GenoM3
Building middleware-independent robotic components

A. Mallet, C. Pasteur, M. Herrb, S. Lemaignan, F. Ingrand.
Paper presentation by Pablo Rauzy.

École normale supérieure

Robotics presentation, February 15, 2011

Introduction

The problem

- Need for reusable code
- ... at every level

Introduction

The solution

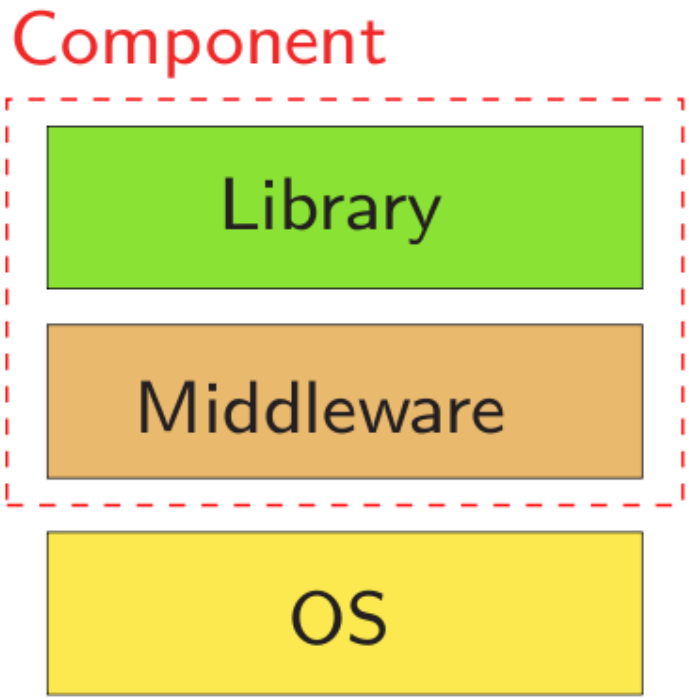
- Software engineering approach
- ... at every level

Introduction

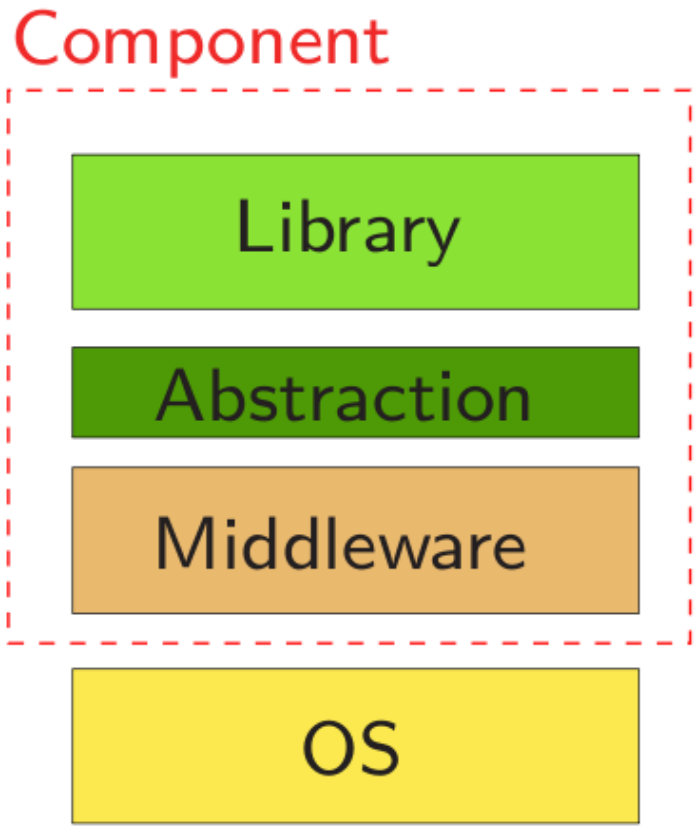
How?

- Classical software engineering and architecture for high level code
- A component based architecture
- An efficient way to abstract middleware (this is what was missing)

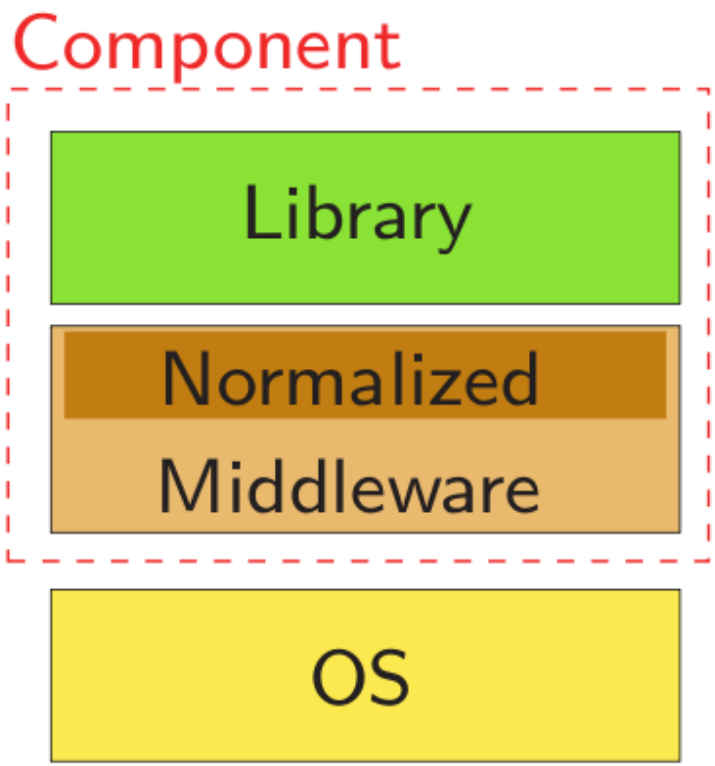
Abstracting middleware?



