

# Meeting architecture & middleware

— Leuven 19–20 nov 2007

Participants:

**Herman Bruyninckx** [Host] Katholieke Universiteit Leuven, Belgium

**Klas Nilsson** [presenter, RoSta WP2, organizer] Dept of Computer Science, Lund University, Sweden, [klas@cs.lth.se](mailto:klas@cs.lth.se)

**Paul Plöger** [day 1], Univ of Applied Sciences Bonn-Rhein-Sieg, Dept. Autonomous Systems, Sankt Augustin, Germany, [paul.ploeger@fh-brs.de](mailto:paul.ploeger@fh-brs.de)

**Azamat Shakhimardanov**, Univ of Applied Sciences Bonn-Rhein-Sieg, Dept. Autonomous Systems, Sankt Augustin, Germany, [azamat.shakhimardanov@fh-brs.de](mailto:azamat.shakhimardanov@fh-brs.de)

**Torsten Kröger**, Institute for Robotics and Process Control, Technical University of Braunschweig, Germany, [t.kroeger@tu-bs.de](mailto:t.kroeger@tu-bs.de)

**Fabio P. Bonsignorio**, CEO Heron Robots s.r.l., Genova, Italy, [fabio.bonsignorio@gmail.com](mailto:fabio.bonsignorio@gmail.com)

**Anders Blomdell**, Dept of Automatic Control, Lund University, Sweden, [anders.blomdell@control.lth.se](mailto:anders.blomdell@control.lth.se)

**Ricardo Velez** [day 2], Visual Components Oy, Helsinki, Finland, [ricardo.velez@visualcomponents.com](mailto:ricardo.velez@visualcomponents.com)

## DAY 1

- 10.00-11.00 Arrival with coffee and informal discussions
- 11.00-11.45 Summary of IROS WS
- 11.45-12.15 Definition of terms and relation to architecture.
- 12.15-12.30 Agenda and topics for the meeting
- 12.30-13.15 Lunch
- 13.15-17.30 Existing solutions and experiences, covering architectures and middleware in more detail
  - IROS WS summary
  - DESIRE
  - [hardrealtimedorba.org](http://hardrealtimedorba.org)
  - OROCOS
  - MIRPA
  - LAAS architecture
  - JAUS
  - Claraty
  - MSRS
- 19.00-- Get together in hotel bar

## DAY 2

- 08.30-09.00 Next steps and agenda of the day.
- 09.00-14.00 Review of open issues
  - component composition
  - State machines (single/multiple activity), petri-net, SFCs. SYSML/Marte/SCML/...
  - Distributed objects/computing
  - One-way communication (what is built on top of what)
  - Real-time communication
  - OSGi, Terracotta, CORBA, ... (non-real-time)
  - PnP/discovery/deployment for real-time systems
  - Resource management
  - Basic technical solutions or managing complex systems
- 14.00-15.00 Closing and final discussions.

- Torsten: Referring to the importance of Ontologies and that there was a lot of discussion on that. The other issue was that why do we need benchmarking/evaluation and standards. This is related to ontologies again and it is mostly about definition and the context it is used. Mentions about the different requirements for different robotic domains, and the software in those domains.
- So basically to summarize, it was on
  1. Ontology
  2. Why do we need evaluation/standardization (if we do, how)
  3. Requirement analysis, and decomposition of different application domains.
- Klas shows and discusses some of the slides from the IROS, see separate info including slides on wiki.
- There is a discussion on what middleware is and is not.
  - Herman says that the solved problems will form the basis for useful APIs
  - Klas says if the problem is well understood then they can be encapsulated and API developed but that they always evolve. But mostly the implementations change and not the APIs themselves.
- Paul mentions MDA/SysML , simulators. Then everybody is talking about the tool support. What tools to choose and to monitor. Open free tools are needed, not to become dependent on have models that require licenses to use.
- There is a further discussion based on the IROS presentations.
- There is a discussion on defining middleware and other terms, which will be used for the discussions.

