The Fawkes Robot Software Framework
Software Stack for the RoboCup Logistics League

Matthias Löbach
Tim Niemueller
Carologistics

- RoboCup participant since 2012
- Joint team of FH Aachen University of Applied Sciences & RWTH Aachen University
- Winner of the RoboCup Logistics League 2014, 2015 and 2016
1. RoboCup Logistics League
2. Fawkes Robot Software Framework
3. Software Components
4. Conclusion
RoboCup Logistics League - At a Glance

RoboCup Logistics League (RCLL)

- In-factory manufacturing logistics in Smart Factory
- Maintain and optimize material flow in production
- Multi-Robot planning/scheduling and coordination

Goals

- Production logistics autonomy
- Benchmarking of robots in a Smart Factory
- *Long-term autonomy* with reasoning/planning systems
RoboCup Logistics League

- Physical processing machines based on Festo MPS
- Almost 250 product variants
RoboCup Logistics League

- Team colors: cyan and magenta
- Exclusive machines randomly spread across field
- Mirrored at middle axis
RoboCup Logistics League - Game Phases

Setup Phase
- Duration: 5 minutes
- Teams prepare robots in insertion zone

Exploration Phase
- Duration: 4 minutes
- Points for correct light signal reports
- Points for correct machine position

Production Phase
- Duration: 15 minutes
- Points for delivered products
- Points for intermediate steps
Mobile production machines

- Conveyor as contact point for robots
- AR-Tag for identification and localization
- Signal light for identification and status
RoboCup Logistics League - Robots

Robotino 3 platform
- Laser range finder
- Additional laptop
- Custom gripper
- Cameras

Signal Camera
- Custom Gripper
- Additional Laptop
- Conveyor Camera
- Marker Camera
- Internal Computer
- Laser Scanner
RoboCup Logistics League - Referee Box

- **Game Control**
  - Maintain game state/score

- **Communication**
  - Publish production plans

- **Data Recording**
  - Collect benchmarking data

- **Visualization and Instruction**
  - Referee/visitor monitoring

- **Machinery Control**
  - Instruct field machines

Logs game information and all communication
RoboCup Logistics League – Production Example

Get base with cap
Fill cap station with cap
Bring surplus base to ring station as resource
Retrieve base with ordered color at base station
Insert base into cap station for assembly
Retrieve assembled product from cap station
Deliver product at delivery station

Most simple product has seven steps
RoboCup Logistics League – Production Example

- Get base with cap
RoboCup Logistics League – Production Example

- Get base with cap
- Fill cap station with cap
- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
RoboCup Logistics League – Production Example

- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
- Retrieve base with ordered color at base station
- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
- Retrieve base with ordered color at base station
- Insert base into cap station for assembly
RoboCup Logistics League – Production Example

- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
- Retrieve base with ordered color at base station
- Insert base into cap station for assembly
- Retrieve assembled product from cap station
- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
- Retrieve base with ordered color at base station
- Insert base into cap station for assembly
- Retrieve assembled product from cap station
- Deliver product at delivery station
RoboCup Logistics League – Production Example

- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
- Retrieve base with ordered color at base station
- Insert base into cap station for assembly
- Retrieve assembled product from cap station
- Deliver product at delivery station
RoboCup Logistics League – Production Example

- Get base with cap
- Fill cap station with cap
- Bring surplus base to ring station as resource
- Retrieve base with ordered color at base station
- Insert base into cap station for assembly
- Retrieve assembled product from cap station
- Deliver product at delivery station

Most simple product has seven steps
1. RoboCup Logistics League

2. Fawkes Robot Software Framework

3. Software Components

4. Conclusion
Middleware

- Interconnect software components
- Make data accessible and observable
- Structure the data
Integration

Middleware

- Interconnect software components
- Make data accessible and observable
- Structure the data

Framework

- Run-time organization and execution flow
- Assert certain properties of the system
Integration

Middleware
- Interconnect software components
- Make data accessible and observable
- Structure the data

Framework
- Run-time organization and execution flow
- Assert certain properties of the system

Toolbox
- Provide libraries for typical robotics task
- Integrate libraries and make available through framework
Fawkes at a Glance

Fawkes

- Component-based architecture (plugins)
- Hybrid BlackBoard/messaging data exchange
- Multi-threaded and distributable
- Aspect-oriented framework feature access
- Structured and synchronized main loop
- Automated coordinate transform system
- Web interface for introspection and control
Run-time Coordination

