RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

Formal Methods for Verification and Control of Robotic Software Architectures

Charles Lesire

ONERA/DTIS, Université de Toulouse

Séminaire R4 – 04/04/2022

Outline

1. Introduction
2. Component-based Middleware and Schedulability
3. DSL for Skills Modeling
4. Petri-net based Mission Programming
5. Conclusion

Context

Photo by Matthew Henry

Context

mission programming

Photo by Matthew Henry

©Charles Lesire

Context

mission programming

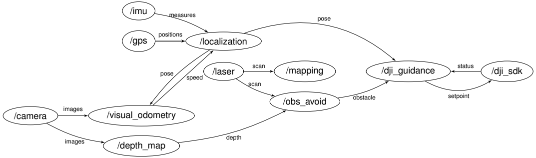
©Charles Lesire

mission-level guarantees?

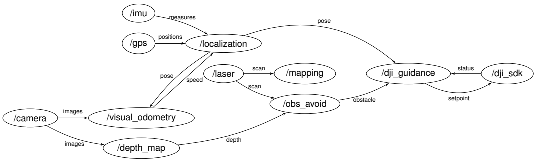
Photo by Matthew Henry

software-level guarantees?

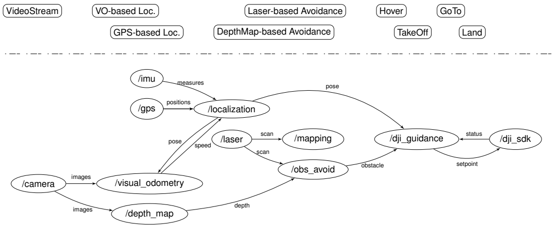
