MAUVE Runtime: a component-based middleware to reconfigure software architectures in real-time

D. Doose, C. Grand, C. Lesire

IEEE Robotic Computing 2017

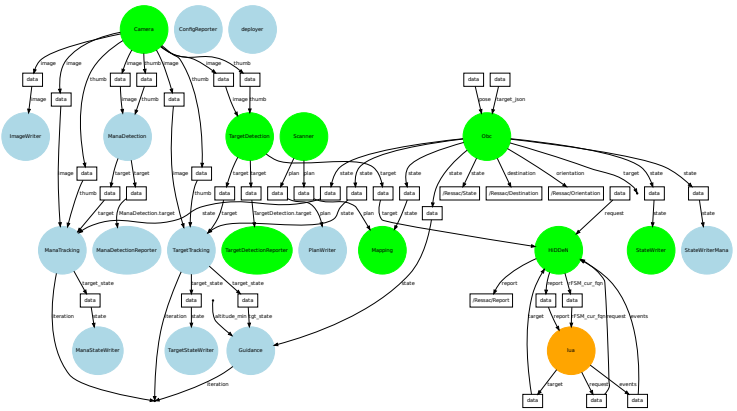
ONERA

THE FRENCH AEROSPACE LAB

retour sur innovation

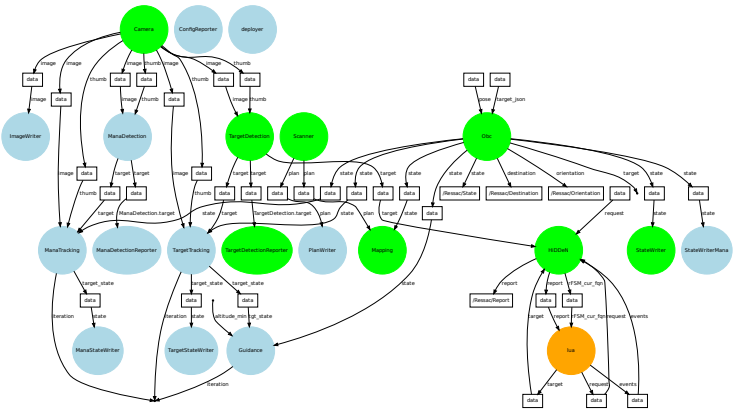
Introduction

- Software architectures in robotics:
  - complex, modular
  - critical (safety guarantees needed)
  - dynamic (reconfiguration mechanisms)