- Fawkes provides a main loop
- Threads *can* be hooked into the main loop
- Threads *can* also run concurrently
- Main loop is replaceable
- Threads for each hook are woken up concurrently
- Threads sleep during execution of other hooks
Fawkes provides a main loop
Threads *can* be hooked into the main loop
Threads *can* also run concurrently
Main loop is replaceable
Threads for each hook are woken up concurrently
Threads sleep during execution of other hooks
BlackBoard created by Fawkes main application
BlackBoard created by Fawkes main application

- Interface storage in the BlackBoard memory
- Interface definition via XML (fields/messages)
Fawkes Threads access the BlackBoard via these Interfaces
Fawkes Threads access the BlackBoard via these Interfaces
Fawkes Threads access the BlackBoard via these Interfaces
Remote applications can access BlackBoard via network
Transactions (read/write)
Only one writer at a time
- Message passing as command channel
- Messages can only be sent from reader to writer
- Any number of messages in queue
Plugins

Components

- Provides specific functionality
- Ideally: parameterizable blackbox
- Can – ideally – be easily replaced
- Interconnected through middleware
Components

- Provides specific functionality
- Ideally: parameterizable blackbox
- Can – ideally – be easily replaced
- Interconnected through middleware

Plugins

- Dynamically loadable libraries
- Set of threads and their initialization
- Framework can introspect threads

Soft Guarantee for Plugins

⇒ Either all threads are successfully initialized, or none is ever started
Plugins

Components
- Provides specific functionality
- Ideally: parameterizable blackbox
- Can – ideally – be easily replaced
- Interconnected through middleware

Plugins
- Dynamically loadable libraries
- Set of threads and their initialization
- Framework can introspect threads

Soft Guarantee for Plugins
⇒ Either all threads are successfully initialized, or none is ever started
Plugins

Components
- Provides specific functionality
- Ideally: parameterizable blackbox
- Can – ideally – be easily replaced
- Interconnected through middleware

Plugins
- Dynamically loadable libraries
- Set of threads and their initialization
- Framework can introspect threads

Soft Guarantee for Plugins
⇒ Either all threads are successfully initialized, or none is ever started

LaserFilterPlugin

FilterThread 1

FilterThread 2

FilterThread 3

... (dynamically allocated based on config)
Aspects

Framework Features

- Threads must access features
- Classic: inquire/get features
- Control executed by requester
  ⇒ Framework has only limited information
Aspects

Framework Features

- Threads must access features
- Classic: inquire/get features
- Control executed by requester
  ⇒ Framework has only limited information

Aspects

- Plugin threads denote requirements
- Framework initializes aspects
- Soft guarantee of feature availability
Aspects

Framework Features

- Threads must access features
- Classic: inquire/get features
- Control executed by requester
  ⇒ Framework has only limited information

Aspects

- Plugin threads denote requirements
- Framework initializes aspects
- Soft guarantee of feature availability

Implementation

- Aspect as simple class
- Threads inherit aspect class
- Framework asserts initialization
Transforms
Transforms

- Various coordinate reference frames
- Keep data in local frame for accuracy
- Need transformations to act on sensed objects
- Generalized transformation system (ported from ROS tf2)
- Tree of linked frames
Transforms Graph

- Represent transforms as graph (tree)
- Walk the tree to calculate transform
- Performs time matching and interpolation
- API to transform points, poses etc.
- Static vs. dynamic transforms
- API from C++, Lua, and CLIPS
Behavioral Architecture

Agent

Behavior Engine

Localization

Motion

Vision

...
Behavioral Architecture

Agent

Behavior Engine

- Localization
- Motion
- Vision
- ...

Blackboard or ROS Topics
Welcome to Fawkes.