Touched the following issues/examples, without detail (this is not WP1):

- Types/purpose of ontology
- Glossary – taxonomy – ontology
- Modeling vs reasoning
- Static concepts vs dynamic data model
- Manufacturing: Product, environment, equipment
- Service robots; PMAC [Korea]:  
perception – model - activity – context
- Principles for grounding
- The following were defined/discussed; see next slides:
  - Middleware
  - Components
  - Architecture
  - Real-time
  - Framework, library and patterns

- Wikipedia def, see <http://en.wikipedia.org/wiki/Middleware>
- Tanenbaum: ref by Torsten
- IROS WS:
  - The plumbing, not noticeable; just well working as needed.
  - It is about the communication, including policies and mechanisms.
- Torsten et al (incl lunch discussion):
  - Process transparency
  - MW is about decoupling processes (not arch dep.)
  - Policies: 1-1, 1-n, sync/async etc etc for communication.
  - MW APIs should support available policies (for config, run-time..)
  - Keep policies separate from mechanism [Herman].
  -

**Middleware is computer software that connects software components or applications. The software consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network.** This technology evolved to provide for interoperability in support of the move to client/server architecture. It is used most often to support complex, distributed applications. It includes web servers, application servers, content management systems, and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture.

The term middleware is sometimes considered a buzzword.

- Pieces of reusable software
- Normally referring to binaries that can be deployed at run time
- The safety-critical community talks about compile/design-time components.
- Hot-plugging not always needed or supported, but desirable for instance in DESIRE
- Even if hot-plugging is technically possible on the run-time level, it needs support from the application/architecture.
- No clear/obvious best component model.
- Ports or parts of the interface with different properties, such as the data ports in OROCOS, is suitable.
- The need for back-propagation of saturations and exceptions upstreams wrt the data flow was emphasized by Klas, who will provide an implementation of SDSblocks (sampled dynamic systems control components) in Java, inspired by the DSblock standard proposal from DLR.

## Top-level description:

- Overall structure
- Purpose of design
- Defined invariants
- Points of variation
- Explicit trade-offs (for the tasks and environments that the system is being built for).

## Should support:

- Efficient engineering (MDA&tooling),
- Reuse/sharing of modules
- Exchange of components (design time, compile time, .. run-time, ...)
- Understanding (to human, and possibly to the machine via ADLs/ontology/..)
- Coping with complexity (by means of abstractions, boundaries, policies, interconnections, hierarchies,.. )

Other types: Software, control, hardware, product line, etc., and buildings.

Relevant but provides only partial solutions.

Architecture is the complement to components.

Robot control architectures are not black boxes...

- A minority of the robot researchers and engineers care about real-time (to start with but perhaps later when they run into problems).
- APIs and tools may not be significantly more complicated by considering real time (then they will not be widely used).
- Klas: Hard real time does (sometimes) matter; ref DLR.
- Real-time software will only be real time if run on a real-time platform (RTOS)
- Real-time software means software that accomplishes real time if run on an RTOS and called from real-time threads. That is, real-time software permits real-time execution, but does not necessarily require complex APIs or frameworks (could even require simplification). To be kept in mind when looking at future middleware solutions.
- Real-time == *permitting* real-time, predictability== *permitting* predictability, etc.  
For a component, what parts of the interface that is 'permitting' need to be well defined.

- Paul gives presentation of DESIRE project (separate slides).
- CORBA-based components due to heterogeneous requirements and platforms.
- There is a discussion on API of the arm component and Kuka API, about the position of elbow which follows from the 7DOF
- Check the DLR IROS presentation which can be useful for the requirement analysis in particular to realtime.
- [DLR proposed the properties of their architecture(hardware, software) as one of the requirements for the realtime]
- Automated scenario generations, how to make many different scenarios...
- Paul also talks about Robocup and mentions about the standardization of hardware facilitates and the software development progress. All teams always develops everything on their own (but some hardware parts but very little software is reused from year to year)
- Dynamic deployment and migration support is needed.

- Herman is saying that when someone is talking about components then they are often talking about policies.
- Different policies on top of asynchronous message parsing. OROCOS wants to provide mechanisms.
- Orocos doesn't have multiprocessing/distributed environment support when it is using Xenomai.
- It uses (semantics of the) UML state charts.....
- Herman says matrix libraries are not real-time, pre allocation of the memory is the way to go.
- Klas says in hard realtime it should be possible/requirement to allocate object at runtime, on heap.
- Not really based on standards yet. SysML etc is interesting.
- There is a .deb packages for OROCOS
- Klas says challenge is to find low entry solutions(easy solutions) that scale well.