Contributions



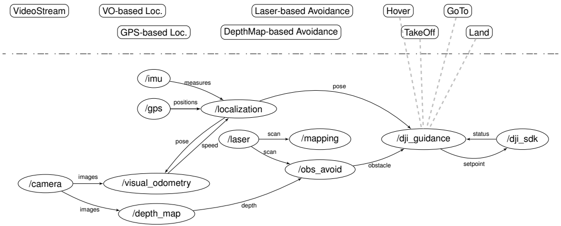
Contributions



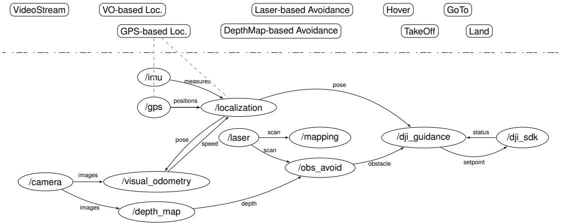
Contributions



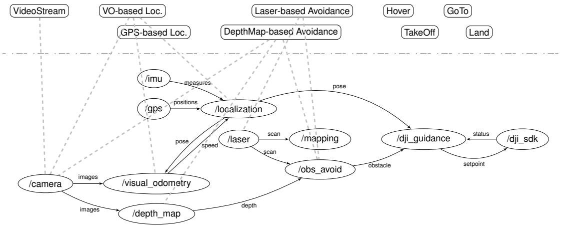
Contributions



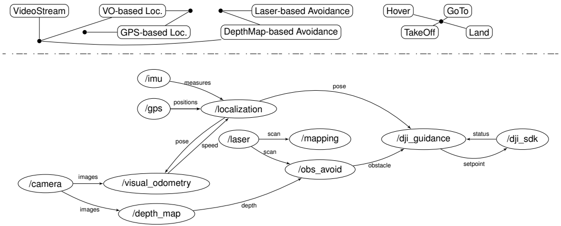
Contributions



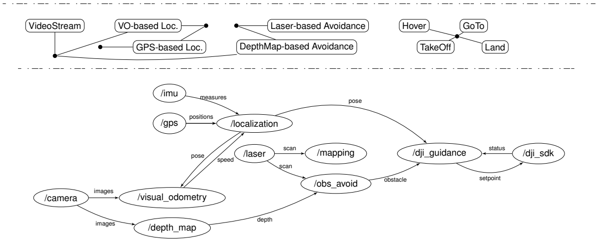
Contributions



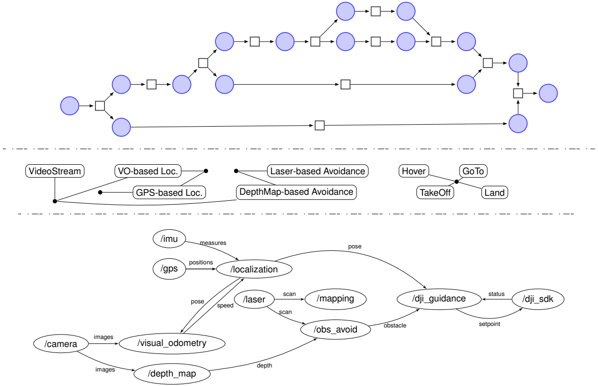
Contributions



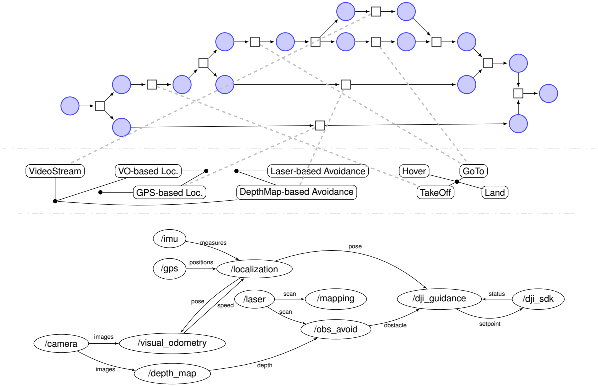
Contributions



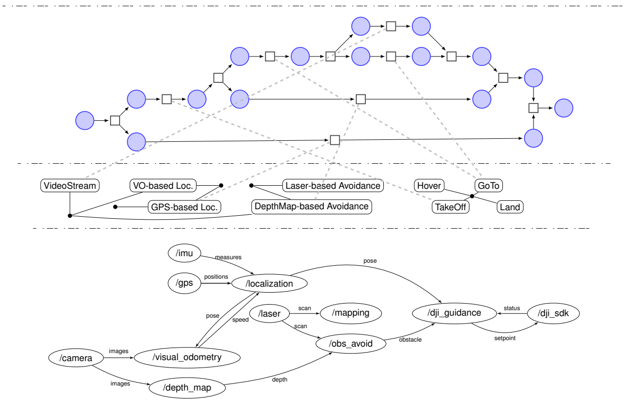
Contributions



Contributions

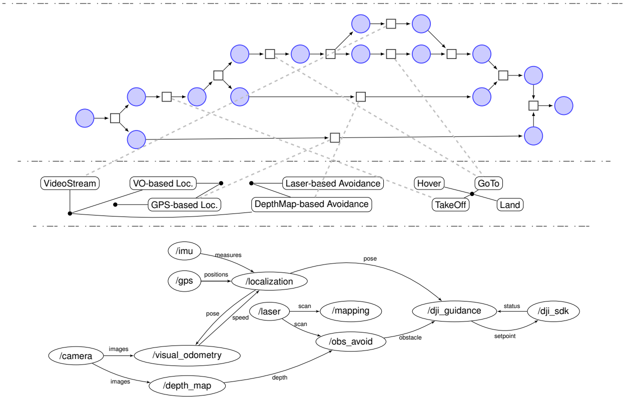


Contributions



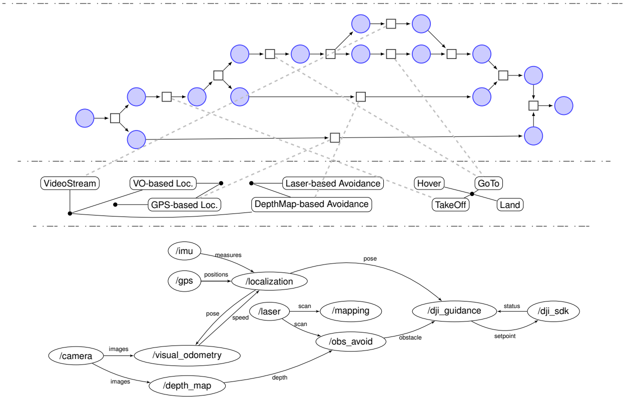
Contributions

GPS_loc || (takeOff ; (laserAvoid || (goTo(wp$_0$) ; (goTo(wp$_1$) || stream))))