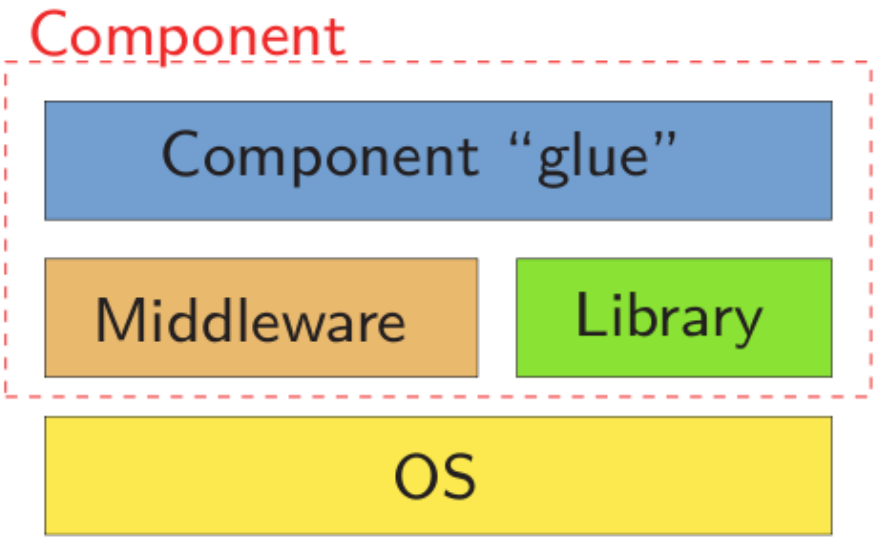
Abstracting middleware?



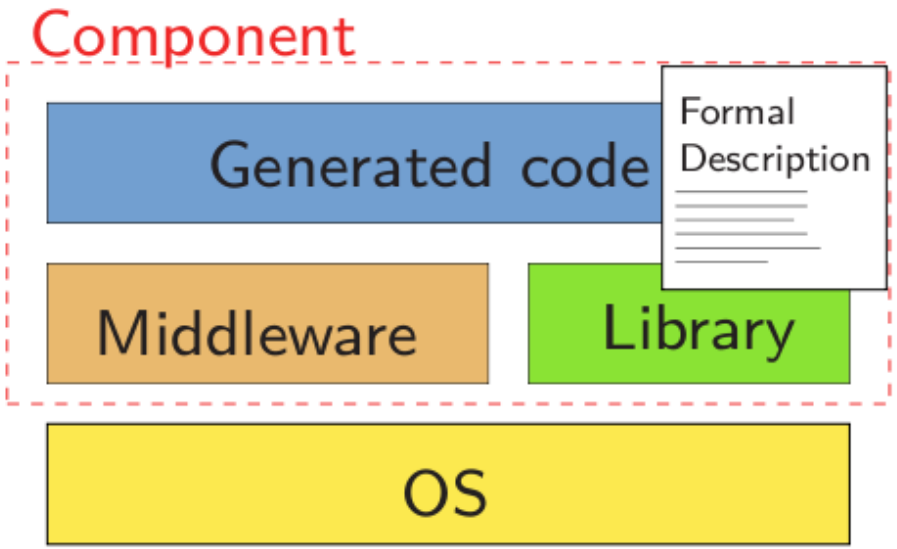
Abstracting middleware?



