Code Reusability Tools for Programming Mobile Robots

Carle Côté, Dominic Létourneau, François Michaud, <u>Jean-Marc Valin</u>, Yannick Brosseau, Clément Raïevsky, Mathieu Lemay, Victor Tran

LABORIUS – Department of Electrical Engineering and Computer Engineering

Université de Sherbrooke, Québec, CANADA

{Carle.Cote, Dominic.Letourneau, Francois.Michaud, Jean-Marc.Valin}@USherbrooke.ca

Current Programming Needs

- Need to integrate many capabilities to work as a whole
- Need to reuse improvements made in each associated research field (Obstacle avoidance, navigation, localization, mapping, planning, modeling, recognition, searching, tracking, interaction, cooperation, decision-making, ...)
- Need a way not to reinvent the wheel every time we have to program a robot
- Need to share implementations with others

Integration & Reusability Issues

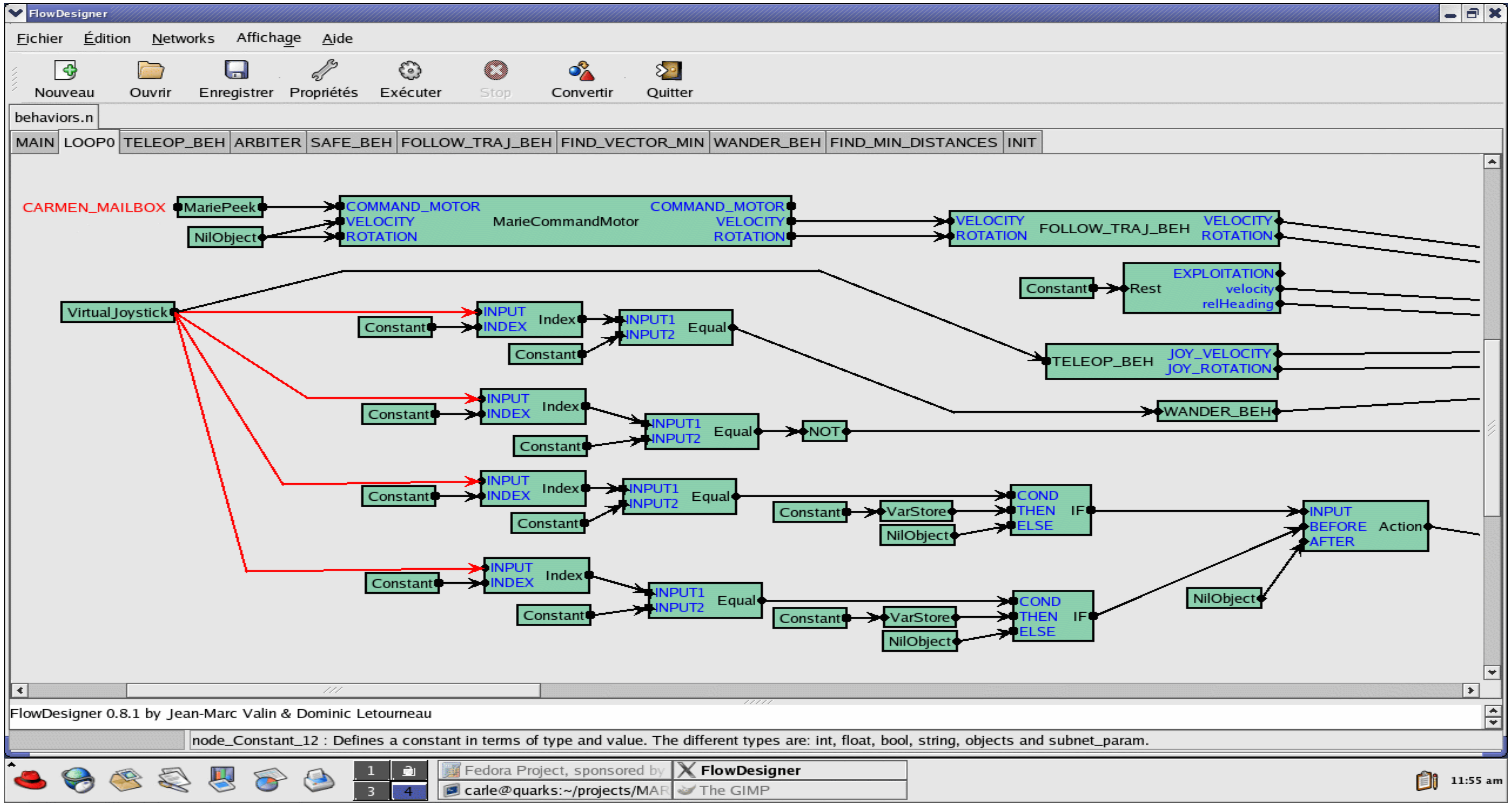
- Lots of robotics platforms, operating systems and programming environments
- Lots of software and algorithms available but mostly incompatible (Player/Stage/Gazebo, CARMEN, OROCOS, MATLAB/Simulink, ...)
- Lack of standards
- Too soon to freeze choices, limit exploration
- Technologies are in constant evolution

