First National Workshop on Control Architectures of Robots - April 6,7 2006 - Montpellier

# Pleading for open modular architectures

Aurélien Godin, Olivier Evain
French Department of Defence
Angers Technical Centre / Vetronics & Robotics Group
{aurelien.godin,olivier.evain}@dga.defense.gouv.fr

***Abstract*— Since first researches in the field of robotics architectures, many advances have been achieved. Nowadays softwares offer modular functionalities, which interests will be reminded in section I, and actually satisfy most of the needs, as shown in section II. However, none of the mentioned approaches managed to spread widely or succeeded in gathering works developed by the very numerous robotics actors, not even those that can already be considered mature. We argue here that the port of these works from an architecture to another is a major difficulty and that only an effort towards standardisation can help to overcome this drawback.**

## I. Reasons for defending modular open designs

We here define the architecture as the structured organisation of components (or “framework”), embedded on a system, that enables their simultaneous and correct execution, by offering the basic services needed for all of them. In this paper, we will more specifically consider software aspects. Modularity will be defined as the ability, for this software, to receive new components that were not included in the original release, thus enabling, *a posteriori*, to extend its functionalities. It will be moreover “open” if interfaces for writing these modules are public, so that a person different from the developper that originally implemented the code can produce some. For instance, Linux is such a system, in which hardware drivers can easily be added by third parties. And so is Windows: these two characteristics are effectively not contradictory with commercial or property policies and do not mean that the original sources must be unveiled.

Recent articles on architectures often insist on their modularity. This must not only be considered a “commercial” announce but, indeed, softwares that satisfy this property offer many interests. First of all, as they are able to receive all sorts of modules, provided that the latter respect the defined interfaces, they can potentially attract many robotic actors and their adoption is eased. The direct corollary of wide-spread architectures is to provide this users community with a common framework, enhancing the possibilities of sharing and exchanging competencies. As a matter of fact, it also enables to validate concurrent approaches in the same conditions/environment for more relevant comparison results.

Besides, modularity permits to focus one’s development only on particular aspects, while working algorithms can be re-used. Hence, there is no need anymore, when conducting a specific research, to redevelop an entire system to test it, but one can take benefit of an already existing complete framework, in which to integrate and evaluate one’s own work.

But interests are not limited to these ones. Indeed, the vehicle (more generally the platform) in which the modular framework will be embedded will allow a fast integration of different modules or their easy replacement. This is particularly interesting for all users. Laboratories take benefit from the flexibility of such structures, manufacturers can easily provide updates or new functionalities to their clients. Finally, end-users, such as the Department of Defence, can both take advantage of such open-targets to support their research programmes and, depending of the mission needs, to get reconfigurable operational systems. Recently, the French Ground Army confirmed that the MiniRoC concepts of ground robots, see [10], presenting such modular characteristics, was of great interest as they would allow soldiers to only take with them the absolutely necessary modules related to their actual task.

By extension, if on the one hand open architectures compel to conform to given software interfaces, on the other hand they make no assumptions as regarding to the underlying hardware. Such frameworks are, ideally, completely independent from platforms, processors or electronics. Said another way, they offer the possibility to evolve as technologies progress : it remains up-to-date. For end-users essentially, it is a guarantee of durability and, consequently, it requires to train maintainers only for one type of software. This durability is however achievable only by ensuring backward compatibility, so that modules running on older versions of the software can be executed in latest releases.

For the sake of exhaustivity, the same arguments that talk in favour of modular architectures on a single robot also are valid when tackling the multi-robots context, as it will help to gather an un-predefined number of agents within a collaborating team. This property is then known as extensibility or scalability.

Thus, from the above discussion, the main requirements for an architecture to be modular can be summarised as:

- permit normalized data exchange through the definition of public interfaces and common communication mechanisms;
- enable extensibility by making no assumptions on underlying platform or candidate peripheral hardware;
- be flexible by making no assumptions on missions that

will be given to the robot, since new ones will irremediably be imagined during system's life.