Contributions

GPS_loc || (takeOff ; (laserAvoid || (goTo(wp$_0$) ; (goTo(wp$_1$) || stream))))



ASPiC

Robot Skills

MAUVE Toolchain

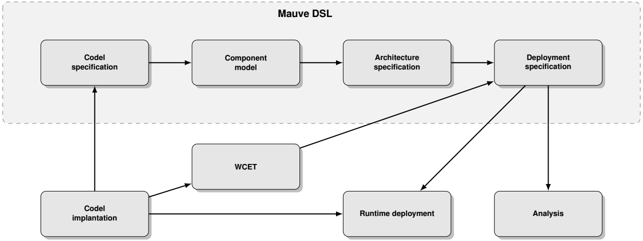
Outline

The MAUVE Toolchain

- MAUVE DSL to model component-based architectures
- MAUVE Runtime to execute real-time architectures
- MAUVE RT Analyses to evaluate WCET/WCRT of components

MAUVE DSL and Toolchain

- Architecture modeling using a DSL[1]
- Code Generation (ROS/OROCOS)
- Model Analysis compliant with Execution Analysis



[1] Nicolas Gobillot et al. "A Design and Analysis Methodology for Component-Based Real-Time Architectures of Autonomous Systems". In: *J. Intell. Robotic Syst.* 96.1 (2019), pp. 123–138. DOI: 10.1007/s10846-018-0967-5.

MAUVE Runtime

MAUVE Runtime[2] rationales:

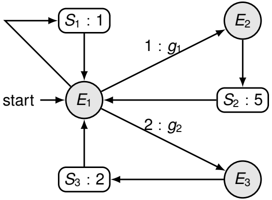
1. Provide a C++ API for programmers compliant with models
2. Provide reconfiguration mechanisms
3. Masterize the synchronization of real-time tasks
4. Provide an execution model both formal (to ease analyses) and expressive (to allow implementing complex behaviors)

[2] David Doose et al. "MAUVE Runtime: A Component-Based Middleware to Reconfigure Software Architectures in Real-Time". In: *IEEE International Conference on Robotic Computing, (IRC)*. Taichung, Taiwan: IEEE Computer Society, 2017. DOI: 10.1109/IRC.2017.47.

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

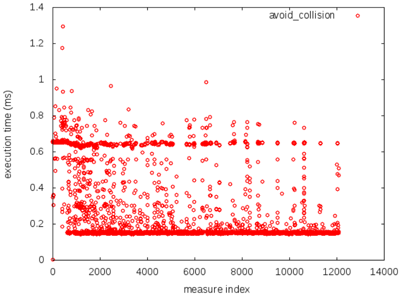


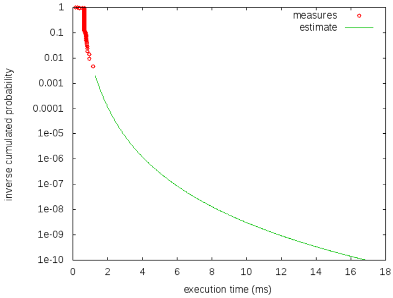
MAUVE RT analysis

- Schedulability analysis:
  - Determine if the software components are executed on time.
- Worst Case Execution Time (WCET):
  - longest CPU time passed to compute a piece of code without any interaction (alone on the CPU)
  - depends on both the source code and the hardawre.
  - an input for the schedulability analysis and the computation of the WCRT
- Worst Case Response Time (WCRT):
  - longest time spent by a “task” from its beginning to its end
  - takes into account preemptions and delays from other “tasks”
  - A “task” is schedulable is its WCRT est lower or equals to its deadline

MAUVE RT analysis

- WCET using probabilistic analysis[3]
  - gather real executions traces of the robot
  - Extreme Value Theory (EVT) to infer rare events
  - computes a probabilistic WCET
  - metrics on the applicability of the theory