UNIVERSITÉ DE SHERBROOKE

IMSI INSTITUT DES MATÉRIAUX ET SYSTÈMES INTELLIGENTS / INTELLIGENT MATERIALS AND SYSTEMS INSTITUTE

LABORIUS

FlowDesigner / RobotFlow


FlowDesigner
Fichier Édition Networks Affichage Aide
Nouveau Ouvrir Enregistrer Propriétés Exécuter Stop Convertir Quitter
behaviors.n
MAIN LOOP0 TELEOP_BEH ARBITER SAFE_BEH FOLLOW_TRAJ_BEH FIND_VECTOR_MIN WANDER_BEH FIND_MIN_DISTANCES INIT
CARMEN_MAILBOX
MariePeek
NilObject
COMMAND_MOTOR VELOCITY ROTATION MarieCommandMotor COMMAND_MOTOR VELOCITY ROTATION
VELOCITY ROTATION FOLLOW_TRAJ_BEH VELOCITY ROTATION
Constant
Rest EXPLOITATION velocity relHeading
VirtualJoystick
INPUT INDEX Index
INPUT1 INPUT2 Equal
TELEOP_BEH JOY_VELOCITY JOY_ROTATION
WANDER_BEH
NOT
VarStore
COND THEN ELSE IF
INPUT BEFORE AFTER Action
1 3 4
Fedora Project, sponsored by
carle@quarks:~/projects/MAR
FlowDesigner
The GIMP
11:55 am

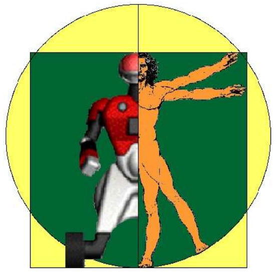
FlowDesigner 0.8.1 by Jean-Marc Valin & Dominic Letourneau

node_Constant_12 : Defines a constant in terms of type and value. The different types are: int, float, bool, string, objects and subnet_param.

FlowDesigner - Objectives

- Create a graphical data-flow processing environment
- Encapsulate functionality in blocks that can be easily reused
- Create standardized interconnections and interactions between blocks to create networks of blocks
- Support data probes and debugging tools at run-time

UNIVERSITÉ DE SHERBROOKE   IMSI INSTITUT DES MATÉRIAUX ET SYSTÈMES INTELLIGENTS INTELLIGENT MATERIALS AND SYSTEMS INSTITUTE   LABORIUS



FlowDesigner – Features (1 of 2)

- C++
- Pull and self-scheduling mechanisms
- Dynamic connection at runtime
- Super-block (Composition pattern)
- Buffered mechanism
- GUI and command line execution
- Standard datatypes and operators

FlowDesigner – Features (2 of 2)

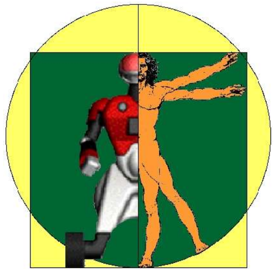
- Block creation API
- Toolkits : audio processing, artificial neural networks, fuzzy logic, visualization probes, vector quantization (VQ), and Gaussian Mixture Models (GMM)
- Linux, Solaris (limited port to Win32)

RobotFlow – Features

- Mobile robotics toolkit based on FlowDesigner containing useful blocks :
  - Pioneer2 robots interfaces
  - Device interfaces (range finder, camera, ...)
  - Behaviors and subsumption arbitration
  - Vision processing blocks
  - Player/Stage/Gazebo interfaces
  - GUI controllers (joystick, camera, ...)
  - ...