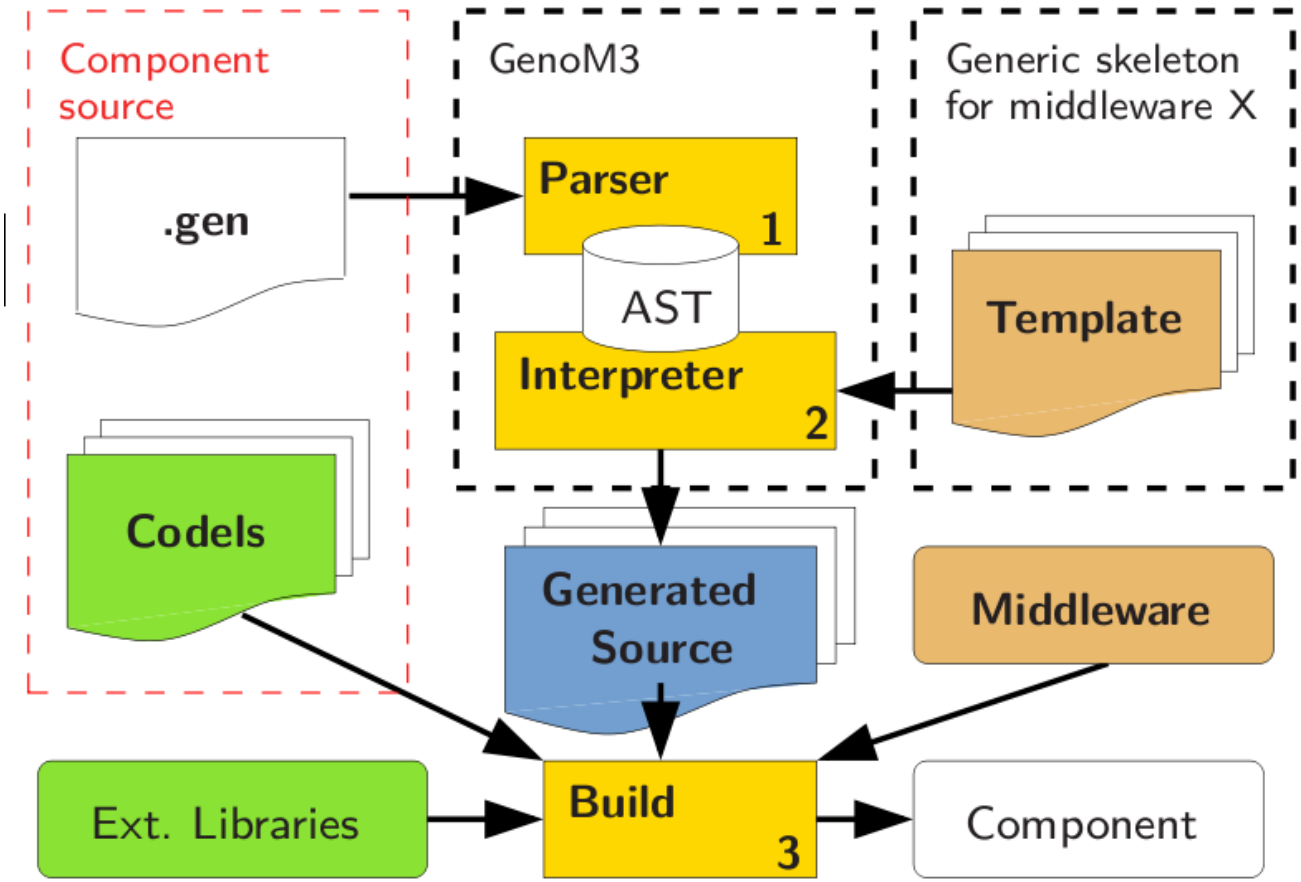
Abstracting middleware!



Abstracting middleware!



An overview of the G^{en}oM3 workflow



GenoM3 usage

Glue

- Link the library and the middleware together
- Middleware independance is deferred to the sole glue software
- Separation of concerns between the algorithmic core and the middleware
- Generated automatically!

GenoM3 usage

Component templates

- Contains all the code that is not part of the algorithmic core of any specific component
- Only one template is *required* for a given middleware
- Yet, different templates allow to switch easily between alternative architecture
- Developped in any language, using the whole language

GenoM3 usage

Component description file

- Contains all the necessary information to describe the component
- GenoM3 parser builds an AST and converts it into a suitable representation for the scripting language of the template interpreter

Data in component description files

- Data types used in the interface
- Meta-data (language, internal data structure)
- Data ports and events
- Tasks
- Services

GenoM3 usage

Codels

- Set of individual functions, performing only elementary actions
- No different from any regular library

Comparing frameworks:

YARP, MS Robotics Dev Studio, URBI, OpenRTM-aist, ROS

Comparing frameworks
YARP

- Collection of programs which communicates (strong focus on communication)
- C++
- Designed for research
- ROS compatible

Comparing frameworks

Microsoft Robotics Dev Studio

- Non-free (as in freedom)!
- Windows only
- C# or VPL (visual programming) / .NET
- Fullstack IDE + advanced simulator
- Natively work with a lot of commercial robots (incl. NXT)

Comparing frameworks
URBI

- Component based
- C++ UObject orchestrated by urbiscript DSL
  - Parallelism for free
  - Easy event-based programming
  - Advanced flow control using tags
- ROS compatible

Comparing frameworks

OpenRTM-aist

- Component oriented
- C++ / Java / Python
- ROS compatible

Comparing frameworks
ROS

- Full OS
- Control low level too
- Notion of nodes (similar to “components”)
- C++, Python and ROS
- Seems to be the reference

Comparing frameworks

Conclusion

ROS wins?

The end

Questions?