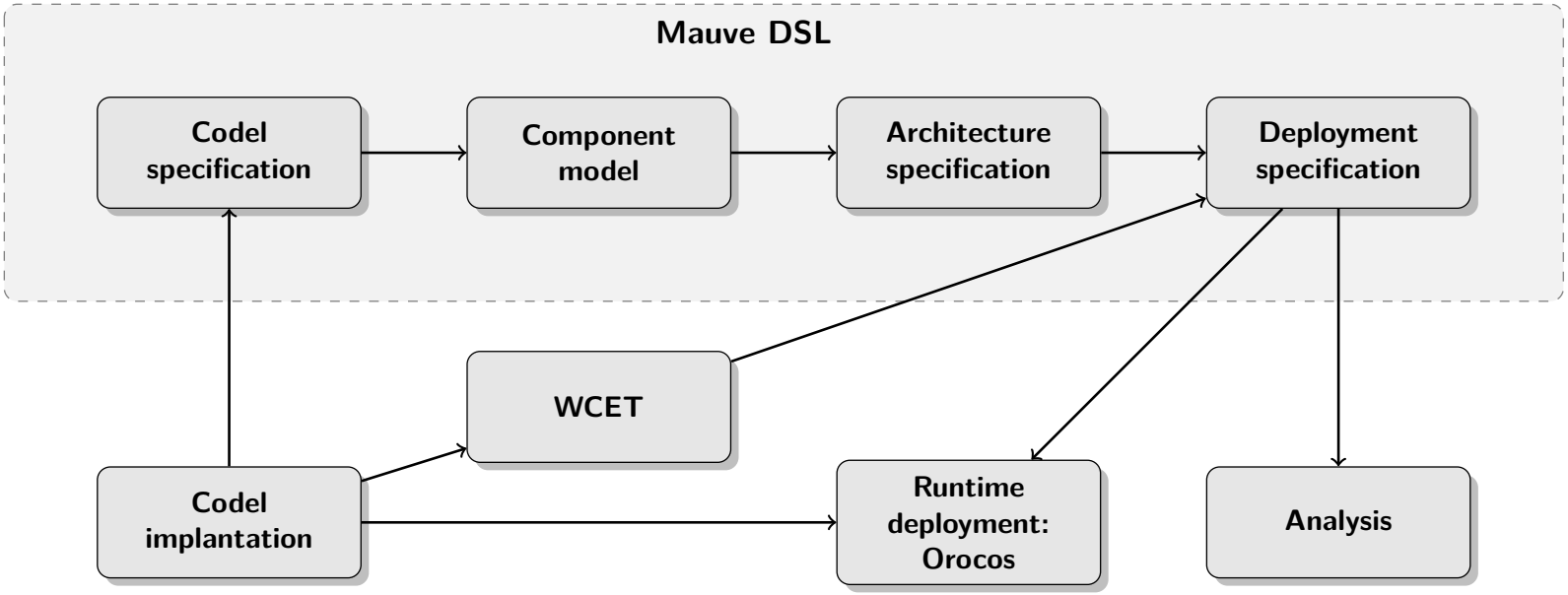
Introduction

- Software architectures in robotics:
  - complex, modular
  - critical (safety guarantees needed)
  - dynamic (reconfiguration mechanisms)
- Requires:
  - models and languages for design
  - safety/robustness mechanisms
  - safety assessment methods



MAUVE DSL and process

- First MAUVE version[1]
  - Architecture modeling using a DSL
  - Analysis tools, including real-time schedulability
  - Generation of OROCOS code



[1] Nicolas Gobillot, Charles Lesire, and David Doose (2014). "A Modeling Framework for Software Architecture Specification and Validation". In: *SIMPAR*. Bergamo, Italy.

Runtime rationales

1. Provide an execution model both formal (to ease analyses) and expressive (to allow implementing complex behaviors)
2. Masterize the synchronization of real-time tasks
3. Provide reconfiguration mechanisms
4. Provide a C++ API for programmers compliant with models

MAUVE Runtime

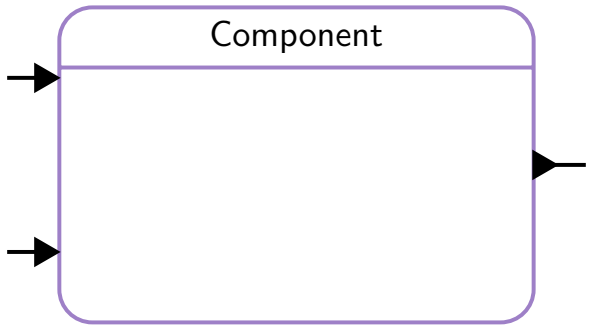
A component-based architecture middleware to design architectures with:

- components, i.e. tasks that execute code
- resources, that own data
- connections between components and resources
- real-time activities assigned to components
- mechanisms to reconfigure parts of the architecture in real-time

MAUVE Components

a component is an entity that supports an activity, i.e. a real-time behavior, composed of:

- a Shell (in/out ports and properties)

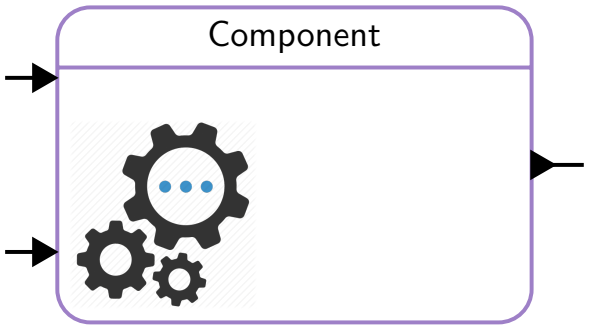



```
struct HelloShell: public Shell {
  Property<string>& who =
    mk_property("who", "World");
  WritePort<string>& talk =
    mk_port<WritePort<string>>("talk");
};
```

MAUVE Components

a component is an entity that supports an activity, i.e. a real-time behavior, composed of:

- a Shell (in/out ports and properties)
- a Core (processing algorithms)