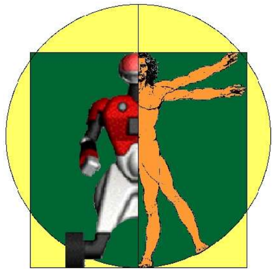
FlowDesigner / RobotFlow – Limitations

- Mostly useful when dealing with sequential (synchronous data-flow) processing
- Pull scheduling policy not well suited for asynchronous processing
- FSM and petri nets more difficult to implement
- Reuse limited to libraries
- Distributed computing not well supported yet



MARIE – Objectives

- Create a development and integration environment focused on software reusability and exploitation of already available APIs and middlewares frequently used in robotics
- Create reusability at system level by using standardized interconnections and interactions between applications
- Create a rapid-prototyping approach to software development in robotics



1) Forcing every applications to use the same communication protocol :

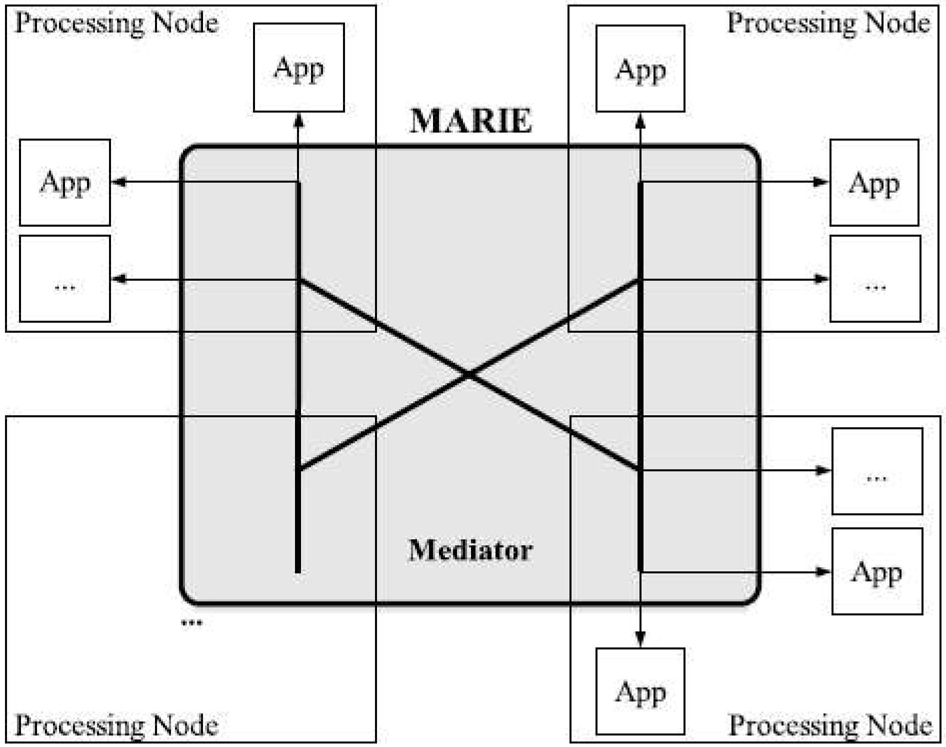
- Cannot modify proprietary code
- Might be difficult or undesirable to modify existing code
- Limits coexistence of multiple communication protocols and communication mechanisms interacting together

2) Importing functionnalities from an application to a common programming framework :

- Error-prone work that requires time, effort and knowledge
- Not good software engineering practices

