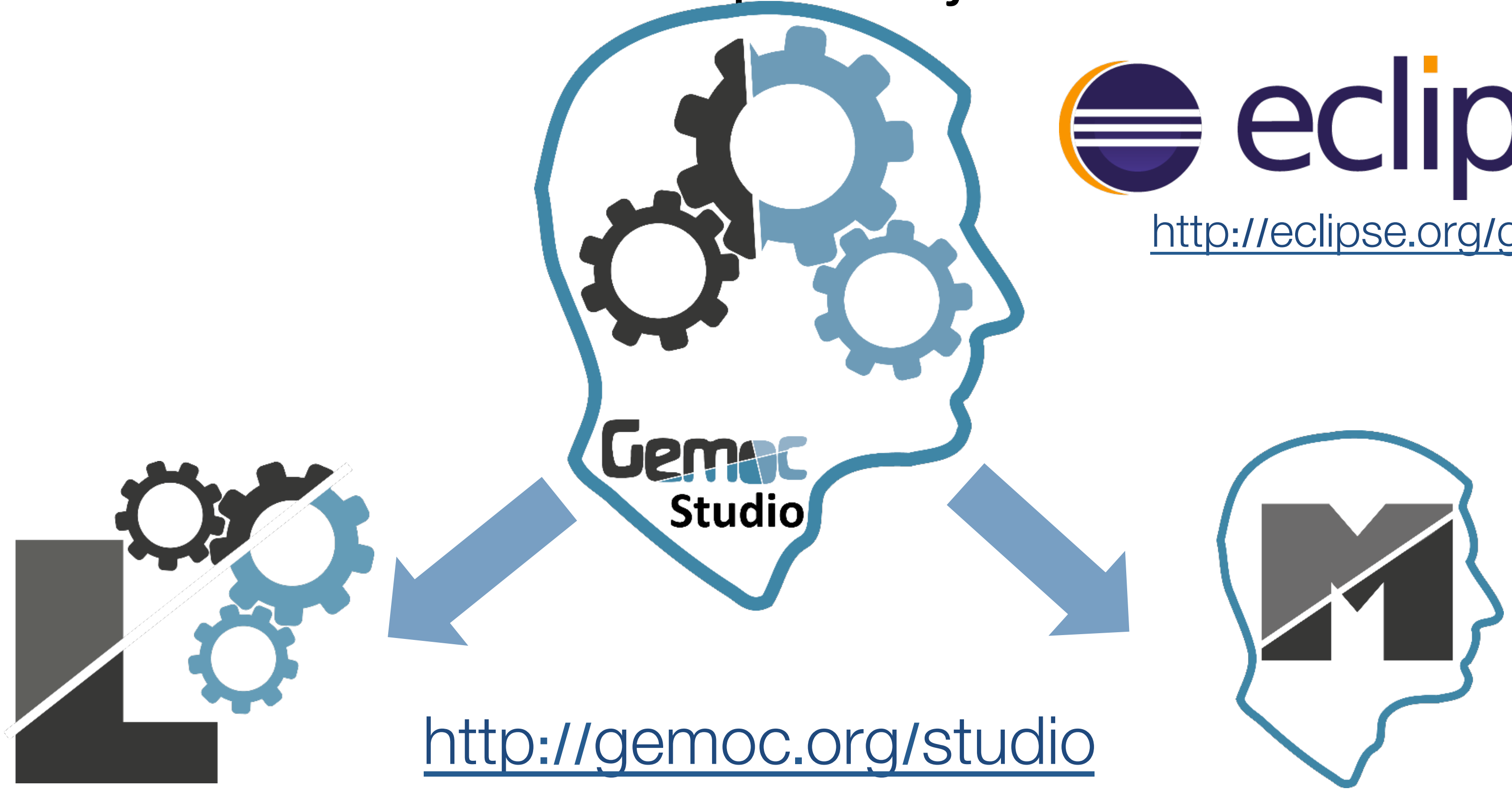


Heterogeneous Modeling

The Eclipse Project

eclipse

http://eclipse.org/gemoc



http://gemoc.org/studio

Language Workbench

Design and compose your executable DSMLs

Modeling Workbench

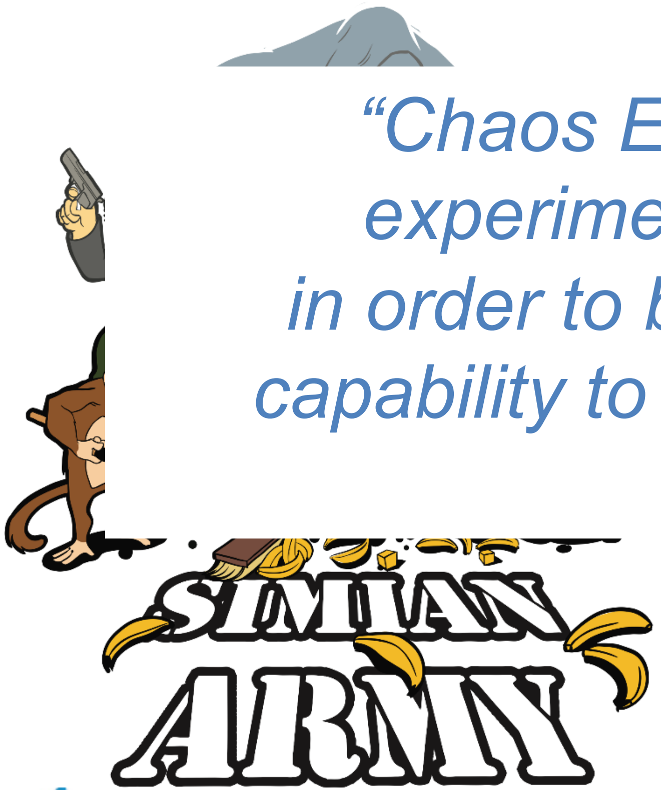
Edit and debug your heterogeneous models

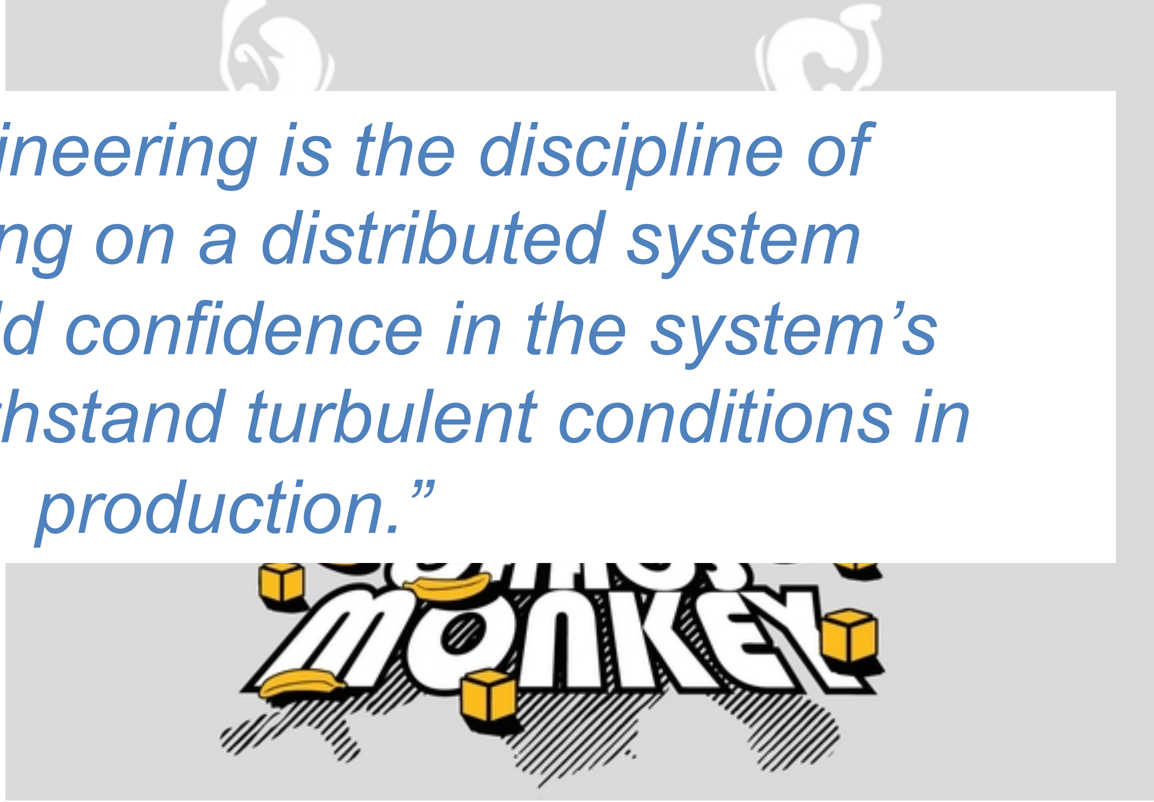
Talk outline

1. Motivations
2. Approach overview
3. Quick introduction to Docker
4. A Sirius-based workflow designer
5. Runtime implementation overview
6. Docker and HPC
7. Lesson learnt and open questions

Your own open-source and low-cost Netflix solution using micro-services

For teaching SLE/MDE, Distributed computing, Operating System and Chaos Engineering





> *"Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production."*

Motivations

- Securely manage high-volume live and on demand video processing/encoding solutions in combination with the scale
- Elasticity of your own/private cloud (edge/fog computing, https://arrayofthings.github.io/)
- Automatically provision and dynamically scale worker instances, and can seamlessly integrate those resources with on site infrastructure to instantly expand video processing capacity

Execute and isolate tasks using containers

- batch computing/cluster management tool using Docker as execution/isolation system

Execute and isolate tasks using containers

- We are like Pirates, pillaging for resources instead of booty!
  - We want to run our jobs. We want to get results
  - And when we find available resources, we need to ensure application and environment compatibility

=> This is where containers can be a perfect fit…

- But as I mentioned, our use-case and needs are different from enterprise!

What is Docker?

- Docker is a ‘container technology’
  - Linux-specific
    - can’t run Mac OSX, Windows *in* docker containers
    - But *can* run docker containers *on* Mac OSX & Windows
  - Shrink-wrap your software, run it on any Linux platform
- *Not* a virtual machine
  - Similar to virtual machines, but more lightweight
    - Smaller, faster to start, easier to maintain and manage
    - Lighter on system resources => vastly more scalable
  - VM-thinking will lead to poor results, avoid it!

Why use it?

- Portability:
  - No need to rebuild your application for a new platform!
    - Build a container once, run it anywhere
      - AWS/GCP/…
    - Stable s/w versions across all platforms, no runtime glitches
  - Think of it as ‘modules-to-go’
    - Instead of ‘module load PQR’ you ‘docker pull PQR’
    - No waiting for modules to be built/deployed for you!

- Reproducibility:
  - Because your s/w is stable, your pipeline is reproducible
    - Run the exact same binaries again 10 years from now ☺ ☹

What can you do with it?

