

## IM-CLeVeR Report on:

- 1) IM-CLeVeR decisions on technical integration
- 2) Comparison of Brahms and Yarp

**Gianluca Baldassarre, 02-10-2009**

Regarding the technical integration of the Consortium, as you know at the kick-off meeting of IM-CLeVeR the Scientific and Management Board (formed by the Partners' Team Leaders) decided to \*investigate\* the possibility of "committing the Partners to use of Brahms on top of Yarp".

Yarp is the system, produced by IIT, which can interface your model-programs with either the iCub robot or the iCub simulators. Brahms is a software developed by USFD in order to allow easy communication between different model-programs, e.g. a rate-neuron model of amygdala written in Matlab with a spiking-neuron model of basal ganglia written in C++.

At the meeting, USFD and AU mentioned that there is the technical viability of using Brahms on top of Yarp. However, CNR proposed to test the ease of this possibility but the test failed: Fabian Chersi (CNR; researcher-level programming skills, but not computer scientist), managed to install Yarp and the IIT iCub simulator (see 'How to install the iCub simulator on Linux', by Fabian, in IM-CLeVeR website <http://im-clever.noze.it/resources/middleware>) on various CNR machines, but so far CNR failed to install Brahms in Windows and Linux machines, even with the distance-guidance of USFD and AU.

In summary, it seems for now the situation is the following one:

- (a) Brahms+Yarp cannot be used under Windows;
- (b) The interface between Brahms and Yarp under Linux is possible (AU and USFD accomplished it)
- (c) However, the interface between Brahms and Yarp under Linux is not easy (CNR failed). Indeed, interfacing the two now still depends on things like your Linux version, Matlab version, other libraries that you have to have on your machine, etc., and this complicates things rather a lot.

To (a) solve this installation problems together and (b) show to the consortium the facilities of Brahms, USFD (in particular by Kevin Gurney, Ansgar Koene, Ben Mitchel, et al.) proposed to organise a 3-full-day workshop at Sheffield on Brahms. Kevin et al. are at the moment working to define the dates of the event and collecting people's participation (Kevin: please summarise for everybody where we are in this process).

**At this point, in terms of decision making of the consortium, I (Gianluca) propose that \*after the Sheffield workshop\* (all the modelist partners participate to the workshop with one/two of their members to 'directly put their hands' on Yarp and Brahms) the Scientific and Management Committee votes between one of these two options (but of course you might propose to vote for different ones):**

- 1) **The consortium commits to use Brahms+Yarp**
- 2) **The consortium commits to use only Yarp, but the consortium let's partners to freely decide if using Bahms on top of Yarp when this does not impair collaborations between partners**

**Independently of this decision, at the kick off meeting we also talked about committing the consortium to use a possible ‘standard of good programming’ to allow different models of different partners to easily communicate and work together.**

**I ask the Team Leaders to vote \*now\* (in favour or against) that:**

**- The consortium definitely commits to use such standard, independently of the decision on the use of Brahms**

**- CNR will lead the definition of the standard with the aid of all other partners.**

**Team Leaders: consider that if I do not receive any feedback from you on this issue within one week time, I assume you agree (however, note that I prefer your explicit feedback instead of silence!).**

Incidentally, if you remember the guidelines of such standard were already partially defined at the kick-off meeting. I give you here some of the elements of the standard we discussed to give you an idea of the standard itself, but details will be discussed and defined after the vote:

- Organisation of the model-programs within an ‘initialise’ and ‘step’ function
- Use of comments in doxygen
- Minimal comments indicating: (a) the function implemented by the model (e.g., processing of images to extract position of target; action selection); (b) its input-output interface needs and properties (e.g. needs to know presence of objects; returns macro-action to be executed); (c) what time-granularity it uses (e.g.: milliseconds of spiking neurons vs. tens of seconds of rate-code neurons); (c) at what level of abstraction it operates (e.g., pixels vs. identity/presence of objects).
- Use of Matlab or C++ clean code
- Other things to be defined...

In parallel with this issue of “technical-viability” of Brahms+Yarp, CNR envisaged the importance to think about a second issue related to Brahms and Yarp: the possibility that Brahms actually supports functionalities already supported by Yarp. Indeed, aside the *usability* of Brahms, the opportunity to use it also depends on its *added value* it has with respect to Yarp alone. As it seems that nobody knows *both* Brahms and Yarp, Gianluca invited two experts on Brahms (Kevin Gurney and Ben Mitchel, of USFD) to furnish a list of features of Brahms potentially useful for IM-CLeVeR, and then invited an expert on Yarp (Girgio Metta, of IIT) to tell if Yarp can support such functionalities. The outcome of this investigation is reported at the end of the report.