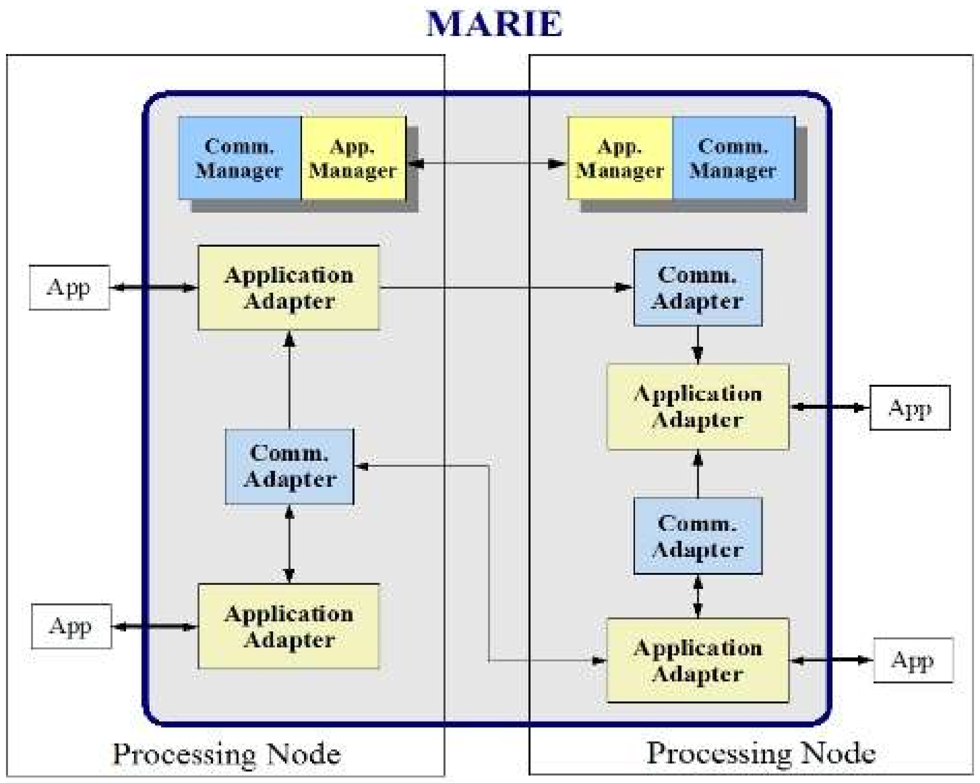
MARIE – Applying Mediator Pattern

- It is easier to change
- It decouples colleagues
- It simplifies object protocols
- It abstracts how objects cooperate
- It centralizes control