- Computational workloads
  - Use applications without having to install them
  - Run your applications anywhere; clouds, HPC centres
  - **Reproducible pipelines**
- Services
  - Web portals/gateways (R/Shiny, Apache, Jupyter…)
  - Continuous build systems (Gitlab…)
  - For prototyping or for production running (databases etc)

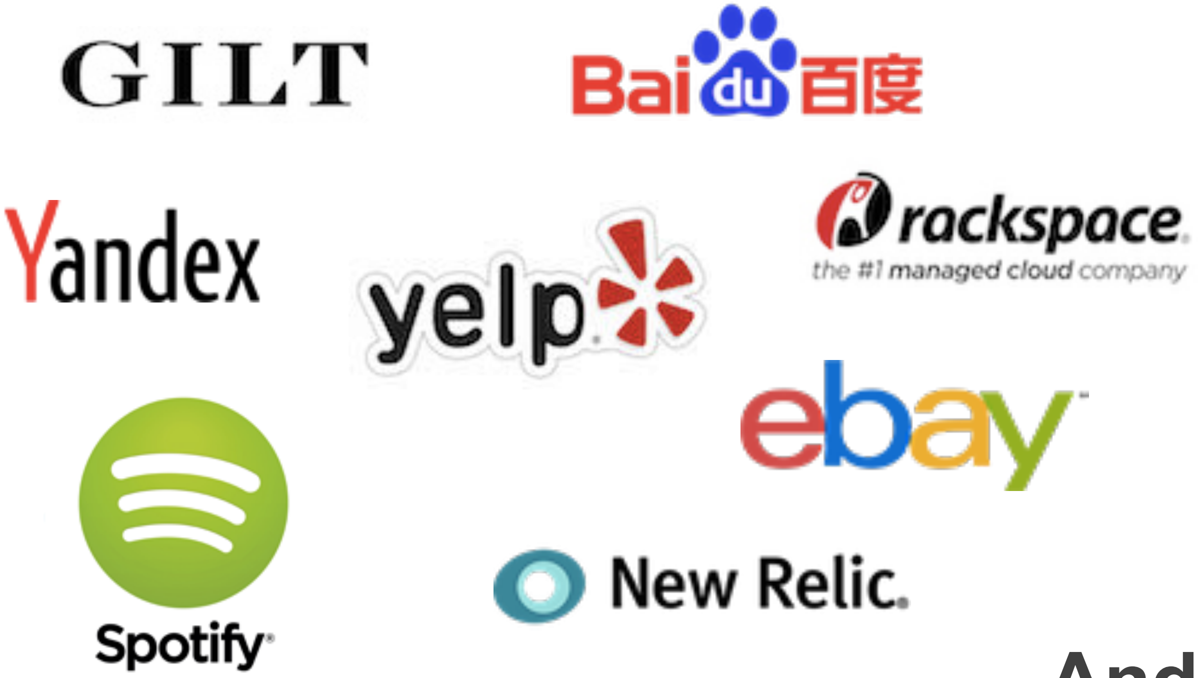
History

- Dotcloud, Inc creates PaaS service
- January 2013, work starts on docker internally
- March 2013, first public release
- Statistics:
  - 44 328 stars on github
  - 13 152 forks
  - 1693 contributors
  - 32 929 Commits
- Massive community interest
- Created by Solomon Hykes (French engineer ;)
- Open source project => Mobby for open innovation

> EVER TRIED.
> EVER FAILED.
> NO MATTER.
> TRY AGAIN.
> FAIL AGAIN.
> FAIL BETTER.
>
> *Samuel Beckett (1906-1989)*

Who uses Docker?

<u>Companies using Docker</u>



And many more…

Who uses Docker?

Docker PAAS Providers

Microsoft Azure

dotCloud

Google Cloud Platform

tutum

StackDock

amazon web services™

Quay.io

DigitalOcean

And many more…

Who uses Docker?

As an Infrastructure Tool along side

CHEF™

puppet labs

ANSIBLE

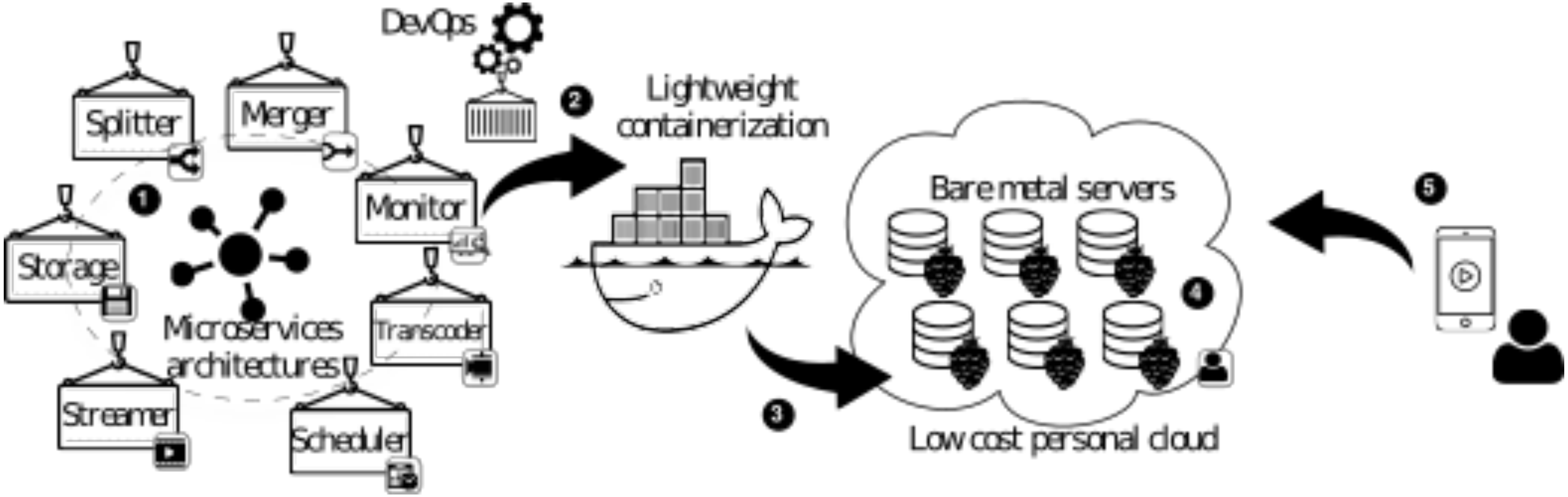
VAGRANT

SALTSTACK

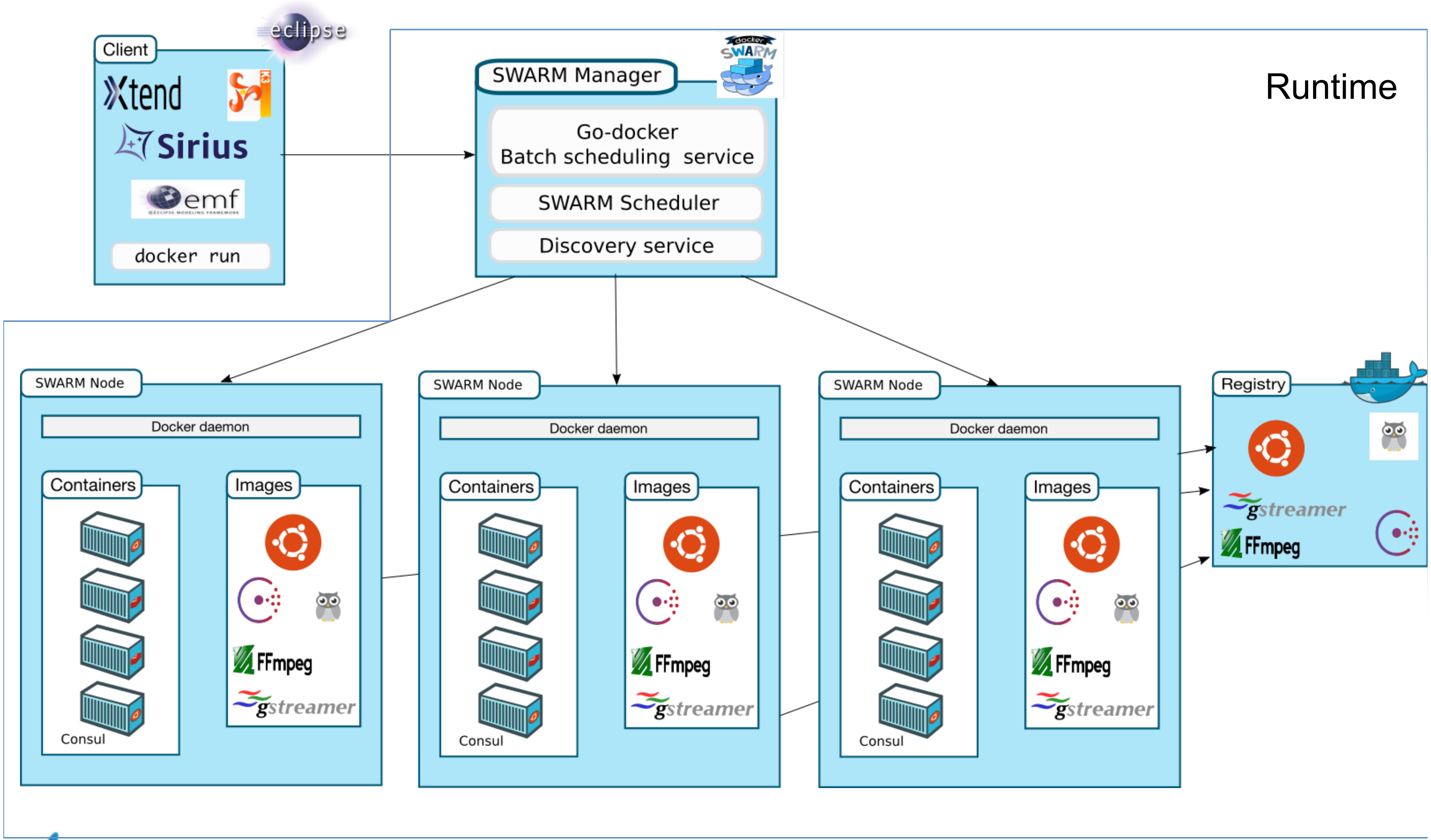
CFEngine®

Jenkins

Approach overview



Architecture Overview



Client

Xtend

Sirius

emf

docker run

SWARM Manager

Go-docker
Batch scheduling service

SWARM Scheduler

Discovery service

Runtime

SWARM Node

Docker daemon

Containers

Consul

Images

SWARM Node

Docker daemon

Containers

Consul

Images

SWARM Node

Docker daemon

Containers

Consul

Images

Registry

Focus on docker swarm

- Native clustering for Docker
- Written in GO
- Swarm is a simple tool which controls a cluster of Docker hosts and exposes it as a single "virtual" host
- Swarm uses the standard Docker API as its frontend, which means any tool which speaks Docker can control swarm transparently