Regarding the Sheffield Meeting, however, I want to say that from the comparison reported below and other considerations it seems to me that it is very important that at the Sheffield workshop people understand very well not only Brahms, but also Yarp and possibly the the iCub simulator(s), as these will be very important for taking any future decision and, anyway, for technical integration of IM-CLeVeR.

**For these reasons I propose that USFD organises the Sheffield Workshop with this macro-agenda:**

- **First day: Yarp + iCub simulator(s)**
- **Second day, morning: Yarp + iCub simulator(s)**
- **Second day, afternoon: installing Brahms+Yarp**
- **Third day: Brahms (+ NodeWeaver and MODLIN if you think it is the case)**

...of course the agenda can be adjusted, but I think the constraint that Yarp and iCub simulators will take at least half of the meeting should be fulfilled.

**Kevin, is this OK?**

**Other Team Leaders: if I do not receive any feedback from you on this within one week I assume you agree.**

I have already contacted IIT, and they agreed to send a Yarp-expert to the Sheffield meeting to tutor the first day and possibly the morning of the second day.

Side issue on this participation by IIT: IIT asked IM-CLeVER to fund the trip and staying of its expert, so I ask Sheffield, if possible, to actually reimburse such expert and then, at the next fund-transfer, the sum so spent will be withdrawn from all other partners (in proportion to their requested budgets) and turned to Sheffield:

**Kevin: is this OK?**

**All other Team Leaders: if I do not receive any feedback from you on this within one week I assume you agree. Consider that divided by all the cost incurred by each partner will be very small.**

## **Comparison of Brahms and Yarp**

### **Ben Mitchel**

I agree with Gianluca's comments, that if the project is already committed to using YARP for some components, and that if YARP meets all the more general requirements, then adding BRAHMS to the mix would add complexity and might add nothing. In this case, I don't anticipate any significant problems interfacing BRAHMS/YARP to satisfy those (such as @shef) who rely on BRAHMS for reasons not specific to this project (as outlined by kev). The other day I ran by kev a couple of ways I know of in which this can be done fairly efficiently, though as I understand it a BRAHMS-YARP proof of principle has already been given, and anything efficient will do within the context of this project.

I will add a couple of specific notes on top of Kevin's to highlight a couple of things (I have zero knowledge of YARP, I'm afraid)...

### **Giorgio Metta**

I don't know much about Brahms, a proper comparison clearly requires some in-depth analysis and especially use of the two frameworks. Superficially, it looks like YARP shares a great deal of functionality with BRAHMS. I suspect the differences are minor. The interface between the two is clearly possible but the point is on the added complexity and learning curve of two systems/frameworks instead of one.

### **Kevin Gurney**

So I give you my version of the list below - Mitch, please amend as you see fit.

BRAHMS can

1) allow user-defined processes to communicate, which may have different time scales (as long as they have an 'init' and 'step' framework - most do I believe). These processes may be simulators of any kind and/or device controllers,

### **Ben Mitchel**

... not implying that the "step" has to be regular for a given process; all that is regular in BRAHMS is link propagation times (all inter-process data links are currently periodic). Links exchange at regular intervals ( $t=0$ ,  $t=T$ ,  $t=2T$ , ...), but a process can send/receive at any combination of sample rates on different ports, so the process timing model is completely general.

### **Giorgio Metta**

Yarp allows processes and threads to communicate in any time scale (this is not defined by Yarp but from the code itself, you instantiate your own timers and control them). Communication happens as soon as possible (or as soon as defined by the specific protocol or buffering policy being implemented). The communication object in Yarp is called a "Port" and can deal with any type of data (e.g. images, arrays) also defined by the user, in any protocol available (even defined by a more "skilled" user, typically UDP, TCP, MCAST, Shared memory), in any machine on a network (or multiple networks). There are protocols for delivering data across threads (e.g. maintaining the same mechanisms but sharing variables instead of using the network) or to send data in a human-readable format (e.g. text).

### **Kevin Gurney**

2) Brahms supports several language environments (C, C++, Matlab and Python currently) for writing processes.

**Ben Mitchel**

matlab works, but is clunky and inefficient. i blame mathworks, to be honest, but that doesn't change the fact of it :). python, in contrast, sings beautifully.