As will be discussed in section II, many architectures developed for about twenty years satisfy these characteritics. Besides, whereas the framework nature - reactive, deliberative or hybrid - is often a central concern, we will note here that modularity is independent from this issue, and that it can be ensured in all cases.

## II. Previous work

### A. Evolution in architectures conception

In the second half of the 80s', Brooks introduced an architecture that can maybe be considered the first modular one, [7]. Contrary to common software structures at that time, which often used sequencial treatments to achieve an action, an organisation based on layers is proposed. Each layer can operate in parallel and corresponds to a particular task to be accomplished, see figure 1. Nowadays designs have inherited this point of view as modules often implement specific behaviours. However, in Brooks' system, layers interactions are based on the subsumption principle (upper layers can block and replace outputs of lower ones) and not on a real standardized data exchange. This may bring to a complex links organisation.

Enhanced exchange schemes are present in most following works. Modules become real "independent computational units", executing concurrently and that can use given communication services, imposed by the architecture, to exchange information. Compared to the subsumption architecture, the level of competence (i.e., the priority of the module) is not fixed *a priori* since all the blocks are considered equal. The choice for the appropriate output can be made, depending on the implementation, by a global referee or another module written by the system designer. DAMN, [18], is a remarkable example of the first category. Each block, called "behaviour" in the sense of Brooks', issues votes in favour of a specific possible action that are then collected by an arbiter (figure 2). The final effective command is, schematically, a weighted sum of these votes.

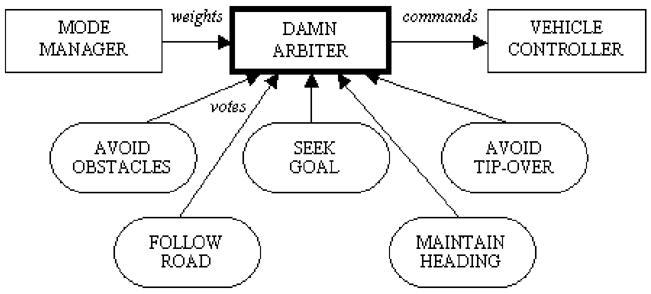

Fig. 2. The DAMN architecture as described in [18].

Another famous approach is the one developed at GeorgiaTech by Arkin, [5]. The principles are close to those of DAMN: behaviours, here called "motor schemas", provide their commands to a process that sums and normalizes them using the potential fields method. A sligth difference is however introduced since a homeostatic control system is added: this can be thought as a bus that collects state variables from robot internal sensors and broadcast them to all of the motor schemas. Resulting monitoring information are used both to influence internal parameters of behaviours and their relative weights. The structure is synthesized on figure 3.

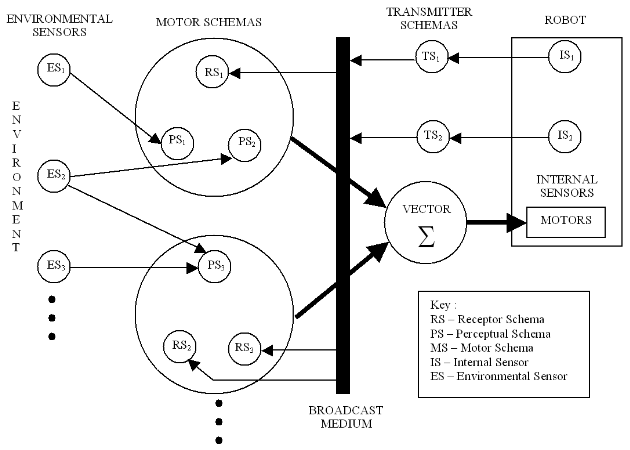

Fig. 3. AuRA principle. This figure both shows the fusion process between two motor schemas and the homeostatic control system that regulates the performance of the overall architecture.

However, theoretically, modularity does not only apply to behaviours (i.e., reactive modules) but also to deliberative capabilities of robots. If the complete AuRA framework already integrates a planning component (as well as a layer responsible for user-robot interaction that can convey human decisions), the reasoning capacities are fixed once for all. But there exists practical cases for which these capacities are more flexible.