Latest log messages

23:56:41.560604  CLIPS (agent)  Accept already locked GET
23:56:41.560870  CLIPS (agent)  Calling still bring_product_to(place="C-CS1")
23:56:41.560926  ClipsAgentThread  Calling still bring_product_to(place="C-CS1")
23:56:41.561148  CLIPS (agent)  Sent mps-instruction for C-CS1
23:56:41.563514  SkillerLua  Sending CartesianGotoMessage(x=-2.124119, y=4.975472, z=-2.980623)
23:56:41.564817  NavGraphThread  cartesian goto (x,y,z) = (-2.124119,4.975472,-2.980623)
23:56:41.565446  NavGraphThread  Starting route: C-CS1-1 - free-target
23:56:41.566627  NavGraphThread  Sending next goal on plan start
23:56:41.566907  NavGraphThread  Node 'free-target' has been reached
23:56:41.635768  CLIPS (agent)  Skill bring_product_to is RUNNING, was: IDLE
23:56:42.089324  CLIPS (agent)  Sent mps-instruction for C-CS1
23:56:42.554319  CLIPS (agent)  Sent mps-instruction for C-CS1
23:56:43.020324  CLIPS (agent)  Sent mps-instruction for C-CS1
23:56:43.020449  CLIPS (agent)  Mps C-CS1 successfully instructed
23:56:45.015696  SkillerLua  Drive to: call global_motor_move with: x(-2.1241192817688) y(4.9754724502563) on(-2.988238868207)
23:56:45.027273  SkillerLua  dist: -0.035002, 0.305290, -0.074570
23:56:51.014941  SkillerLua  dist: -0.035002, 0.305290, -0.074570
23:56:51.015006  SkillerLua  dist: -0.035002, 0.305290, -0.074570
23:56:51.076905  SkillerLua  check_tag: Search for tag: 1
23:56:51.077046  SkillerLua  check_tag: Found tag with id: 1

Fawkes 0.5.0
### Showing `SkillerInterface::Skiller`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>skill_string</td>
<td>string</td>
<td>[1024]</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>[128]</td>
</tr>
<tr>
<td>exclusive_controller</td>
<td>uint32</td>
<td>222</td>
</tr>
<tr>
<td>msgid</td>
<td>uint32</td>
<td>0</td>
</tr>
<tr>
<td>status</td>
<td>SkillStatusEnum</td>
<td>S_INACTIVE</td>
</tr>
</tbody>
</table>

#### Interface

- `AK12GripperInterface::Gripper AK12`
- `IMUInterface::IMU Roboticino`
- `Laser360Interface::Laser cil`
- `Laser360Interface::Laser mapless`
- `Laser360Interface::Laser time`
- `Laser360Interface::Laser urg-filtered`
- `Laser360Interface::Laser Map Laser`
- `LaserClusterInterface::Laser-cluster/mps`
- `LaserClusterInterface::Laser-cluster/robots`
- `LaserLineInterface::Laser-lines/1`
- `LaserLineInterface::Laser-lines/2/moving_avg`
- `LaserLineInterface::Laser-lines/3/moving_avg`
- `LaserLineInterface::Laser-lines/4/moving_avg`
- `LaserLineInterface::Laser-lines/5/moving_avg`
- `LaserLineInterface::Laser-lines/6/moving_avg`
- `LaserLineInterface::Laser-lines/7/moving_avg`
- `LaserLineInterface::Laser-lines/8/moving_avg`
- `LocalisationInterface::AMCL`
- `MotorInterface::Roboticino`
- `NavGraphGeneratorInterface::navgraph-generator`
- `NavGraphWithMPSGeneratorInterface::navgraph-generator-mps`

#### Reader(s) & Writer

<table>
<thead>
<tr>
<th>Reader(s)</th>
<th>Writer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>GazsimGripperThread()</td>
</tr>
<tr>
<td>0</td>
<td>RobotinoSimThread</td>
</tr>
<tr>
<td>3</td>
<td>LaserFilterThread(cil)</td>
</tr>
<tr>
<td>2</td>
<td>LaserFilterThread(map)</td>
</tr>
<tr>
<td>3</td>
<td>LaserSimThread</td>
</tr>
<tr>
<td>5</td>
<td>LaserFilterThread(simulation)</td>
</tr>
<tr>
<td>3</td>
<td>MapLaserGenThread</td>
</tr>
<tr>
<td>0</td>
<td>LaserClusterThread(mps)</td>
</tr>
<tr>
<td>0</td>
<td>LaserClusterThread(robots)</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>1</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>1</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>1</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>1</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>2</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>1</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>1</td>
<td>LaserLinesThread</td>
</tr>
<tr>
<td>4</td>
<td>RobotinoSimThread</td>
</tr>
<tr>
<td>1</td>
<td>NavGraphGeneratorThread</td>
</tr>
<tr>
<td>1</td>
<td>NavGraphGeneratorThread</td>
</tr>
</tbody>
</table>

