

Objective: Creation of the first draft of the document on standardization concepts in robotics software.

Place: University of Lund, Lund, Sweden.

Schedules: **main meeting: Monday 21.05.07 10:00-20:00.**

supplementary meeting: Tuesday 22.05.07 10:00-13:00.

Participants: **Klas Nilsson, Ander Nilsson - Lund University, project partner.**

Herman Bruyninckx - Catholic University of Leuven, external expert.

Azamat Shakhimardanov - FH Bonn-Rhein-Sieg, project partner.

Main meeting, Morning session, start at 10:30:

Keywords: Structuring, ontology, common representation

Some suggestions given by

Herman -

What shall we discuss? C level API, stateful API, define binary, timers, low resolution position? Shall we discuss ontology, shall we discuss such things as matrices. Do we use projects, standards which already exist? How far do we go? What do we do with the results? Who is going to do standards? OMGRobotics 25-29 Brussels? BROS was established between Player/Stage, MIRO and OROCOS - for position and orientation concepts mostly. But Herman thinks it is not worth to start over with it again.

Herman suggests to start from the low level? Device driver interface: Raw device drivers, RTDM, Logical device drivers. Do we have read/write, treat them like files in UNIX. These are mostly questions which can be discussed.

Klas

Emphasizes the importance of the concept representations and discussed different ontologies at particular representation level. He suggests to have the following three levels of representations:

1. General concepts/meaning (Orientation, position, ... etc) repr. - the elements on this level conduct some idea/meaning/description of a robotic element under discussion.

2. Mathematical repr. - elements from the layer above can be represented/described different mathematical methods/concepts. For instance orientations can be represented in terms of Matrices or quaternions.

Further, it is suggested to differentiate between mathematical **(a) DATA** and **(b) ALGORITHM**. That is, in our example, entries of the matrix are data and the way to process them is relevant to algorithms

3. Computer-repr. - on this level, elements from the level above are represented in terms of concepts/syntax/grammar that can be comprehended by a computer system. For instance, using unsigned int for position of a mobile platform or double precision floating point numbers for the manipulator positions, C data structure for the quaternions. Since most of current ROS, more or less, revolve about this level of representation, there have given many example "standards" from other domains such as computer graphics and Internet. There were discussions what would be the optimal step to take, to adopt one of these standards, to extend it or to create sth similar from the scratch.

For instance what format one is to adopt for the log files, component/simulator/robot hardware description files. Examples can be **C3D** file format for the pos, orient, quaternions. **RRS** - realistic robotics simulations. **X3D** standard (**Xj3D**), **OWL** - ontologies(web ontology language), **Modelica**, **3D-XML**, **SysML**, **SCXML**.

Later Herman suggests the fourth representation layer. **Hardware representation.**

Klas talks about different constraints for each of the above mentioned constrained.

He identifies three types of constraints:

1. Constraints on the meaning
2. Finite domain constraints
3. Physical constraints/Mathematical constraints

(do not remember details, Klas please fill in)

Main meeting, Afternoon session, start at 13:30:

Keywords: Code generation, mapping from math to computer representation, requirement analysis.

There was a discussion on how to bridge the gap between levels/how to map from one to another without problems. This can be done for instance using a description language and specific compiler. The compiler compiles the description of component/module/algorithm/etc (general concept/ math repr.) and generates a code(computer repr.). Examples are IDL, GenoM, LabComm.

KLAS PLEASE FEEL FREE TO FILL IN HERE

Requirement analysis for robotic software systems

1. Decoupling "real-time" and nonreal-time

2.

(a)Functionality - more on implementation details??? Interfaces:

Events,

Method calls,

Commands

Data flow

COnfiguration/Property interface

CASE STUDY 1:

Player/Stage - mobile platform, model the robot and I move in the world

CASE STUDY 2:

Manipulation/Realt Time -

CASE STUDY 3:

Grasping/manipulation -

To summarize things up -

The four layers are

1. Meaning/Robotics object ontology - orientation, position
2. Math repr./ level specific ontology (matrix, quaternion for the orientation) -
 $f(x) = ax + b$. Mathematical representation of (a)DATA and (b)ALGORITHM
3. Comp repr./level specific ontology (data structure, range and resolution of the data structure, boolean etc)- description language, parser and generated code.
4. Hardware repr. - with respective hardware architecture, Machine language.

Suggestions are - Starting point can be common logging system/common log file format. The file content is - force, angles, positions, distances, 3D, motion etc. This information is structured according to application domain (wiki section ontologies). Here one also needs to take into account the scope each of the representation layers addresses. This we should do in cooperation with experts. (OVER THE MAILING LIST, SOME CONSENSUS SHOULD BE REACHED BEFORE MOVING TO BIGGER GOALS)

What does this give - common file format can be used among different software projects. Further, if many software project leaders agree on common representation, the ultimate goal is to put forth interoperability among projects. This should go beyond source code/compilation units. For instance it can be that there is a common module/component/driver/etc representation language which compiles to platform specific code. This can also be further used to provide a

comparison among systems using the "same component/module/driver/etc/" (WELL, WE WILL SEE).

Relevant information can be found in ROSE(robotics ontology for semantic web)on EURON

Homework mostly for WP2 and WP3

1. Overview of the legacy systems (Player, ORCA2, etc)- is actually being done currently. This is required to identify common features, API, data structures, log files. This info can then be used to structure/extend/enrich proposed "common representation/log file format, for instance".
3. Think of/propose to experts (after the first consensus is reached) common description language for components/modules/algorithm/etc, which then can be parsed/compiled to platform specific code.

PLEASE FEEL FREE TO COMMENT ON THE POINTS PRESENTED!