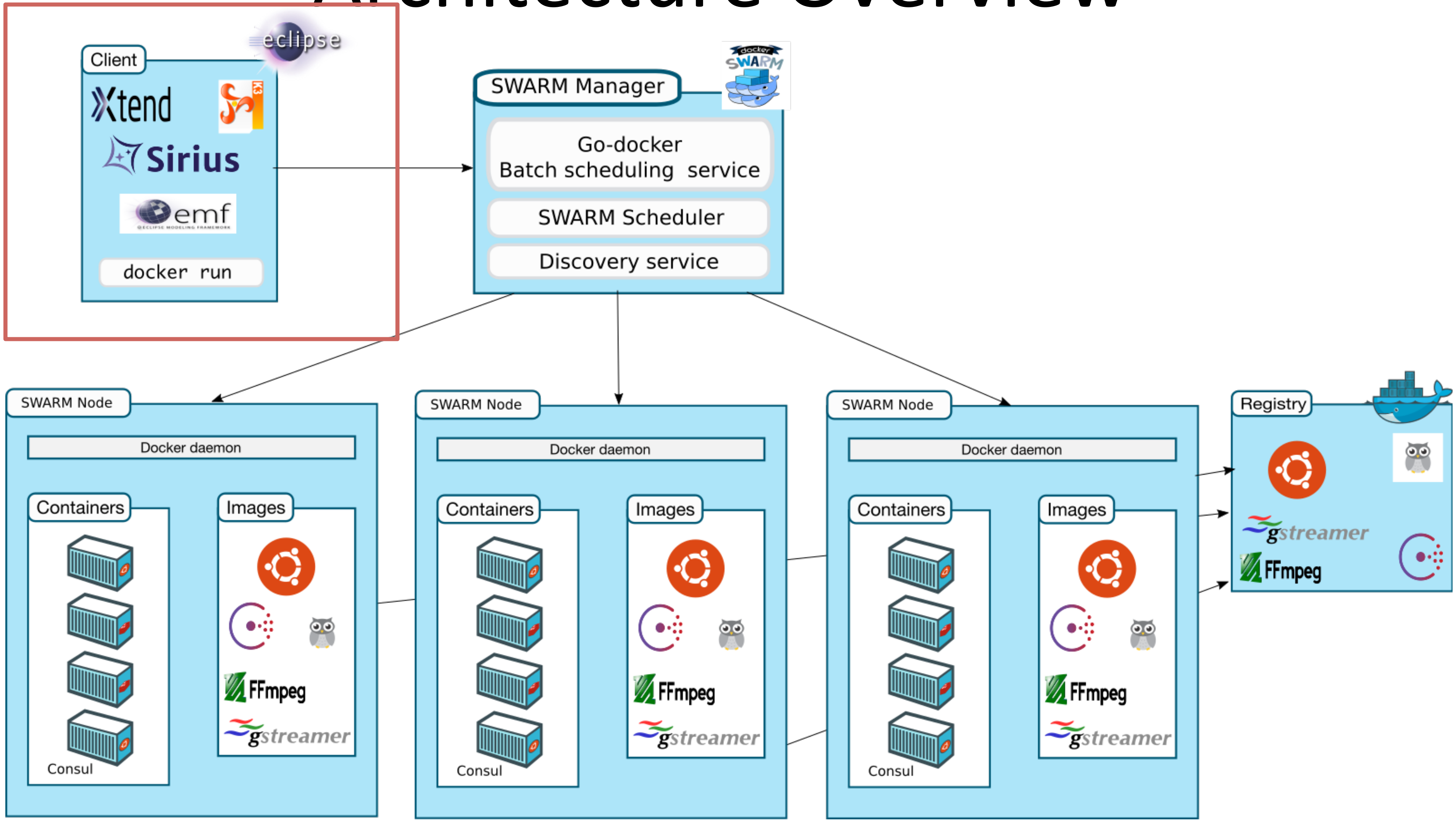
But, at this time, swarm mode is focused on long-running services, no real support for batch scheduling

Focus on Go-docker

- Batch computing/cluster management tool using Docker as execution/isolation system
- Written mostly in Python
- Open source
- like Sun Grid Engine/Torque/Slurm...
- Main contributors (GenOuest BioInformatics Platform)
  - Olivier Sallou [IRISA],
  - Cyril Monjeaud [IRISA]

But, at this time, it is a bit complex to extend for complex scheduling policies based on QoS priorities or resources availability

Architecture Overview



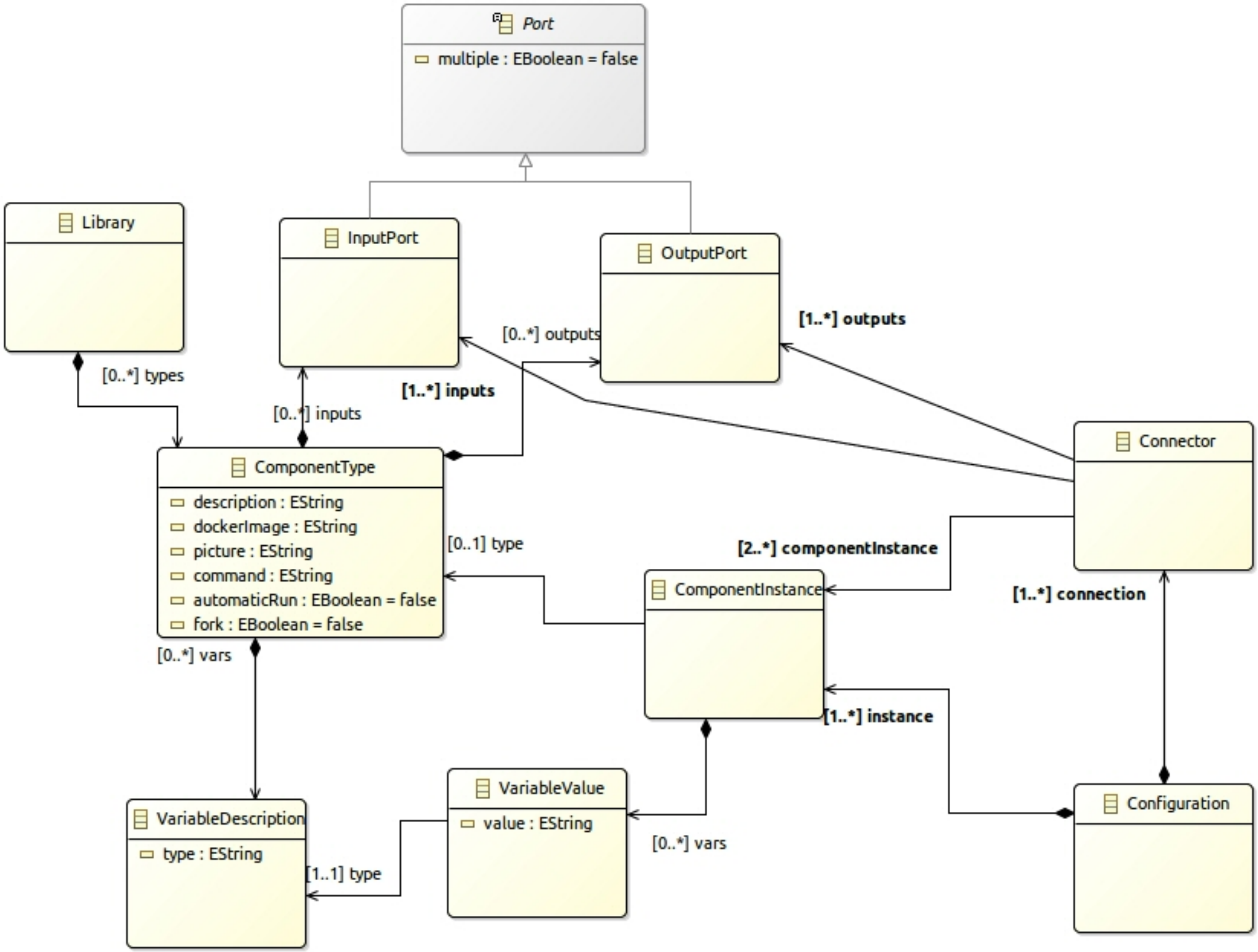
What else is missing?

- Tooling for non-experts
  - Graphical workflow designer to let an end-user defining its own video-processing workflow

=> Eclipse technologies to the rescue

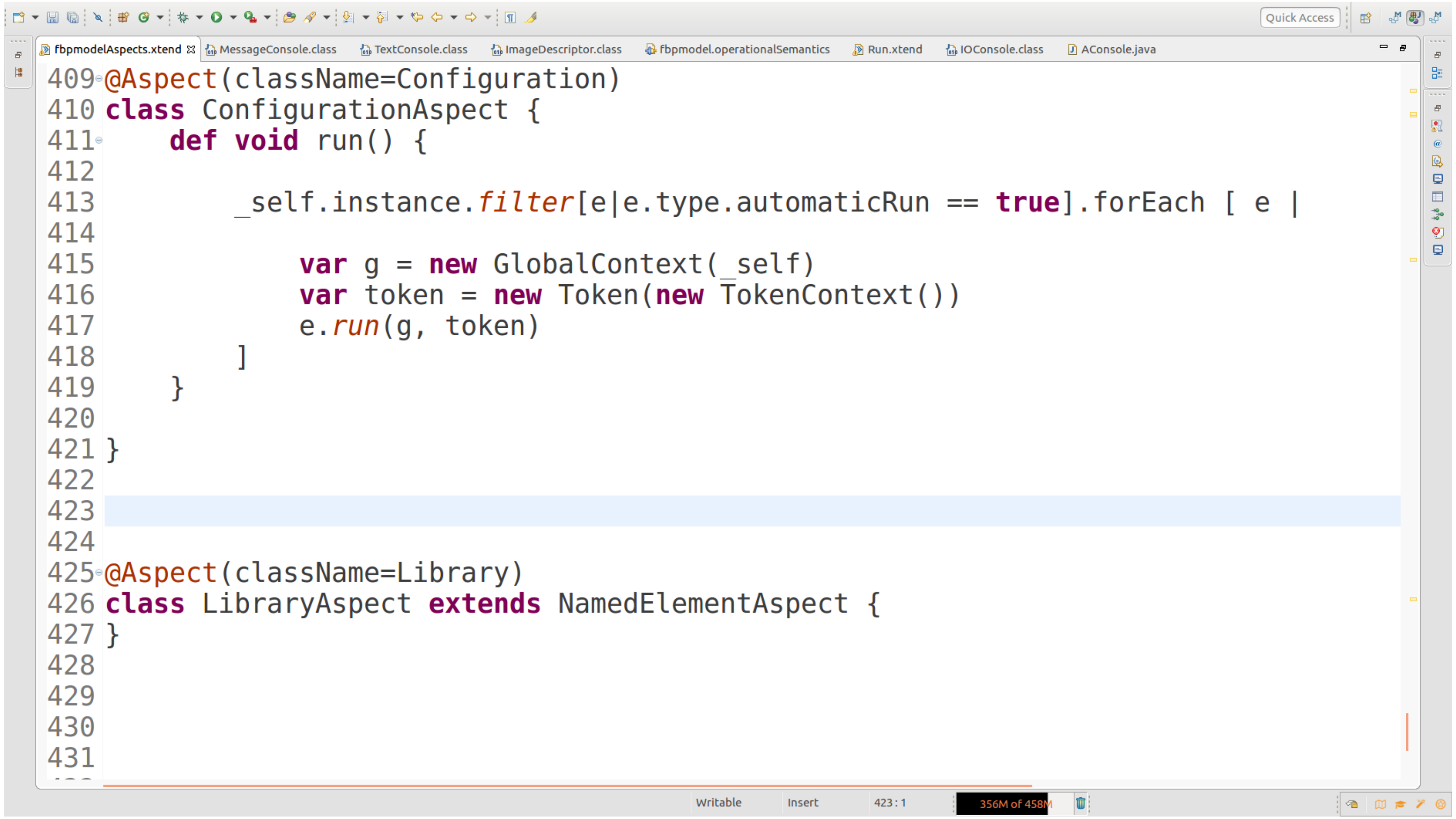
- A language based on flow-based programming paradigm
  - With its meta-model in Ecore
  - With its static semantics in OCL
  - With its operational semantics in Xtend/K3
  - With its graphical concrete syntax in Sirius
  - With its animator with Sirius Animator (Release soon)