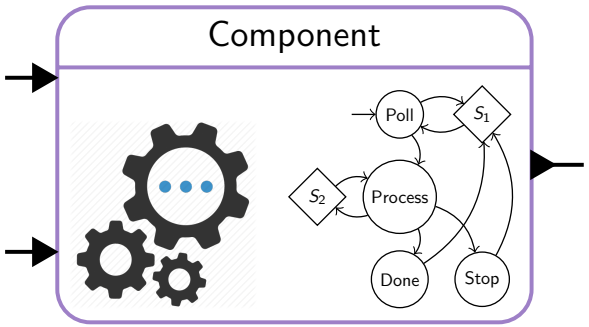



```
struct HelloCore : public Core<HelloShell> {
  void update() {
    string speech = "Hello " + shell().who;
    shell().talk.write(speech);
  };
};
```

MAUVE Components

a component is an entity that supports an activity, i.e. a real-time behavior, composed of:

- a Shell (in/out ports and properties)
- a Core (processing algorithms)
- a Finite State-Machine (behavior)

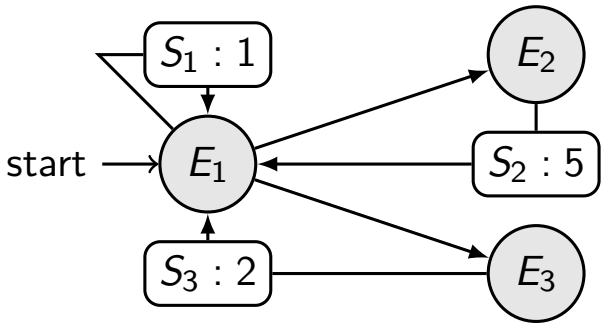



```
struct HelloFSM :
  public FSM<HelloShell,HelloCore>
{
  ExecState<HelloCore>& U =
    mk_execution("U", &HelloCore::update);
  SyncState<HelloCore>& S =
    mk_synchronization("S", sec_to_ns(1));
  bool configure_hook() {
    set_initial(U);
    set_next(U,S);
    set_next(S,U);
    return true;
  };
};
```

MAUVE Finite State-Machines

enhanced clock-based FSM to model complex behaviors:

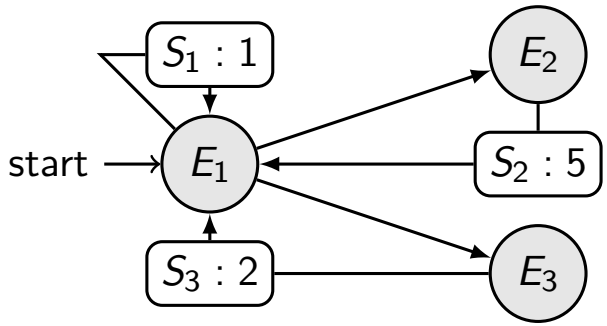
- Execution states (call a Core function)
- Synchronization states (pause and wait for a clock trigger)

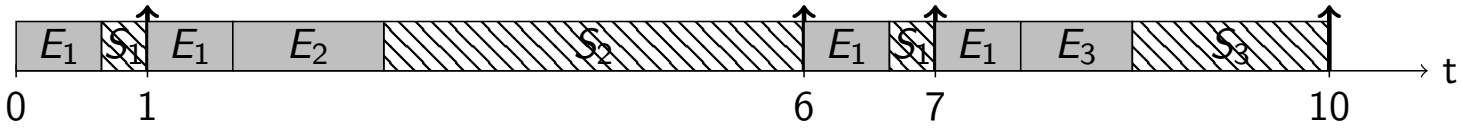


MAUVE Finite State-Machines

enhanced clock-based FSM to model complex behaviors:

- Execution states (call a Core function)
- Synchronization states (pause and wait for a clock trigger)

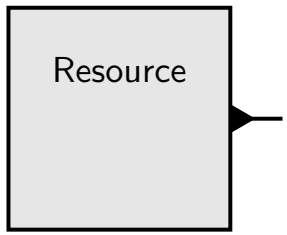




MAUVE Resources

a resource is an entity that owns data and/or provides a library of functions, composed of:

- a Shell (in/out ports and properties)

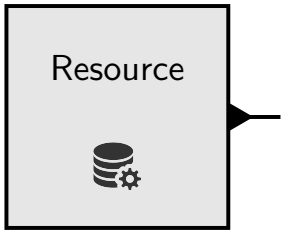



```
template <typename T>
struct SharedDataShell: public Shell {
  Property<T> & default_value;
};
```

MAUVE Resources

a resource is an entity that owns data and/or provides a library of functions, composed of:

- a Shell (in/out ports and properties)
- a Core (library and data)