**Giorgio Metta**

Same for Yarp. Yarp can bind to C, C++, Matlab (via Java), Java, Python, etc. The implementation is native in C++, so the most efficiency is obtained in C++. Other languages are ok too, Matlab is not particularly efficient because of the Java wrappers. Users have been exploring other methods (though not fully supported yet) using ActiveX or the mex/dll interfaces. Moreover, Yarp works on Windows and Linux (fully supported) or on MacOS (many users but partial support so far), but also on Solaris if I remember correctly. It doesn't force the use of a particular IDE and/or build structure by relying on CMake (a popular meta-project specification). Documentation for the C++ classes is available online together with regression tests and compilation servers.

**Kevin Gurney**

3) Brahms is available NOW in a parallel form which allows rapid deployment on parallel hardware. As a demonstration of the power of this facility, we currently have a very large gaze-control model, working in real time, on a robot, only because it runs over 8 processor cores (in BRAHMS 'concerto'). Our final models will not be small...

**Ben Mitchel**

Yes, also with automatic inter-hardware data compression \*and\* network performance monitoring, yippee! :D

**Giorgio Metta**

Yarp runs on parallel machines, multi-core, etc. it also provides a standard for creating networkable device drivers (to manage hardware). We don't have data compression (though theoretically possible) because of efficiency reasons. Yarp has been written as a lightweight wrapper of interprocess communication and synchronization and hardware interface for controlling robots. Thus, it doesn't do anything on the data beside some trivial formatting or endian/type conversion where strictly required. We leave network performance to external tools. Yarp doesn't want to replace the OS or other co-existing tools or frameworks but possibly coexist.

**Ben Mitchel**

in addition to points 1-3, i would note the following:

A) BRAHMS is an execution framework; that is, it supplies all the base services that processes need to marshal and communicate as part of a system. in contrast to layers which only \*facilitate\* these operations (which may serve adequately for bespoke solutions that do not target modularity and reusability to the same extent) this takes the onus off the process developer and avoids the building of system-specificity into the processes. developing a BRAHMS process is comparable to developing a standalone algorithm implementation (assuming familiarity with the API).

**Giorgio Metta**

The same for Yarp. The idea is very similar, provide an infrastructure to build complex distributed applications. The algorithms can be kept entirely separate from the "plumbing". Various methods exist to make system specificity into "external" configuration files (code is general, configuration can be system-specific).

**Ben Mitchel**

B) baseline efficiency is really pretty good (an empty C++ process cycles in around 200ns on my desktop). this is important because it allows relatively fine-grained modularisation without loss of performance, and is presumably key to the success of the MODLIN effort. it is much less important if the modularisation is at a higher (e.g. "subsystem") level, since the compute time per step is correspondingly higher. BRAHMS performance also scales well with very large systems, with multiple compute threads, and is consistent across platforms. more details in [1]. other frameworks (e.g. YARP) might well share this property, or they might well not.

### **Giorgio Metta**

There's no overhead in Yarp connected to using it. You can cycle at whatever your machine allows it. The network efficiency on a Gbit/s network shows a delay of about 200 microseconds. We have a facility for running/killing/controlling distributed applications (many processes running on many machines) but Yarp never "owns" these processes. They're still running on their own and communicating via Yarp (which imposes a thin layer on top of existing OS communication facilities).

### **Ben Mitchel**

C) BRAHMS allows any types of data to be passed between processes (rather than, e.g., just double-precision arrays). for some framework architectures, this issue doesn't come up, but for those that provide data transport (such as BRAHMS and many others), better to be without a constraint, obviously.

### **Giorgio Metta**

Anything can be passed through Yarp ports. There's no limitation. For user defined types, marshaling/demmarshaling has to be provided by the user. For types for which "sizeof" would return the correct size (simple structs, no pointers, etc.) of the type, the user can rely on automatic marshaling/demmarshaling.

### **Ben Mitchel**

D) on the downside, BRAHMS is relatively immature, with all that brings; i presume YARP has seen a fair bit of action. mitigating that, i can usually offer fairly immediate support and bug fixes (though i'm not on the payroll, as you know).

### **Giorgio Metta**

Yarp started in 2001. The current release 2.xx is fairly stable with a reasonable support and community, one person on payroll on maintaining it.

I've described only functionality that is currently implemented. Having said this, note that Yarp is basically a C++ library providing classes for IPC and device drivers plus a standardized way of building modules and configuring them (none of these is mandatory anyway). The users have complete freedom of using only a subset of this functionality.