- MIRPA presented, see IROS WS slides and paper.
- Why yet another architecture, why MIRPA? The solution for distributed real-time supporting synchronous object interaction is unique, with commonly requested features (distributed method calls).
- QNet, a lot of discussions on messaging/RPC threads, scheduling, priority inheritance and so on.
- (Opal-RT was referred to, <http://www.opal-rt.com/>)
- Torsten asks whether it is a good idea to release it as OpenSource. But QNX and Qnet is not open.
- Lengthy discussion on efficiency and communication overhead. Most meeting participants were in favor of having asynchronous one-way message passing as the basis for communication between (potentially) distributed components.

- Ricardo gives presentation of Visual components 3DCreate, with references to ABB Robot Studio, etc..
- RRS- realistic robot simulation, standardized interface for many robot controllers,
  - RRS1 - focus was on motion commands for industrial robots, in some cases robot dependent extensions but the standardized core works well. Putting component in binary format.
  - RRS2 - focus was to include IO-operations and complete robot programs. No general acceptance.
- Program has PLC add-on. So that you can do validation of some machines such as conveyors etc.
- Program connects to external PLC. Inline production (ILP) with Siemens
- Products - 3DVideo, 3DRealize, 3DCreate.
- Knowledge-intensive concurrent engineering, using X3D for geometry part of ontology. Also uses Collada.
- Advanced Design, Planning and Optimization of Assembly Systems
- Product can interface with MSRS, as will be shown at Automatica.
- Other tools such as blender was discussed.....

# Physics simulation

- There is a discussion about simulation/physics and if what cases it is possible to do simulation of force control/interaction. The MSRS and game oriented engines are based on modeling impact and then generating forces and motions that look realistic, but they actually do not reflect the physics well enough so simulating force sensing and control would give no realistic behavior. No shortcut to proper FEM analysis and proper simulation eg using tools based on Modelica ([www.modelica.org](http://www.modelica.org)).
- Still with the very best tools and appropriate modeling, to feasibility to predict the behavior of sensor-based robot motions such as high-performance force/torque control was questioned; Herman considered it as not possible to simulate in practice with reasonable accuracy and engineering, whereas Klas meant that it would to some extent be possible with proper architectures and algorithms for system identification and control.

- Composition of two components should result in a new component, not in two glued components (not appearing a single entity in the system). Related to SDSblocks.
  - Do you really need to expose component state to outside world?  
No, but for mode changes and hot-swapping you need proper management of internal states.
- (Component-Oriented Engineering) or description of state machine?
- Herman mentions MARTE/SysML. Basically we are searching for some architecture description language with many features, i.e. real-time features, IDE with code generation. Do the already existing ADLs have such support?
- Herman presents some slides on Objects and components, Section “3.1 Component programming semantics” from <http://people.mech.kuleuven.be/~bruyninc/robotstandards/standards-howto-20071101.pdf>
- Azamat: Check Nancy Lynch's book on "Distributed Algorithms"
- Klas: There is some talk about Simulink and code chain tools/code generation. Algorithms are mentioned too.
- Opposed to blocks in Simulink, express algorithms in some description language so that they are reusable.
- OpenMath is a candidate on the algorithmic/mathematical level.
- Lots of discussions.....

- Referred to as a promising standard for modeling components and their interfaces.
- Overlap and differences relative to UML was mentioned.
- How are resources and resource utilization modelled, and how to make use of that information during deployment?
- Different opinions if SysML will be useful; will it be simple enough so developers get the immediate benefits?
- Tools (typically based on eclipse) need to be open-source, not to be dependent on something that suddenly might not be useful or affordable.

## Classifications

Makarenko:

- Drivers and Algorithms (DA)
- Communication Middleware (CM)
- Robotic Software Framework (RSF)

Meeting: Separation between Drivers and Algorithms made sense for the purpose of his paper but not for actual MW, so we have four categories of software.