---

**WebView**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Loaded</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent-monitor</td>
<td>Monitoring the CLIPS agents in LLIF via webview</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>amcl</td>
<td>Adaptive Monte Carlo Localization</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>ax12_gripper</td>
<td>AX12 Gripper Plugin</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>bblogger</td>
<td>Write BlackBoard interface data to files</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>bbloggerplay</td>
<td>Replay BlackBoard log files</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>bbsync</td>
<td>Synchronize with remote Fawkes BlackBoards</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>clips</td>
<td>Provides CLIPS environments</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>clips-agent</td>
<td>CLIPS-based agent plugin</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>clips-motor-switch</td>
<td>Motor switching from CLIPS</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>clips-navgraph</td>
<td>CLIPS feature to access the NavGraph</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>clips-protobuf</td>
<td>Protobuf communication for CLIPS</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>clips-ros</td>
<td>ROS integration for CLIPS</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>clips-tf</td>
<td>CLIPS feature to access transforms</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>clips-webview</td>
<td>CLIPS introspection via webview</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>coll</td>
<td>Local locomotion path planning with collision avoidance</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>conveyor_vision</td>
<td>Conveyor Vision plugin</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>dynamixel</td>
<td>Robotis Dynamixel servo driver plugin</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>eclipse-clp</td>
<td>Runs the ECLiPse CLP interpreter</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>festival</td>
<td>Festival speech synthesis integration</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>flute</td>
<td>Flute speech synthesis integration</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>frbase</td>
<td>FineVision Base provides access to camera and handles timing</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>frfountain</td>
<td>Provides access to images, colormaps etc. via network</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>firetdevice</td>
<td>Reads images from cameras and stores them in shared memory</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>gazebo</td>
<td>Provides access to Gazebo</td>
<td>Yes</td>
<td>unload</td>
</tr>
<tr>
<td>gazsim-comm</td>
<td>Simulates and manages communication for testing with Gazebo</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>gazsim-conveyor</td>
<td>Simulation of a Conveyor</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>gazsim-gripper</td>
<td>Simulation of a Gripper</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>gazsim-laser</td>
<td>Simulation of the Hokuyo in Gazebo</td>
<td>No</td>
<td>load</td>
</tr>
<tr>
<td>gazsim-light-beam</td>
<td>Simulation of the LightBeam Plugin</td>
<td>No</td>
<td>load</td>
</tr>
</tbody>
</table>
Further Features

- Text and data logging facilities
- Configuration sub-system
- Network messaging infrastructure and discovery
- ROS integration
- Plugins for performance analysis (RRD, mongodb-log, ...)

⇒ Batteries included
1 RoboCup Logistics League
2 Fawkes Robot Software Framework
3 Software Components
4 Conclusion
Perception

Machine Signal
- Recognize light signal on MPS
- Combine laser and image data to detect signal
- Robust to disturbances

Conveyor Detection
- Detect conveyor on MPS
- Based on point clouds from 3D vision
- Used for precise positioning
Self-localization
- Adaptive Monte Carlo Localization
- Particle filter on poses
- Works on 2D laser data

Global Path Planning
- Topological graph search
- Provides points of interest
- Statically collision-free paths

Local Path Planning
- Grid map approach on laser data
- Avoid dynamic and static obstacles
- Take global path as guide line
**Gazebo-based Simulation**
- Full 3D simulation with physics
- Based on well-known Gazebo simulator

**RCLL in Simulation**
- Use referee box for environment agency
- Very close to the real game
Basic actions for reasoning layer
- Emphasize description over programming
- Allow programming where necessary
- Modeled using Hybrid State Machines
- Abstract low-level system
- Implemented for Fawkes and ROS
- Written in the Lua scripting language
Incremental Task-Level Reasoning

- Only commit to single step at a time
- Strategic behavior with coarse tasks
- Reason about current knowledge

- CLIPS rule-based system
- Efficient reasoning with many updates
- Distributed, local-scope, incremental

```
(defrule s1-t23-s0
  (state IDLE) (holding S1)
  (machine (mtype \?mt&M2\_3) (name \?n)
    (loaded-with \$?lw&:(contains$ S0 \?lw)) )
 =>
  (assert (task-candidate goto \?n))
)
```
1. RoboCup Logistics League
2. Fawkes Robot Software Framework
3. Software Components
4. Conclusion
Fawkes as a versatile software framework is the foundation for the publicly released Carologistics software stack.

- Hybrid blackboard middleware
- Massively multi-threaded software components
- Versatile Behavior Engine and reasoning agent
- Focus on integration with reasoning components
- RCLL software stack released as open source software

https://www.fawkesrobotics.org/
https://www.carologistics.org/