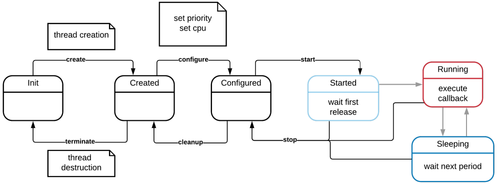


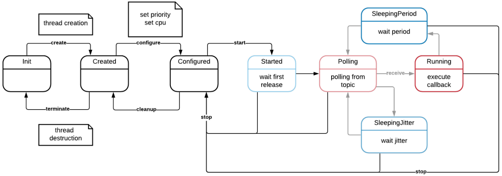


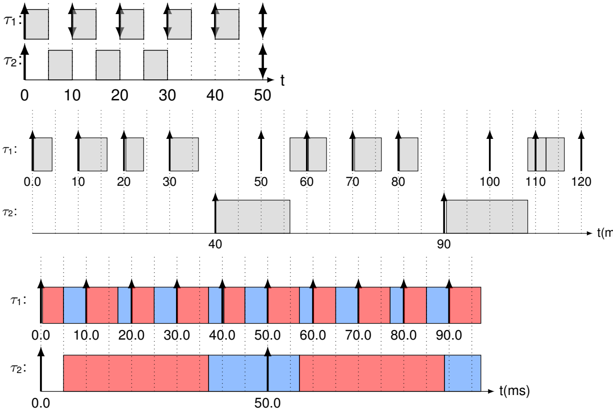
[3] Nicolas Gobillot et al. "Measurement-based real-time analysis of robotic software architectures". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, 2016. DOI: 10.1109/IROS.2016.7759509.

Corail / ROS2

- Integration of the RT core as a ROS2 extension[4]







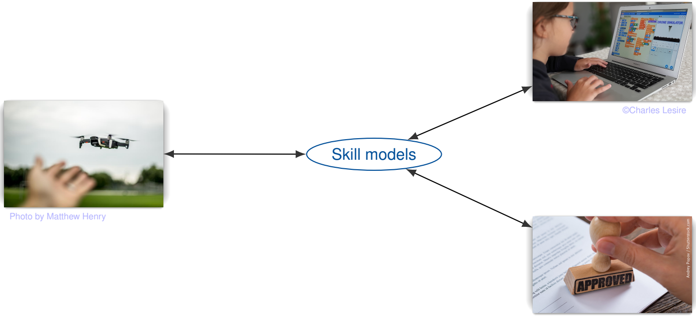
[4] Benoit Varillon et al. “Corail: Real-Time ROS2”. In: *ROS World*. 2021.

Outline

Robot Skills Models

- Organize the software architecture into functions, or skills

⇝ DSL for Skill Modeling... and tools[5]

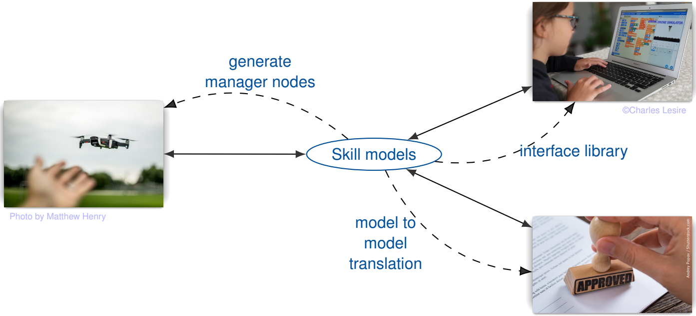


[5] Charles Lesire et al. "Formalization of Robot Skills with Descriptive and Operational Models". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA, 2020.

Robot Skills Models

- Organize the software architecture into functions, or skills

⇝ DSL for Skill Modeling... and tools[5]



[5] Charles Lesire et al. “Formalization of Robot Skills with Descriptive and Operational Models”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA, 2020.

Robot Skills DSL

Functional layer organized into SkillSets (e.g., `guidance`, `image_processing` or `PR2`) ;
a SkillSet defines:

- Data provided by the skillset / robot;
- Functions that can be used on other parts of the model;
- Resources, represented as state-machines, that can be used by skills, or that reflect actual state of a system (e.g., a driver);
- Skills, i.e. functions provided by the skillset.