A metamodel


Port
multiple : EBoolean = false
Library
InputPort
OutputPort
[1..*] outputs
[0..*] outputs
[0..*] types
[1..*] inputs
[0..*] inputs
Connector
ComponentType
description : EString
dockerImage : EString
picture : EString
command : EString
automaticRun : EBoolean = false
fork : EBoolean = false
[0..1] type
[2..*] componentInstance
ComponentInstance
[1..*] connection
[0..*] vars
[1..*] instance
VariableValue
value : EString
Configuration
VariableDescription
type : EString
[1..1] type
[0..*] vars

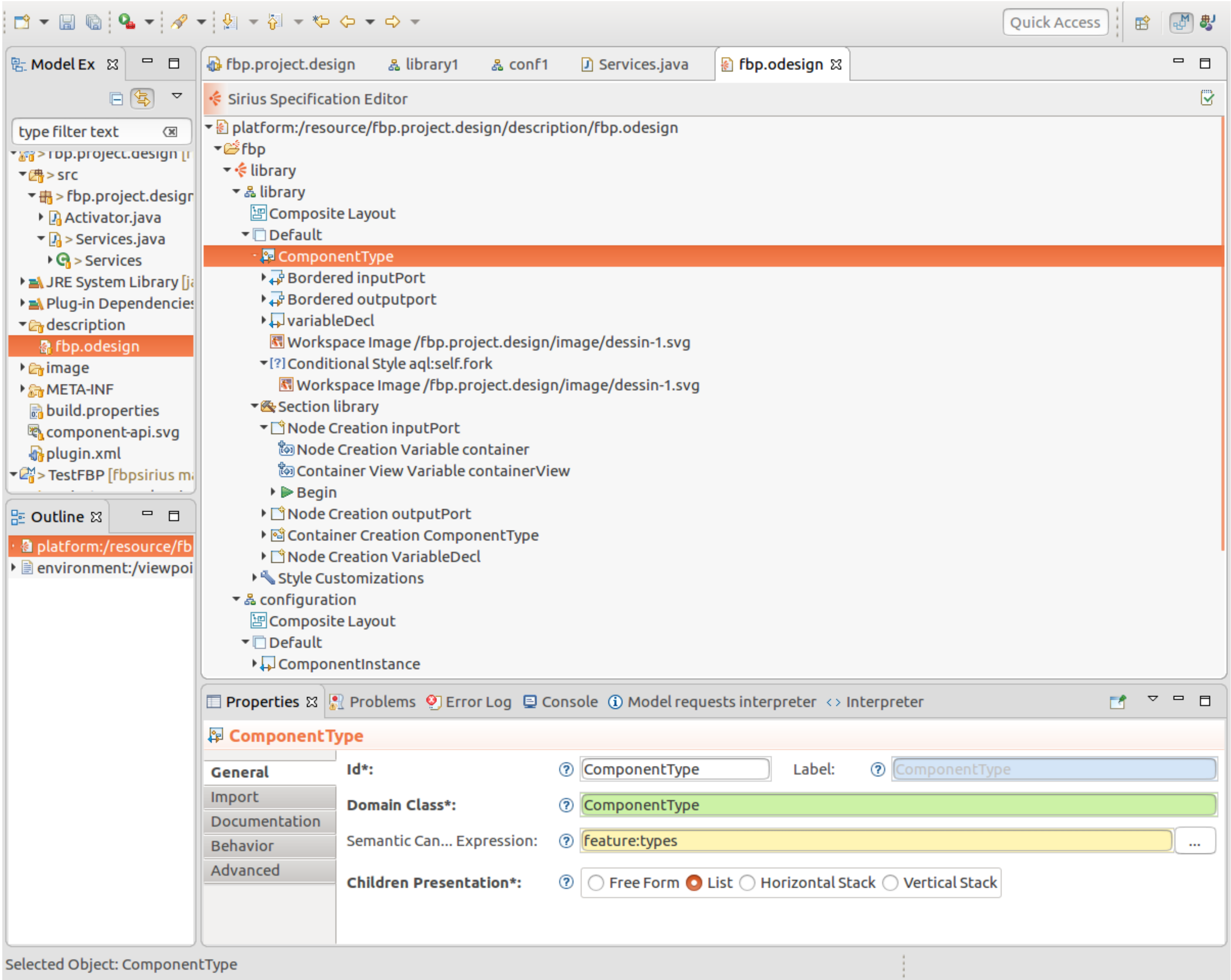
Xtend

Its operational semantics



```
@Aspect(className=Configuration)
class ConfigurationAspect {
    def void run() {

        _self.instance.filter[e|e.type.automaticRun == true].forEach [ e |

            var g = new GlobalContext(_self)
            var token = new Token(new TokenContext())
            e.run(g, token)
        ]
    }

}



@Aspect(className=Library)
class LibraryAspect extends NamedElementAspect {
}
```

Its graphical syntax description



Quick Access

Model Ex | fbp.project.design | library1 | conf1 | Services.java | fbp.odesign

type filter text

```
> fbp.project.design [
  > src
    > fbp.project.desigr
      Activator.java
      > Services.java
        > Services
  JRE System Library [j
  Plug-in Dependencies
  description
    fbp.odesign
  image
  META-INF
  build.properties
  component-api.svg
  plugin.xml
> TestFBP [fbpsirius m
```

Outline

```
platform:/resource/fb
environment:/viewpoi
```

Sirius Specification Editor

```
platform:/resource/fbp.project.design/description/fbp.odesign
  fbp
    library
      library
        Composite Layout
        Default
          ComponentType
            Bordered inputPort
            Bordered outputport
            variableDecl
            Workspace Image /fbp.project.design/image/dessin-1.svg
            [?] Conditional Style aql:self.fork
              Workspace Image /fbp.project.design/image/dessin-1.svg
          Section library
            Node Creation inputPort
              Node Creation Variable container
              Container View Variable containerView
              Begin
            Node Creation outputPort
            Container Creation ComponentType
            Node Creation VariableDecl
          Style Customizations
      configuration
        Composite Layout
        Default
          ComponentInstance
```

Properties | Problems | Error Log | Console | Model requests interpreter | <> Interpreter

ComponentType

General | Import | Documentation | Behavior | Advanced

Id*: ComponentType   Label: ComponentType

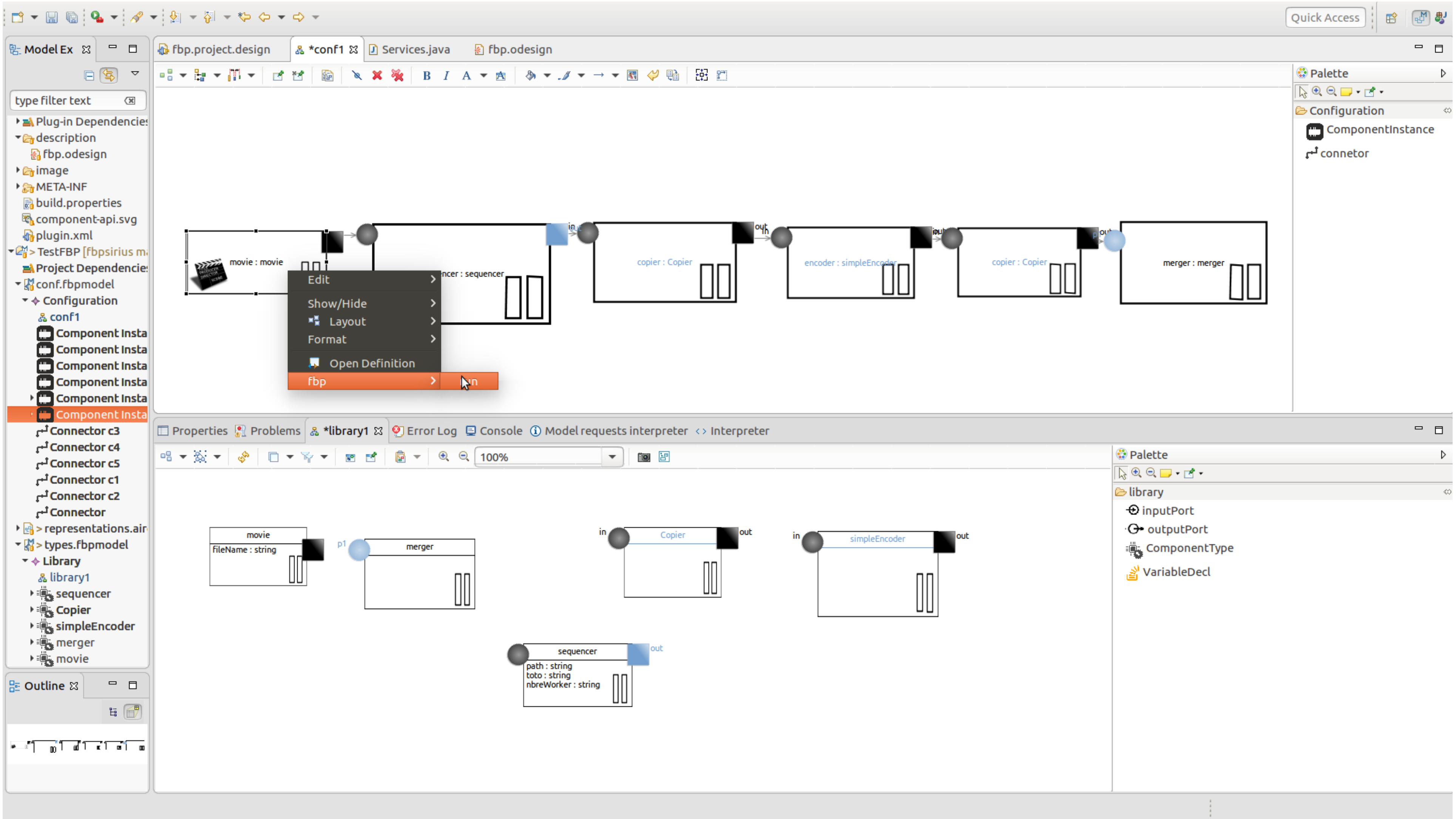
Domain Class*: ComponentType

Semantic Can... Expression: feature:types

Children Presentation*: ○ Free Form ● List ○ Horizontal Stack ○ Vertical Stack

Selected Object: ComponentType

A flow-based programming editor



Lesson learnt

- You can use directly Docker
  - You can! It works great for local and private resources. You can use it to develop and share your work with others using Docker-hub
- The basic value proposition is that they help to manage and run applications with complex dependencies easily and efficiently
- Specialize Docker swarm or SwarmKit is not "complex"