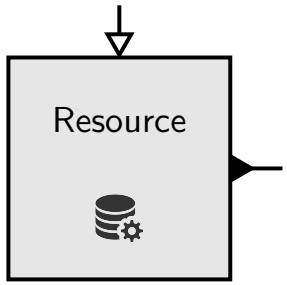



```
template <typename T>
struct SharedDataCore:
  public Core<SharedDataShell<T>>
{
  T read_value();
  void write(T value);
  void reset();
private:
  T value;
  RtMutex mutex;
};
```

MAUVE Resources

a resource is an entity that owns data and/or provides a library of functions, composed of:

- a Shell (in/out ports and properties)
- a Core (library and data)
- an Interface (services to call library and access data)




```
template <typename T>
struct SharedDataInterface:
  public Interface<SharedDataShell<T>,
       SharedDataCore<T>>
{
  EventService & reset =
    mk_event_service("reset", &SharedDataCore::
        reset);
  ReadService<T> & read_value =
    mk_read_service<T>("read_value", &
        SharedDataCore::read_value);
  WriteService<T> & write =
    mk_write_service<T>("write", &SharedDataCore
        ::write);
};
```

MAUVE Resources

a resource is an entity that owns data and/or provides a library of functions

- no specific activity: Core functions are executed by the service caller thread
- a way to implement any communication pattern (push/pull, queries, databases, ...)
- MAUVE Runtime provides `SharedData` and `RingBuffer` standard resources to connect component's ports

MAUVE Architectures

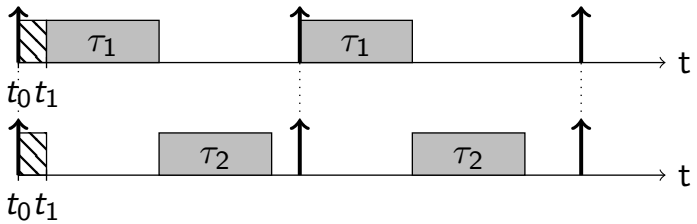
an architecture instantiates components and resources, configure and connect them

```
struct HelloArchitecture : public Architecture {
  Component<HelloShell,HelloCore,HelloFSM>& hello =
    mk_component<HelloShell,HelloCore,HelloFSM>("hello");
  SharedData<string>& text =
    mk_resource<SharedData<string>>("text", "");
  bool configure_hook() {
    hello.shell().talk.connect(text.interface().write);
    hello.shell().who = " you";
    text.configure();
    hello.configure();
    return true;
  };
};
```

MAUVE Deployment

- a real-time task is created to execute each component
- components clocks are synchronized

```
int main() {
  auto a = new HelloArchitecture();
  auto d = mk_deployer(a);
  a->configure();
  d->create_tasks();
  d->activate();
  d->start();
  d->loop();
  d->stop();
  a->cleanup();
};
```



Reconfiguration

- Reconfiguration mechanisms:
  - change properties values
  - replace a part of an element; most common uses:
    - replace a FSM: changes the behavior of the component while keeping the same functions
    - replace a Core: changes the computational part (switch between algorithms)
- Reconfiguration implementation:
  - always needs a `cleanup` then `configure` of the container
  - preserves the real-time synchronization of the components

MAUVE Toolchain

- Runtime additional features:
  - a Python binding for interactive deployment
  - A logging library (using `spdlog`)
  - A tracing library (using `lttng`)

MAUVE Toolchain

- A full Toolchain including:
  - base components, such as PSM, Mux/Demux resources, ...
  - base types (light version of ROS messages)
  - ROS Publisher/Subscriber resources (with type conversion)
  - tracing library with probabilistic WCET estimation[2]
  - WCRT analysis algorithms for Periodic SM[3]

[2] Nicolas Gobillot, Fabrice Guet, David Doose, Christophe Grand, Charles Lesire, and Luca Santinelli (2016). "Measurement-based real-time analysis of robotic software architectures". In: *IROS*. Daejeon, South Korea.

[3] Nicolas Gobillot, David Doose, Charles Lesire, and Luca Santinelli (2015). "Periodic state-machine aware real-time analysis". In: *ETFA*. Luxembourg.