Nernas: Three categories for measures and procedures:

1. Programmatic – software architecture and management related measures and procedures
2. System-engineering – related to interactions between sub-systems
3. Component-specific – specific to particular components or sub-systems

Each of these contain numerous items

Bill Smart @IROS

(reformulated): I ain't going to use your architecture, but lets agree on some middleware and algorithms.

For principles of levels to find agreements on, the following slides are from the BROS presentation at IROS, providing the generic representation levels (not to be confused with multi-layered architectures).

Four levels of representation

1. **Ontology**
2. **Mathematical**
3. **Computer**
4. **Native HW**

Multiple mappings (typically automated)  
from high to low level permitted.

Do not confuse with layered architectures!

- Define meaning in a computer&human understandable way
- Formal def of meta-level.
- Provides definitions and restrictions for lower levels.
- Our test cases uses:
  - The OWL XML-based language: [www.w3.org/TR/owl-features](http://www.w3.org/TR/owl-features)
  - Ontology editor, and reasoner framework: [protege.stanford.edu](http://protege.stanford.edu)
  - Toolkit for implementation of mappings to related standards [jastadd.cs.lth.se](http://jastadd.cs.lth.se)

## From Wikipedia:

An ontology is a data model that represents a **domain** and is used to **reason** about the objects in that domain and the relations between them.

Ontologies are used in artificial intelligence, the semantic web, software engineering and information architecture as a form of knowledge representation about the world or some part of it.

Ontologies generally describe:

- Individuals: the basic or "ground level" objects
- Classes: sets, collections, or types of objects
- Attributes: properties, features, characteristics, or parameters that objects can have and share
- Relations: ways that objects can be related to one another

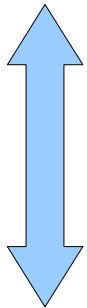
- Today often implicitly related to software frameworks and libraries.
- Examples: Coordinate systems, units, transformations, measurement filters, estimators, feedback control laws, optimization algorithms and criteria.
- Proposed:
  - Anchor data definitions on the ontology level
  - Express definitions/relations/equations/functions formally
  - Suppress computer/software decisions (but cope with implications)
  - Generate (parts of) the actual resulting software:
    - Make use of definitions generated from the ontology level
    - Generate platform independent software, or explicit dependencies
- Our test case uses: Maple, MathML and LabComm, with output to C (e.g., for use in Simulink/RTW), Java, Python.
- Open Issue: Algorithm descriptions

- Encoding of computations in computer languages
- Today tightly connected to the actual platform
- Should be expressed in a platform independent way
  - Stating if floating point or fixed point is to be used
  - Required resolutions and features from the environment
  - Data organization (matrix storage, etc.)
  - Function properties (args by value or by ref, allocations, etc.)
  - Resource management (memory, IO, etc.)
- Make use of code generated from level 1 and 2, add (possibly multiple) computing specifications, compile to specific platforms.
- Our setup: Using the output from the math level (C/Java), hand written additions in real-time Java, Java for Internet-ready simulations, specific interfaces to tools (Simulink). Java2C translator to reach any target hardware.

- Variation in hardware and binary (message/file/code) formats
- Part of a standard if influence on practical value of it.
- Today, often unnecessary hardcoding, implicit dependencies (fragile implementations), and closed systems.
- System interconnections often need to be binary, but useful for different type of hardware/formats/messages/compilers/libs.
- Relevant technologies for binary data exchange:
  - <http://www.asn1.org>
  - <http://torvalds.cs.lth.se/moin/LabComm>
  - <http://hdfgroup.com/products/hdf5>
  - <http://www.unidata.ucar.edu/software/netcdf/index.html>
- Open issues:
  - Best combination of the above relevant technologies
  - Native (migrateable) representation of a algorithms

- Separate (but related) standards are needed for
  - Data
  - Transformations that operate on that data.
- Use/combine existing standards if possible.

Abstractions  
and notions



Generated  
executable

Level	Data	Operation
Ontology	OWL	JastAdd aspects
Mathematical	MathML, SCML,..	Maple, modelica..
Computer	LabComm, ASN1	sub/super-set of RT Java ???
Native	HDF5, netCDF, LabComm, ASN1	Native binaries or CLR & JIT (proxy)

# Power Law in Data Standards

- The more systems that must adopt a common standard, the simpler it must be