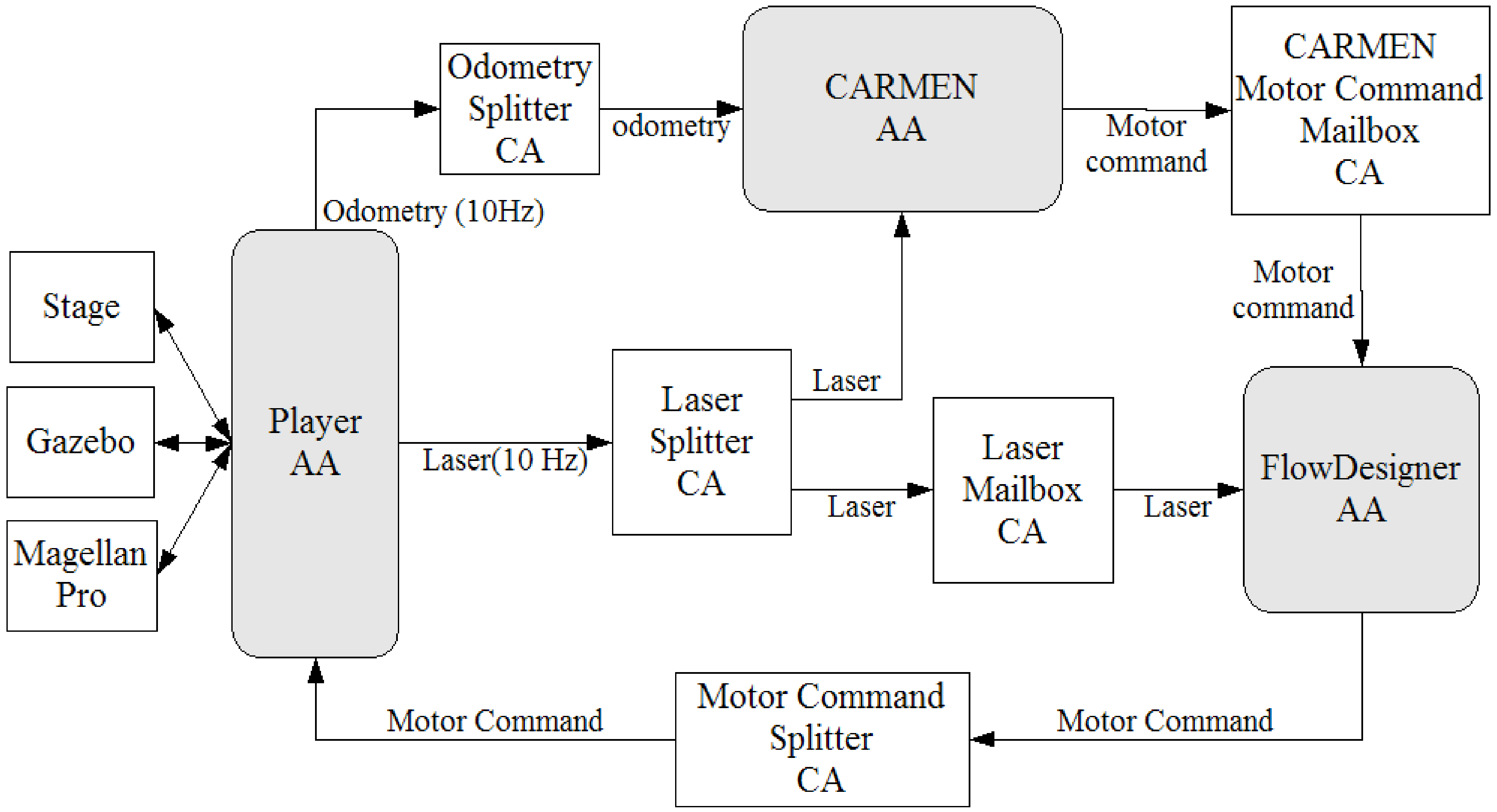


MARIE – Fonctionnal Components

- Application Adapters (AA)
- Communication Adapters (CA)
- Application Managers (AM)
- Communication Managers (CM)