Robot Skills DSL

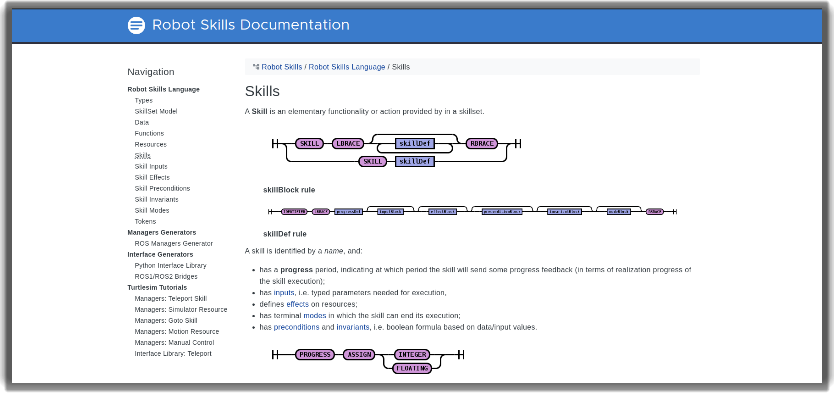
A Skill is defined by:

- some inputs, i.e. parameters of the skill execution (e.g., position to reach);
- a precondition describing some logical condition for accepting execution;
- the list of used resources, with constraints (pre, invariants);
- the terminal modes of the skills, each described by post-conditions and effects on resources;
- a progress period at which the skill will report some progress.

Robot Skills DSL

Full language grammar described at:

`http://oara-architecture.gitlab.io/robot-skills/robot_lang/`



Robot Skills Documentation

Navigation

Robot Skills Language
- Types
- SkillSet Model
- Data
- Functions
- Resources
- Skills
- Skill Inputs
- Skill Effects
- Skill Preconditions
- Skill Invariants
- Skill Modes
- Tokens

Managers Generators
- ROS Managers Generator

Interface Generators
- Python Interface Library
- ROS1/ROS2 Bridges

Turtlesim Tutorials
- Managers: Teleport Skill
- Managers: Simulator Resource
- Managers: Goto Skill
- Managers: Motion Resource
- Managers: Manual Control
- Interface Library: Teleport

Robot Skills / Robot Skills Language / Skills

Skills

A **Skill** is an elementary functionality or action provided by in a skillset.

skillBlock rule

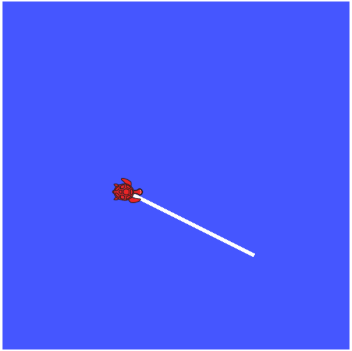
skillDef rule

A skill is identified by a *name*, and:

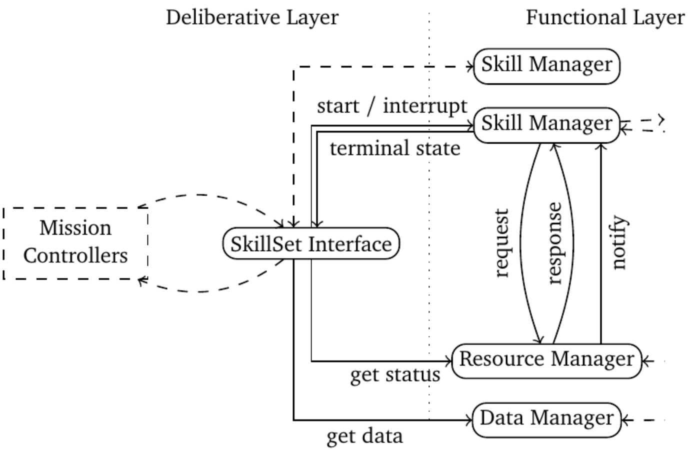
- has a **progress** period, indicating at which period the skill will send some progress feedback (in terms of realization progress of the skill execution);
- has inputs, i.e. typed parameters needed for execution,
- defines effects on resources;
- has terminal modes in which the skill can end its execution;
- has preconditions and invariants, i.e. boolean formula based on data/input values.