Cluster manager complexity

- For go-docker (28 lines of shell scripts)
- For Sirius and EMF stuffs (2 working days to get an initial version)

http://olivier.barais.fr/blog/posts/2015.12.01/GODocker_VideoEncoding.html

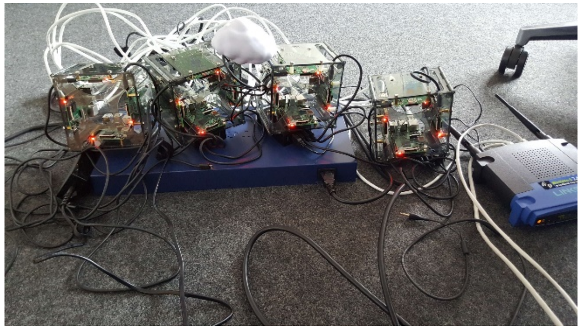
http://olivier.barais.fr/blog/posts/2016.03.24/GODocker_on_top_rpi.html

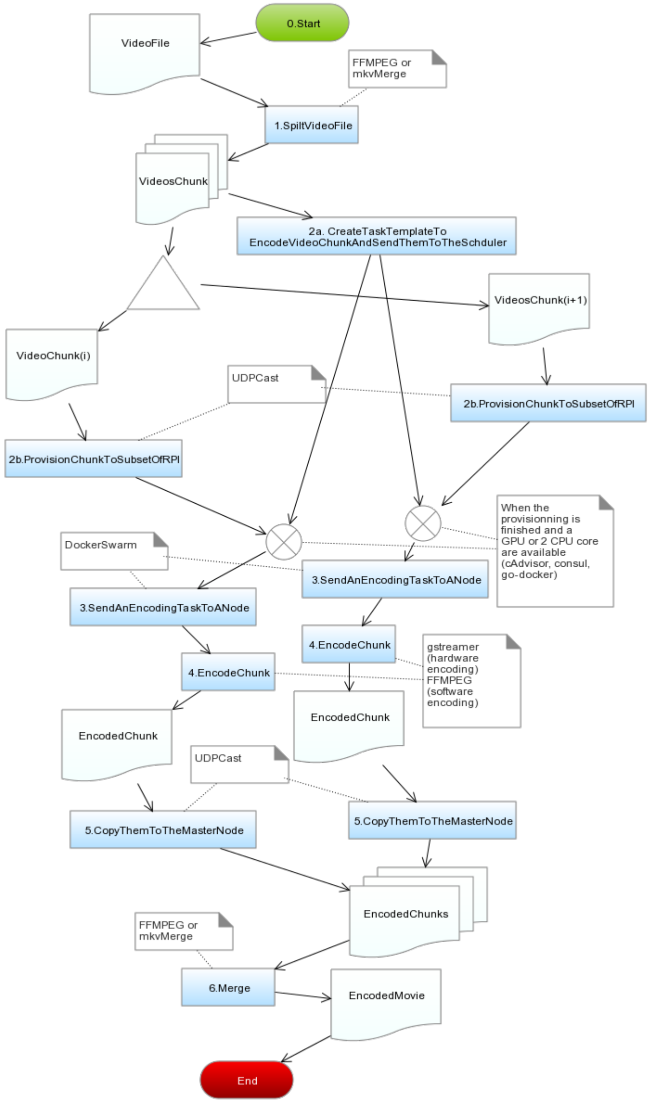
https://github.com/barais/swarm

https://github.com/barais/fbpsirius

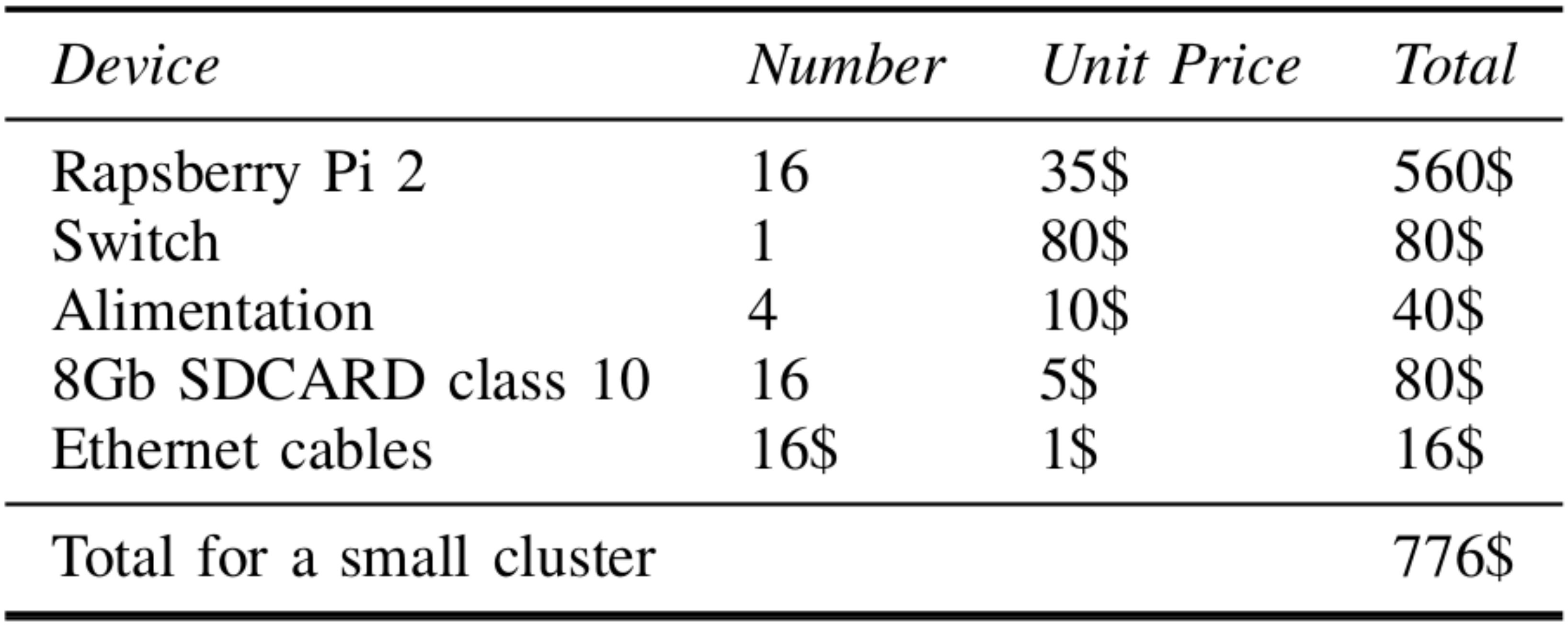
Scheduling policy

- Pinning on each raspberry pi
  - One core for managing the GPU encoding
  - One core for cluster management (consul agent, docker swarm agent) and chunk transfer management
  - Two cores for Software video encoding




0.Start
VideoFile
FFMPEG or mkvMerge
1.SpiltVideoFile
VideosChunk
2a. CreateTaskTemplateTo EncodeVideoChunkAndSendThemToTheSchduler
VideosChunk(i+1)
VideoChunk(i)
UDPCast
2b.ProvisionChunkToSubsetOfRPI
2b.ProvisionChunkToSubsetOfRPI
When the provisionning is finished and a GPU or 2 CPU core are available (cAdvisor, consul, go-docker)
DockerSwarm
3.SendAnEncodingTaskToANode
3.SendAnEncodingTaskToANode
4.EncodeChunk
gstreamer (hardware encoding) FFMPEG (software encoding)
4.EncodeChunk
EncodedChunk
EncodedChunk
UDPCast
5.CopyThemToTheMasterNode
5.CopyThemToTheMasterNode
EncodedChunks
FFMPEG or mkvMerge
6.Merge
EncodedMovie
End

Cost



Device	*Number*	*Unit Price*	*Total*
Rapsberry Pi 2	16	35$	560$
Switch	1	80$	80$
Alimentation	4	10$	40$
8Gb SDCARD class 10	16	5$	80$
Ethernet cables	16$	1$	16$
Total for a small cluster			776$

Energy consumption

- 16 Raspberry Pis with the switch consume about 80 Watts when used at full power

VS

- i7 5775C which consumes on average 99 watts when performing a x264 encoding task[1]

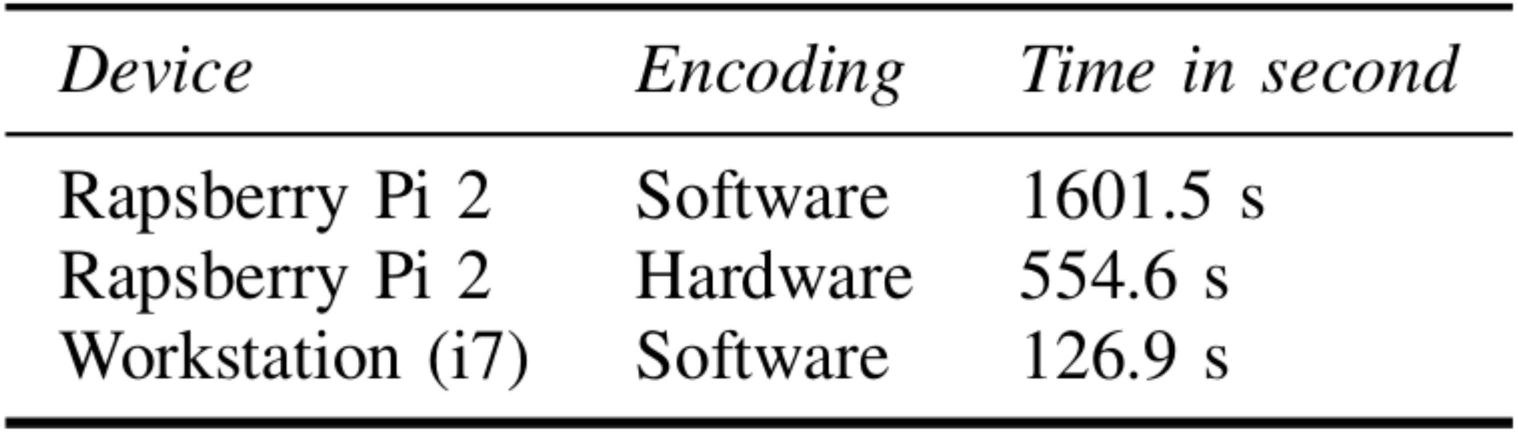
[1] Taken from the hardware test conducted here http://techreport.com/review/28751/intel-core-i7-6700k-skylake-processor-reviewed/5

Performance evaluation

- Comparison of the encoding of a video using a workstation (featuring an Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz, 16 Gb of memory, running on Linux Ubuntu) and our cluster of 16 Raspberry Pi 2
- Encoding a H264 video file into another H264 video file with the ”High profile”.
- Input video and output video resolution = 1280*688 Px.