MARIE

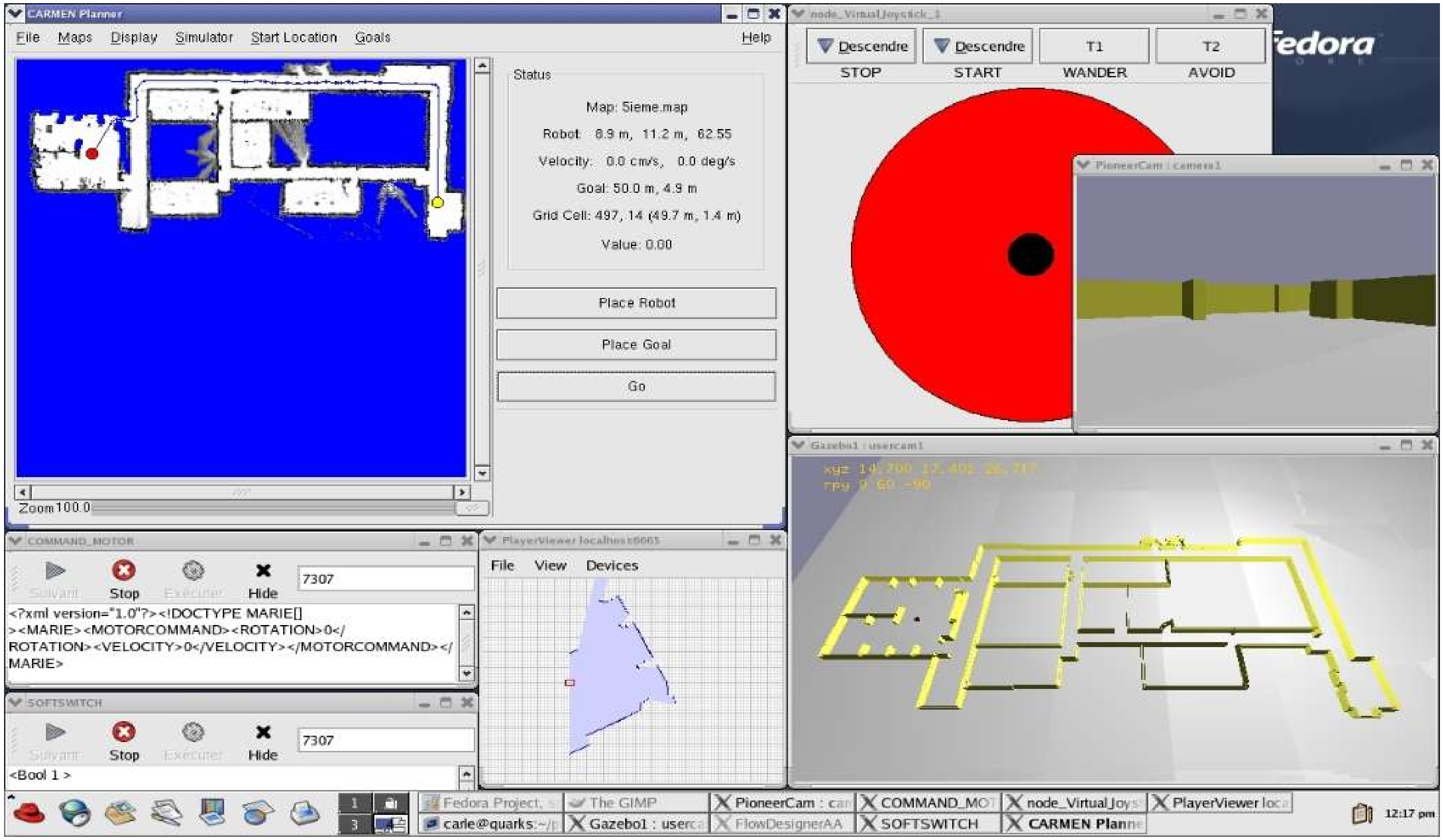

Odometry Splitter CA
odometry
CARMEN AA
Motor command
CARMEN Motor Command Mailbox CA
Odometry (10Hz)
Motor command
Stage
Gazebo
Magellan Pro
Player AA
Laser(10 Hz)
Laser Splitter CA
Laser
Laser
Laser Mailbox CA
Laser
FlowDesigner AA
Motor Command
Motor Command Splitter CA
Motor Command

UNIVERSITÉ DE SHERBROOKE

IMSI INSTITUT DES MATÉRIAUX ET SYSTÈMES INTELLIGENTS INTELLIGENT MATERIALS AND SYSTEMS INSTITUTE

LABORIUS

MARIE



Status

Map: Sieme.map

Robot: 8.9 m, 11.2 m, 62.55

Velocity: 0.0 cm/s, 0.0 deg/s

Goal: 50.0 m, 4.9 m

Grid Cell: 497, 14 (49.7 m, 1.4 m)

Value: 0.00

7307

```
<?xml version="1.0"?><!DOCTYPE MARIE[]
><MARIE><MOTORCOMMAND><ROTATION>0</
ROTATION><VELOCITY>0</VELOCITY></MOTORCOMMAND></
MARIE>
```

7307

```
<Bool 1 >
```

MARIE - Limitations

- System performances might be affected by code overhead
- Coherent and stable system might be difficult to achieve with many heterogeneous applications interacting
- Applications to integrate must have a clear method of interactions (API, communication links, files, ...)
- System resources (memory, drivers, hardware, ...) might be impossible to manage correctly

Conclusion

- Approaches to enhance code reusability : FlowDesigner (functional level) and MARIE (system level)
- Importance of code reusability :
  - Allows to communicate knowledge and implementation results
  - Allows exchange of ideas by sharing implementations
  - Accelerates exploration of novel ways to integrate capabilities
  - Scientific process of studying intelligence in autonomous systems.
    - MARIE : http://marie.sourceforge.net
    - FlowDesigner : http://flowdesigner.sourceforge.net
    - RobotFlow : http://robotflow.sourceforge.net