MAUVE Toolchain

- Open-Source software under LGPL Free Software Free as in Freedom
- Catkin-based install ROS Enabled
- Reference manual and tutorials

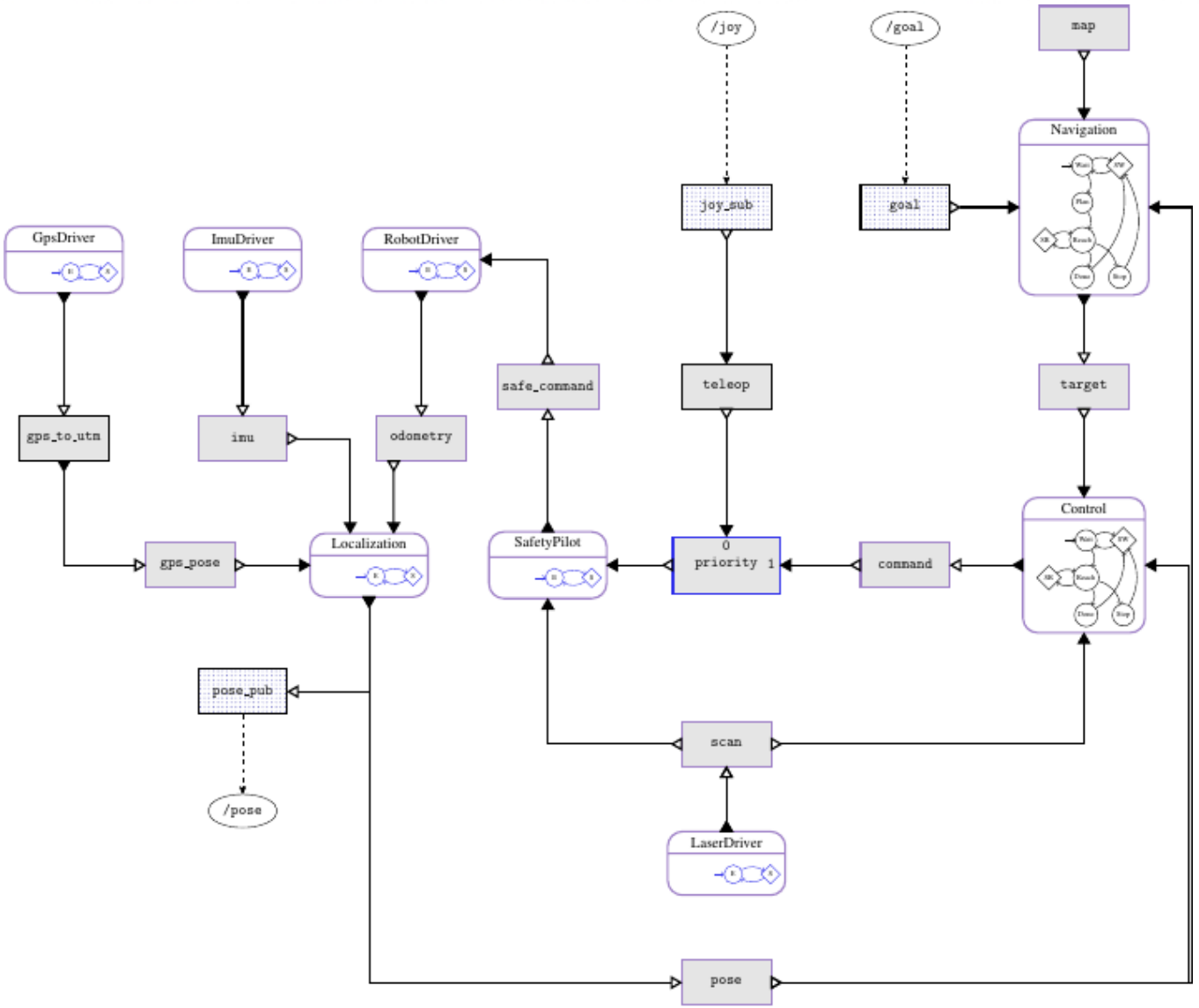
`https://gitlab.com/mauve/mauve_toolchain/wiki`

give it a try! we provide support

Application

- A navigation architecture on a ground robot
- Multi-sensor fusion for localization
- Reconfiguration of the localization architecture when GPS breakdown

Application



Conclusion

- A runtime with an expressive while deterministic execution model
- Reconfiguration mechanisms that preserve the execution model
- Simple concepts consistent with Component-Based Software Engineering and traditionnal Robotics Middlewares
- A toolchain to help fast development of reconfigurable and modular architectures, and analysis

Ongoing work

- Evolution towards architecture composition
- Managers to supervise and reconfigure sub-architectures
- Adapt the Periodic SM WCRT analysis to Clock-based FSM
- Application to other platforms

https://gitlab.com/mauve/mauve_toolchain/wiki