First performance evaluation

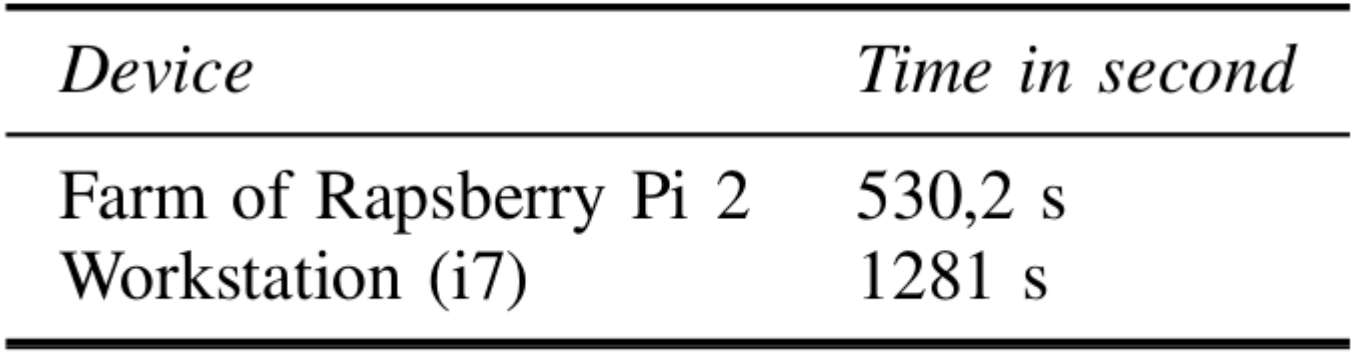
- time needed to encode a small video chunk of 2 mins and 30 seconds both on the workstation and on a single raspberry pi 2



Device	*Encoding*	*Time in second*
Rapsberry Pi 2	Software	1601.5 s
Rapsberry Pi 2	Hardware	554.6 s
Workstation (i7)	Software	126.9 s

Second performance evaluation

- time needed to encode a small video chunk of 25 min both on the workstation and the farm of raspberry pi 2



Device	*Time in second*
Farm of Rapsberry Pi 2	530,2 s
Workstation (i7)	1281 s

The good news

- Docker helps to support such requirements
  - I need root!
  - Complex environments
  - Custom distros
  - Bringing your own application+stack
  - Preserving the stack for reproducibility
  - Sharing validated HPC stacks to users
  - Enhancing cluster management and testing

The good news

- HPC community is moving
  - HPCS Singularity
  - NeRSC Shifter
  - Keep compatibility with SLURM for resources allocation
- Jupyter
- R-studio

Wrap up

- Data Intensive computing often require complex software stacks
- Efficiently supporting “big software” in HPC environments offers many challenges

Open questions

- Lots of domains such as videos editing requires their own cluster manager and scheduler
- Container scheduler such as apache Mesos or Docker swarm are extensible
  - clean framework for developing and integrating these extensions but Python, Java, or Go are low-level GPL for developing Scheduler

⬄ But why not providing a clean and safe DSL for **designing cluster scheduler ?**

Discussion/Comments/ Questions

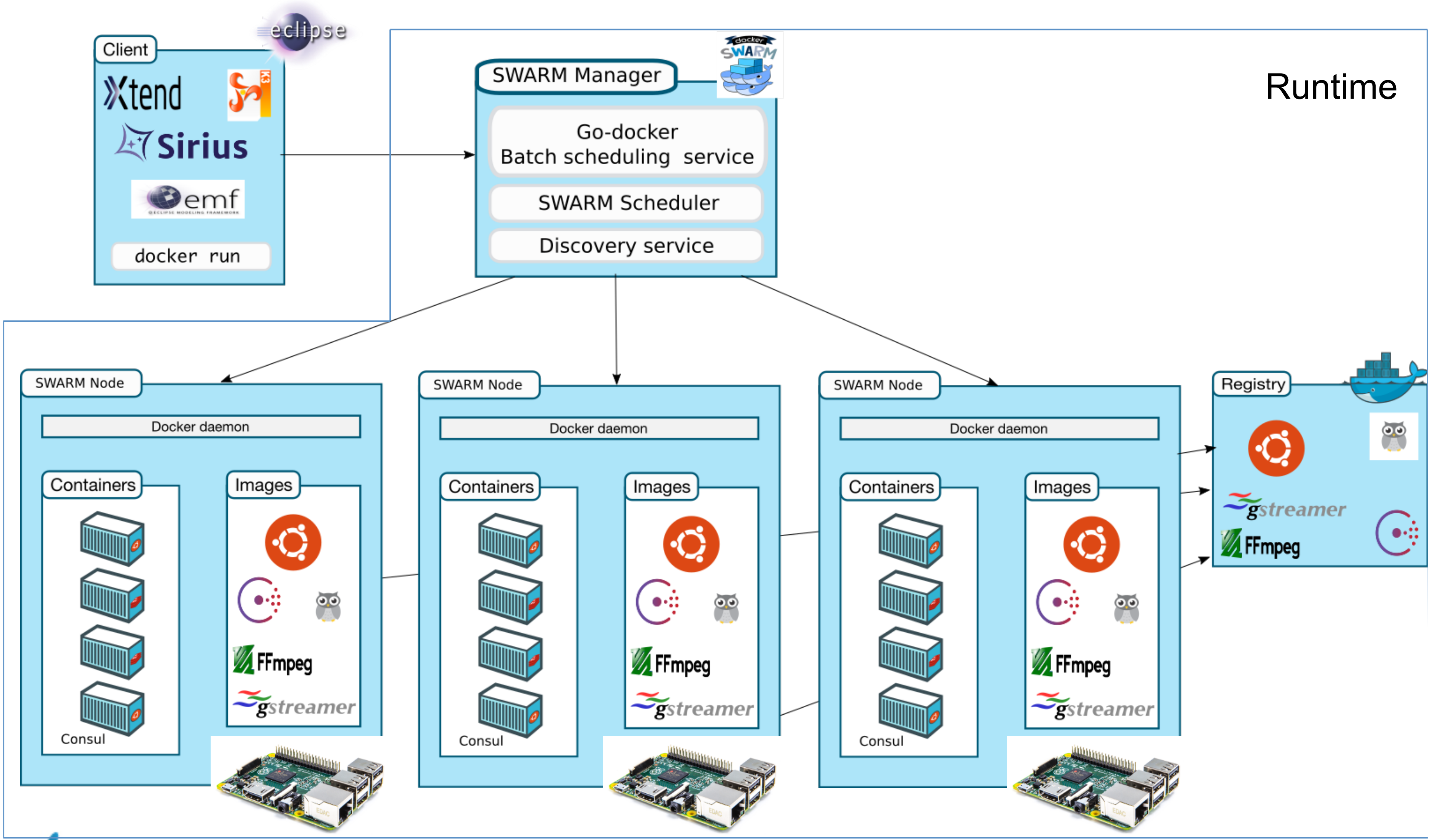
Towards micro-services architecture to transcode videos in the large at low costs - Olivier Barais, Johann Bourcier, David Bromberg, Christophe Dion, In Proceedings of the International conference on Telecommunications and Multimedia (TEMU), 2016

Greening the Video Transcoding Service with Low-Cost Hardware Transcoders - Peng Liu, Jongwon Yoon, Lance Johnson, Suman Banerjee. In proceedings of the 16th USENIX Annual Technical Conference (USENIX ATC '16). June 22–24, 2016, Denver, CO, USA

Current work, next steps and open questions

- Compare the performance in fixing the core used by each process
- Implement a version with nomad, mesos (IRT B-COM)
- Compare with the use of Hadoop and Apache Hadoop YARN (IRT B-COM)

Architecture Overview



Focus on docker swarm

- Native clustering for Docker
- Written in GO
- Swarm is a simple tool which controls a cluster of Docker hosts and exposes it as a single "virtual" host
- Swarm uses the standard Docker API as its frontend, which means any tool which speaks Docker can control swarm transparently

But, at this time, swarm mode is focused on long-running services, no real support for batch scheduling

Focus on Go-docker

- Batch computing/cluster management tool using Docker as execution/isolation system
- Written mostly in Python
- Open source
- like Sun Grid Engine/Torque/...
- Main contributors (GenOuest BioInformatics Platform)
  - Olivier Sallou [IRISA],
  - Cyril Monjeaud [IRISA]

But, at this time, it is a bit complex to extend for complex scheduling policies based on QoS priorities or resources availability

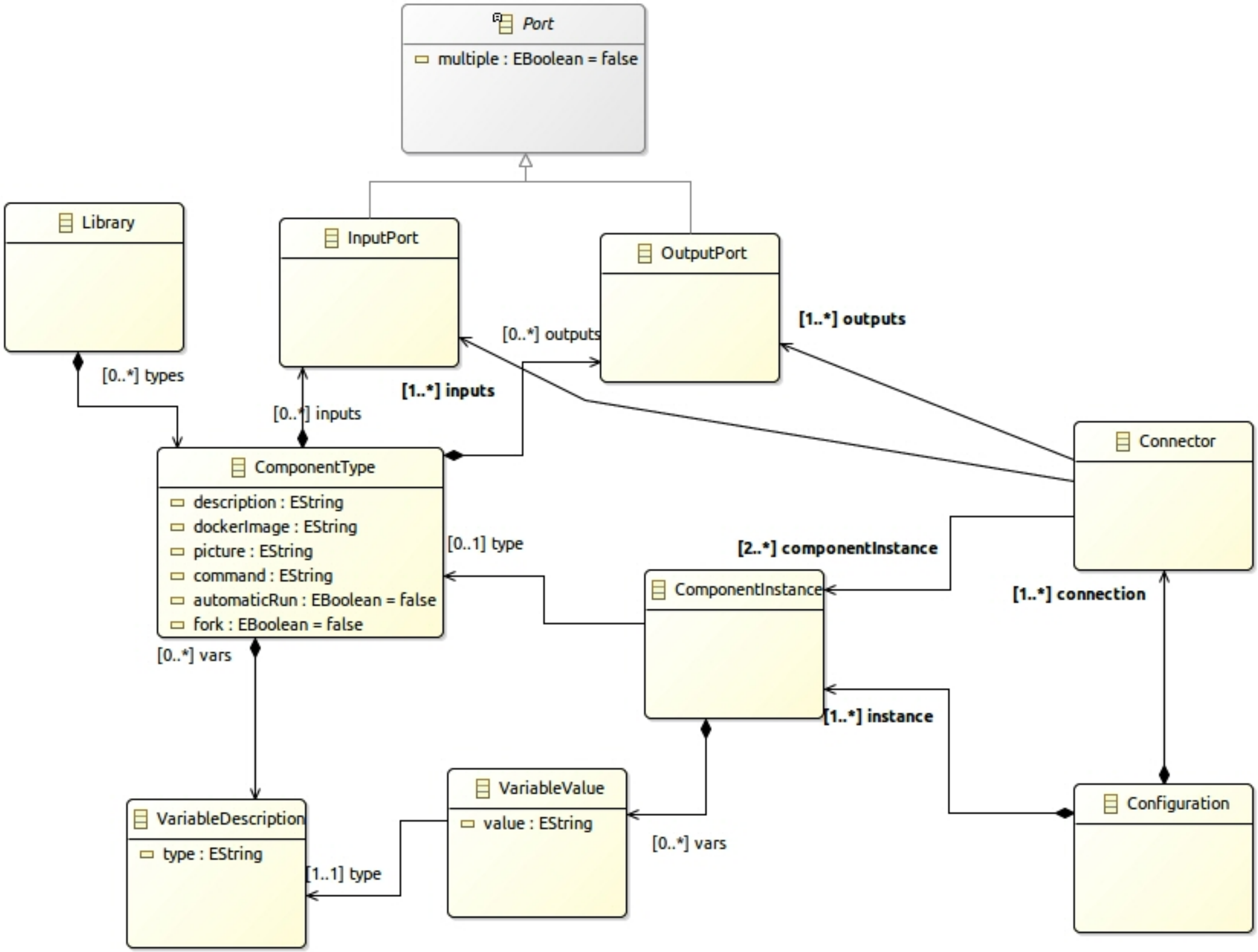
What else is missing?