Robot Skills DSL: Turtlesim

```
type {
  Pose
  Pen
  Bool
}

skillset turtlesim {
  data position: Pose

  function isSafe(p: Pose): Bool

  resource simulator {
    initial UNAVAILABLE
    extern UNAVAILABLE -> AVAILABLE
    extern AVAILABLE -> UNAVAILABLE
  }

  skill teleport {
    progress=0
    input target: Pose
    precondition sim_avail: resource=(simulator==AVAILABLE)
    invariant sim_inv: resource=(simulator==AVAILABLE)
    mode {
      ARRIVED {}
      COLLISION {}
    }
  }
```



Robot Skills Execution Tools



Robot Skills Managers

Managers

- Data manager subcribes to internal data and provide getters
- Resource manager keeps track of the resources states, from skills request or from external requests
- Skill managers implement the management of skills in relation to functional modules

Generators

- ROS2 Generator
- One package for the ROS `msg`, one for skillset management
- Inherit from the skillset manager to implement the link with the functional modules
- Type mapping between model types and ROS types
- Skills can be triggered using ROS topics

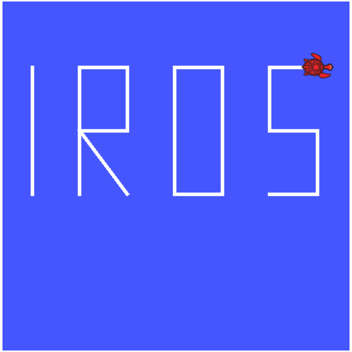
Robot Skills ClientInterface

Client Library

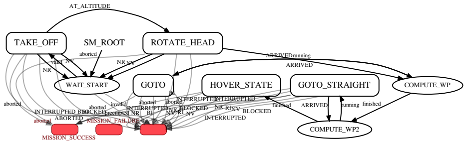
- ROS/Python library to get access to data and ressources, and control skills
- Simple API that hide ROS constructs
- Generated from a skillset model

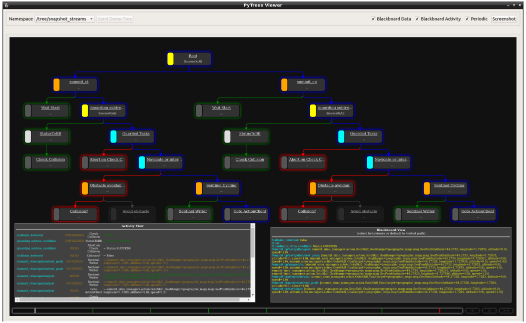
Robot Skills Interface: Turtlesim

```python
from turtlesim_interface import TurtlesimSkillsetInterface
from turtlesim.msg import Pose

turtlesim = TurtlesimSkillsetInterface()
teleport = turtlesim.skills.teleport
paint = turtlesim.skills.paint

# draw I
teleport.start(Pose(x=1, y=5))
teleport.wait()
paint.start()
teleport.start(Pose(x=1, y=9))
teleport.wait()
paint.interrupt()
paint.wait()

# draw R
teleport.start(Pose(x=3, y=5))
teleport.wait()
paint.start()
teleport.start(Pose(x=3, y=9))
teleport.wait()
teleport.start(Pose(x=5, y=9))
teleport.wait()
teleport.start(Pose(x=5, y=7))
teleport.wait()
teleport.start(Pose(x=3, y=7))
teleport.wait()
teleport.start(Pose(x=5, y=5))
teleport.wait()
paint.interrupt()
paint.wait()
```

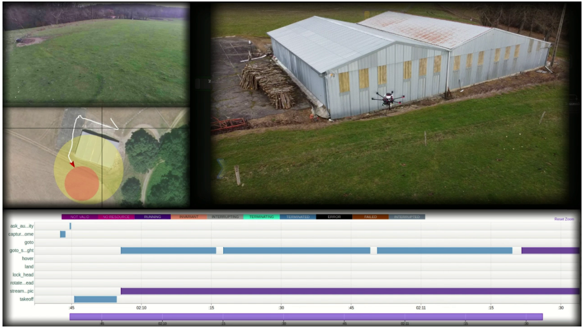


Robot Skills Interface





Safe Inspection Mission by a UVA



`https://youtu.be/mZxyl6v-tDw`

Alexandre Albore et al. “Skill-Based Architecture Development for Online Mission Reconfiguration and Failure Management”. In: *International Workshop on Robotics Software Engineering*. Madrid, Spain, 2021. DOI: 10.1109/RoSE52553.2021.00015