- Tooling for non-experts
  - Graphical workflow designer to let an end-user defining its own video-processing workflow

=> Eclipse technologies to the rescue

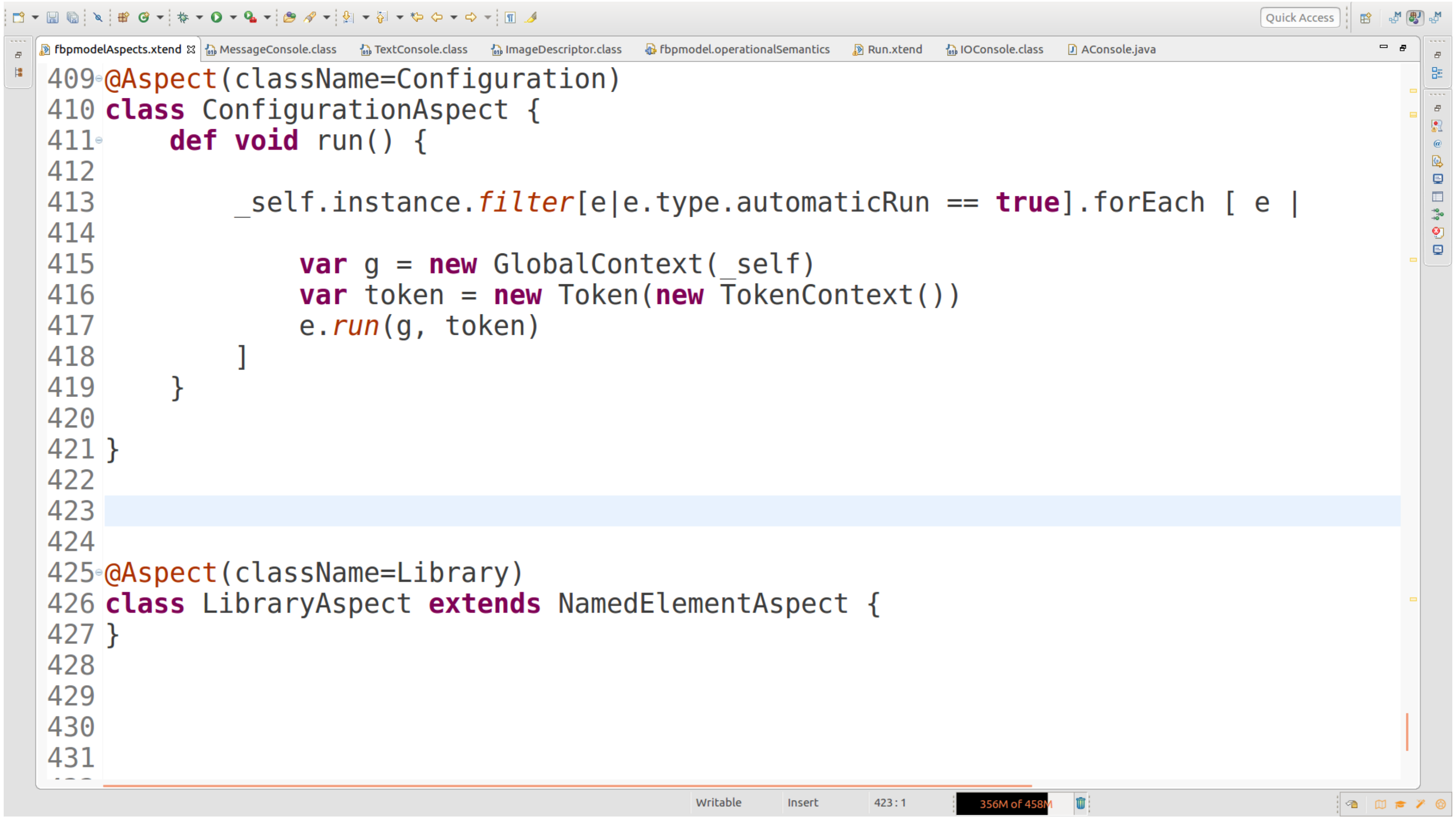
- A language based on flow-based programming paradigm
  - With its meta-model in Ecore
  - With its static semantics in OCL
  - With its operational semantics in Xtend/K3
  - With its graphical concrete syntax in Sirius
  - With its animator with Sirius Animator (Release soon)

A metamodel


Port
multiple : EBoolean = false
Library
InputPort
OutputPort
[1..*] outputs
[0..*] outputs
[0..*] types
[1..*] inputs
[0..*] inputs
Connector
ComponentType
description : EString
dockerImage : EString
picture : EString
command : EString
automaticRun : EBoolean = false
fork : EBoolean = false
[0..1] type
[2..*] componentInstance
ComponentInstance
[1..*] connection
[0..*] vars
[1..*] instance
VariableValue
value : EString
VariableDescription
type : EString
[1..1] type
[0..*] vars
Configuration

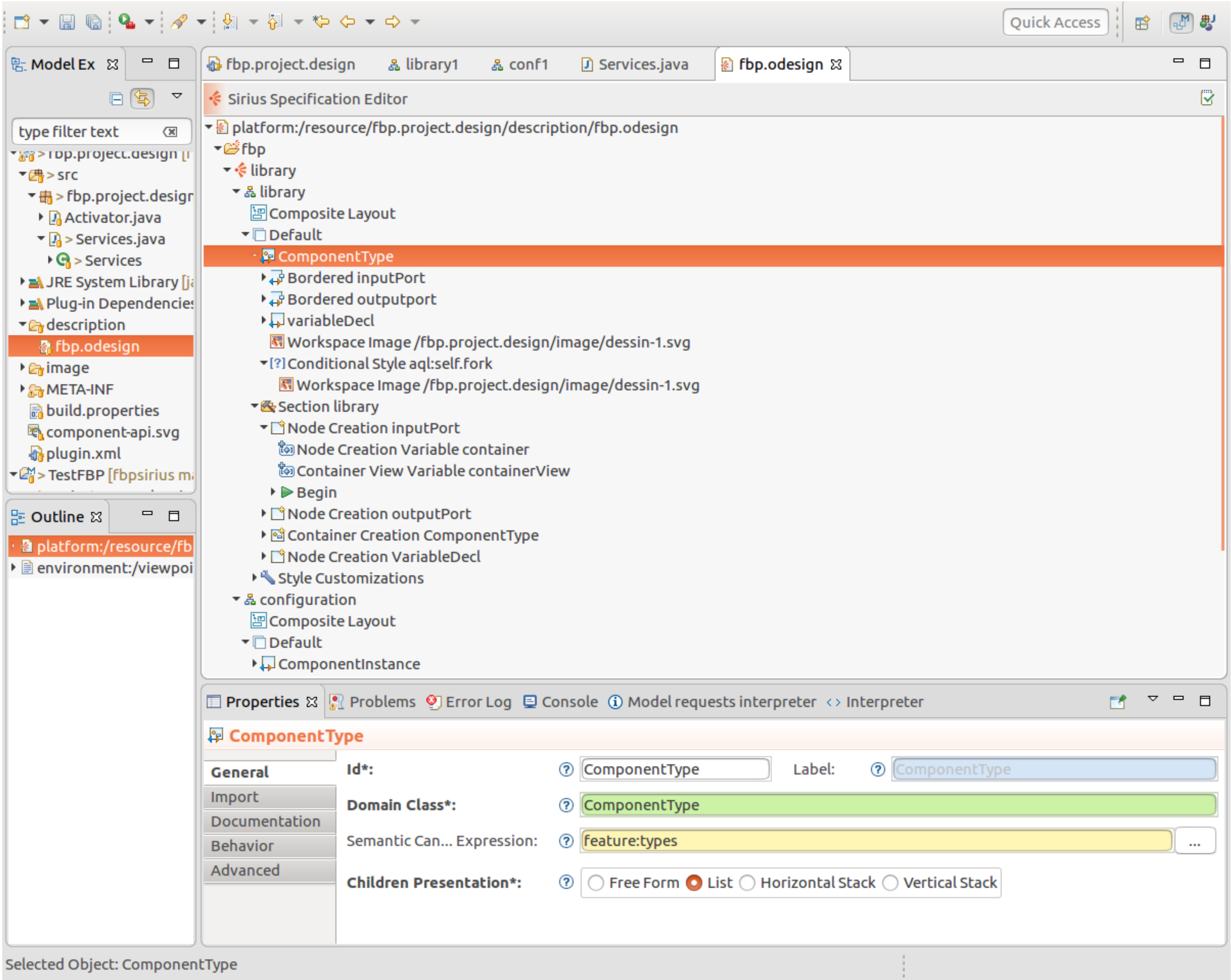
Its operational semantics

Xtend



```
@Aspect(className=Configuration)
class ConfigurationAspect {
    def void run() {

        _self.instance.filter[e|e.type.automaticRun == true].forEach [ e |

            var g = new GlobalContext(_self)
            var token = new Token(new TokenContext())
            e.run(g, token)
        ]
    }

}



@Aspect(className=Library)
class LibraryAspect extends NamedElementAspect {
}
```