Outline

1. Introduction
2. Component-based Middleware and Schedulability
3. DSL for Skills Modeling
4. **Petri-net based Mission Programming**
5. Conclusion

ASPiC

- Task-level (or Mission) Programming:
  - Expressive "language" with complex operations
  - Management of failures in task execution
  - Petri nets as an underlying formal model
  - Execution control based on playing the resulting Petri net model

[6] Charles Lesire and Franck Pommereau. “ASPiC: An Acting System Based on Skill Petri Net Composition”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018. DOI: 10.1109/IROS.2018.8594328.

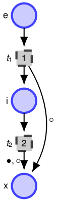
ASPiC

- Task-level (or Mission) Programming:
  - Expressive "language" with complex operations
  - Management of failures in task execution
  - Petri nets as an underlying formal model
  - Execution control based on playing the resulting Petri net model
- ASPiC[6]:
  - Petri nets with Control-Flow semantics
  - Colored PN with control-flow tokens $\bullet$ and $\circ$
  - Control-flow places $P^{\bullet\!\circ}$: entry, internal, exit
  - Modelling of *skills* using specific CFPN
  - composition operators that (tend to) preserve properties by construction

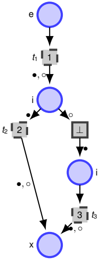
[6] Charles Lesire and Franck Pommereau. "ASPiC: An Acting System Based on Skill Petri Net Composition". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018. DOI: 10.1109/IROS.2018.8594328.

ASPiC Operators

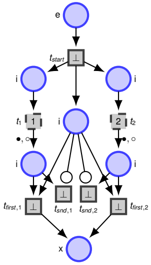
- Composition operators to build up plans from skills
- Integrate exception propagation
- Each operator modelled as a CFPN with placeholder transitions
- Operators: sequence, choice, concurrency, if/then/else, race concurrency, loop/retry



$N_{\odot}$



N_{ite}



$N_{\otimes}$

ASPiC Properties

- Well-formedness
  - preserved when connecting with a handler
  - preserved by operators
- Cleanness of control-flow

$$\forall M : M_0 \xrightarrow{*} M \wedge M(p_x) \neq \emptyset \Rightarrow$$
$$M(p_x) \in \{\{\bullet\}, \{\circ\}\} \wedge (p \in P^{\mathbb{C}} \setminus \{p_x\} \Rightarrow M(p) = \emptyset)$$

- Control-safe: $P^{\mathbb{C}}$ is safe
- Cleanness and control-safety are not ensured by all operators (typically $N_{\otimes}$)

Conclusion

Being able to guarantee a correct behavior, from mission-level to functional components, with models compliant to the actual execution.

Thank you for your attention.

Questions?

References

- Albore, Alexandre et al. "Skill-Based Architecture Development for Online Mission Reconfiguration and Failure Management". In: *International Workshop on Robotics Software Engineering*. Madrid, Spain, 2021. DOI: 10.1109/RoSE52553.2021.00015.
- Doose, David et al. "MAUVE Runtime: A Component-Based Middleware to Reconfigure Software Architectures in Real-Time". In: *IEEE International Conference on Robotic Computing, (IRC)*. Taichung, Taiwan: IEEE Computer Society, 2017. DOI: 10.1109/IRC.2017.47.
- Gobillot, Nicolas et al. "A Design and Analysis Methodology for Component-Based Real-Time Architectures of Autonomous Systems". In: *J. Intell. Robotic Syst.* 96.1 (2019), pp. 123–138. DOI: 10.1007/s10846-018-0967-5.
- Gobillot, Nicolas et al. "Measurement-based real-time analysis of robotic software architectures". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, 2016. DOI: 10.1109/IROS.2016.7759509.
- Lesire, Charles and Franck Pommereau. "ASPiC: An Acting System Based on Skill Petri Net Composition". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018. DOI: 10.1109/IROS.2018.8594328.