Its graphical syntax description



```
> fbp.project.design [
  > src
    > fbp.project.design
      Activator.java
      > Services.java
        > Services
  JRE System Library [ja
  Plug-in Dependencies
  description
    fbp.odesign
  image
  META-INF
  build.properties
  component-api.svg
  plugin.xml
> TestFBP [fbpsirius m
```

```
platform:/resource/fbp.project.design/description/fbp.odesign
  fbp
    library
      library
        Composite Layout
        Default
          ComponentType
            Bordered inputPort
            Bordered outputport
            variableDecl
            Workspace Image /fbp.project.design/image/dessin-1.svg
            Conditional Style aql:self.fork
              Workspace Image /fbp.project.design/image/dessin-1.svg
          Section library
            Node Creation inputPort
              Node Creation Variable container
              Container View Variable containerView
              Begin
            Node Creation outputPort
            Container Creation ComponentType
            Node Creation VariableDecl
          Style Customizations
      configuration
        Composite Layout
        Default
          ComponentInstance
```

Outline:

```
platform:/resource/fb
environment:/viewpoi
```

ComponentType

General | Import | Documentation | Behavior | Advanced

Id*: ComponentType   Label: ComponentType

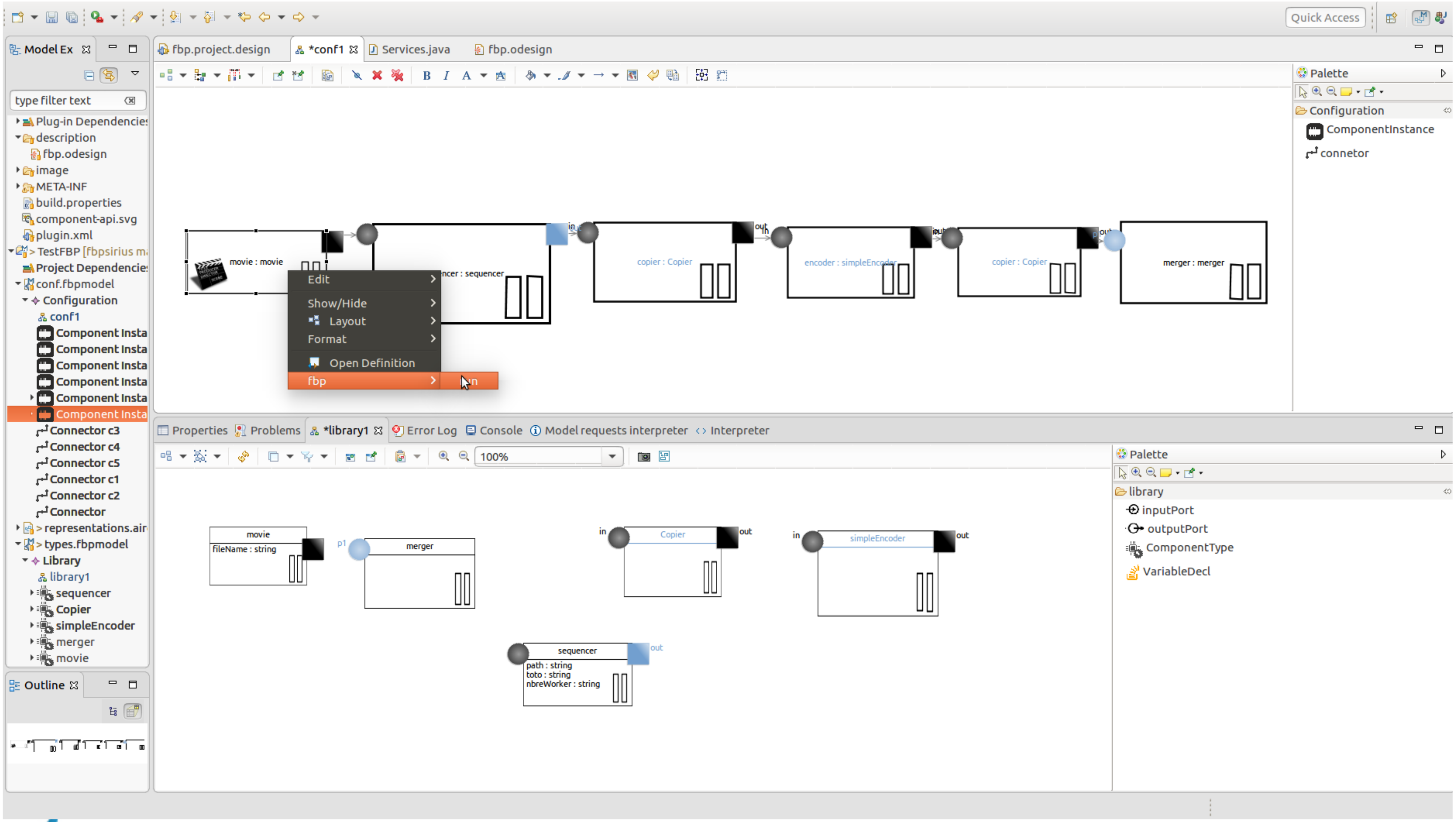
Domain Class*: ComponentType

Semantic Can... Expression: feature:types

Children Presentation*: ○ Free Form ● List ○ Horizontal Stack ○ Vertical Stack

Selected Object: ComponentType

A flow-based programming editor

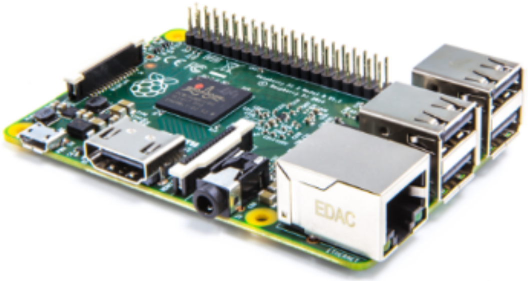


Demo

http://olivier.barais.fr/blog/posts/2015.12.01/GODocker_VideoEncoding.html
http://olivier.barais.fr/blog/posts/2016.03.24/GODocker_on_top_rpi.html
https://github.com/barais/swarm
https://github.com/barais/fbpsirius

Running on open source hardware and software

- Low-cost open source hardware



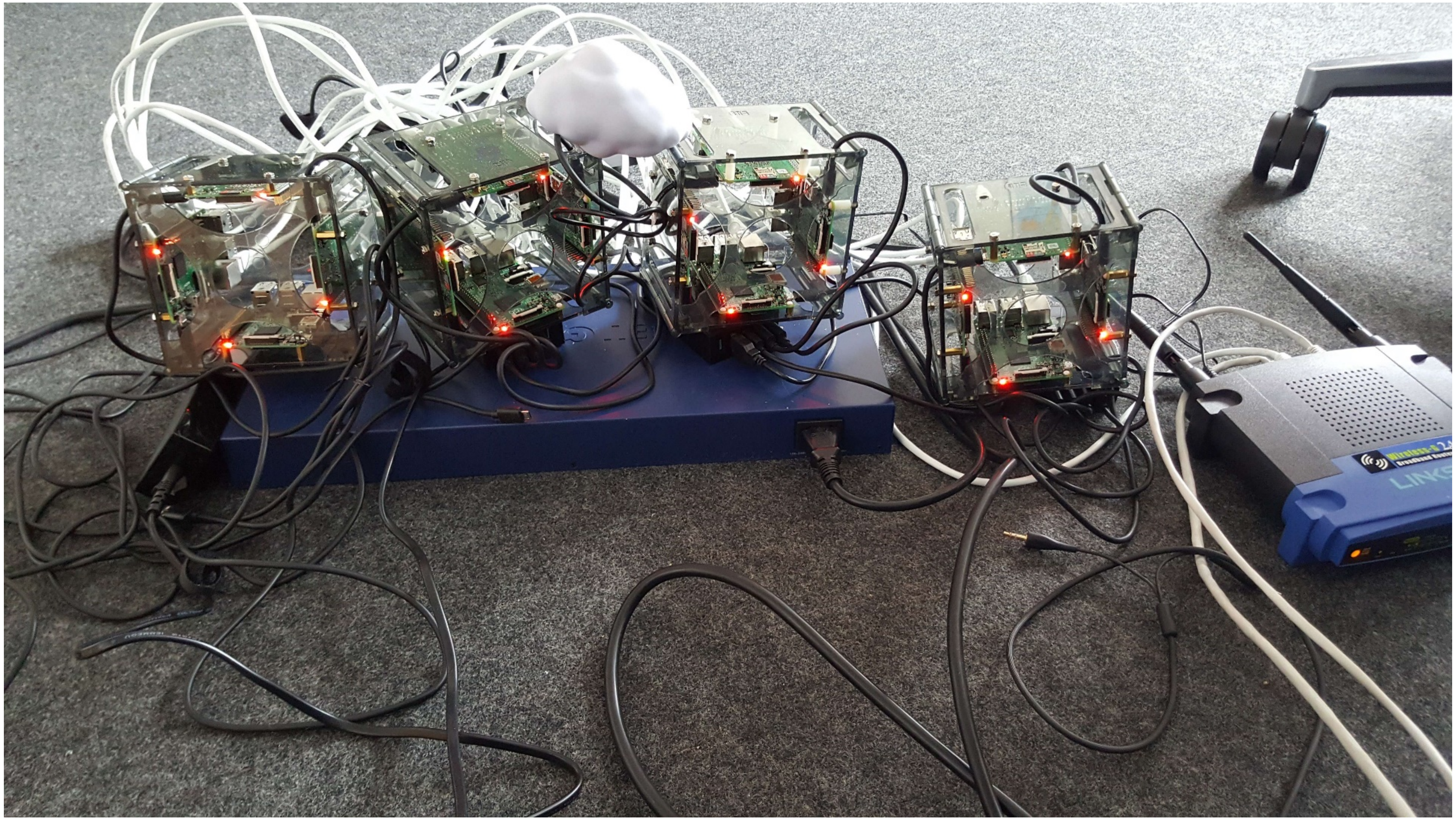
- Videos editing open source solutions

gstreamer FFMPEG

- System containers solution

docker

Implementation 1/4



Implementation 2/4

- A video transcoding workflow:
  - a *splitter* task
  - a *chunk transfer* task to transfer each chunk to a set of targeted host
  - a *scheduler* takes the decision to start the transcoding process on a specific node based on runtime information
  - a *video encoding* task with two different implementations: one for software encoding and another one for hardware encoding

Implementation 3/4

- A video transcoding workflow:
  - an *encoded chunk transfer* to transfer the encoded chunk back
  - a *merger* task which gather all encoded video chunk and assemble them incrementally
  - a *streamer* task which takes the output of the merger task to stream the newly encoded video

Implementation 4/4

- Splitter and merger: *MKVToolNix* or *FFMpeg*
- Chunk transfer: *udpcast*, *nfs*, *glusterfs*
- Videos encoding: *FFMpeg*, *OpenMAX*, *Gstreamer*
- Scheduler: Go-Docker or a modified version of swarm
- Key/Value data store. Consul
- Performance analysis: *Cadvisor*, *Grafana*, *InfluxDB*

Lesson learnt

- The development effort required to setup such platform for video transcoding
- The cost of our solution, both regarding financial investment and energy consumption
- The intrinsic encoding performance of our specific deployment setup

Current work, next steps and open questions

- Use of Sirius animator to view the status of the running workflow
- Tooling for ensuring the correctness of the resulting videos
- Improve the rate control management between encoding tasks