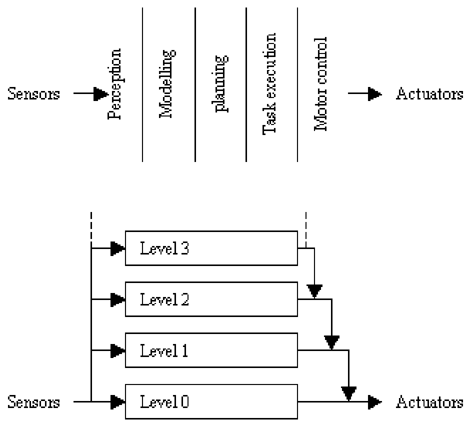

Fig. 1. Whereas traditional approaches presented a sequential organisation (above schema), Brooks proposed a structure where layers, often corresponding to a given level of competence, can execute in parallel and influence the global robot behaviour by subsuming outputs of lower ones. Figures are taken from [7].

An army is a typical example of an efficient organisation in which deliberative agents are not gathered in a centralized structure, as each soldier is not only able to act but also to learn, acquire experience, and use complex reasonings to succeed in his elementary task. And a robot architecture is, in some way, comparable: in both cases, an upper objective (the goal of the robot or of the army) can be achieved by getting elementary agents (modules or men) to work in a coordinated fashion (i.e., respecting rules imposed by the architecture or the hierarchy). Such comparisons naturally lead to propose new robot frameworks, in which deliberative capacities, and not only reactive behaviours, are also designed in a modular manner.

Albus' researches on 4-D/RCS[1] have been conducted based, partly, on these reflexions and lead to a node-oriented architecture. Each elementary component, called a node, integrates sensory processings and reactive parts, like "classical" modules, but can also simultaneously gather modeling, learning and reasoning capabilities, as shown figure 4. Each node is then arranged within a global hierarchy modeled on the military structure. A natural way of implementing this architecture is to grant more deliberative responsibilities to nodes that are placed high in the hierarchy, whereas lower nodes are rather dedicated to information processing. Furthermore, by construction, this framework is multirobot-ready: from a macroscopic point of view, a robot can itself be considered a node and teams of robots can hence be constituted the same way nodes are structured inside each robot. More detailed explanations can be found in [2].

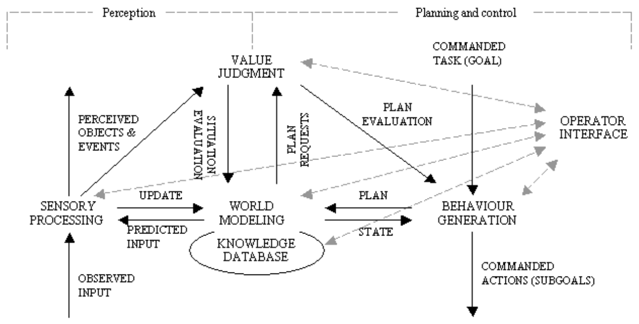

Fig. 4. A typical 4-D/RCS node.

The emergence of modern programming methods, i.e. object-oriented (OO) approaches, hugely contributed to ease the implementation of the above concepts. Inheritance and related mechanisms (polymorphism, methods over-writing) are directly useful to derive efficient modules and permit to easily extend robots capacities, whereas the encapsulation of properties and methods enables objects to share only the useful interfaces. But these approaches did not only reveal interesting for programmers but also inspired the conception of recent architectures. The most appealing one is probably CLARAty in which the whole functional layer is thought as a hierarchy of objects. A simple example is shown on figure 5, whereas very detailed explanations, including coding considerations, can be found in [19]. One interest of OO-conception is to provide all the mechanisms to build proper extensible interfaces, without imposing any limitations on the way objects are internally implemented. COSARC is a very recent example of this trend: it uses an extension of OO-methods (the component-based approach) to define four types of components which internal structure is described with Petri Nets, please refer to [4] for details. This is a relevant illustration of the ability of object-based languages to both ease the development of open frameworks, satisfying modularity requirements, and take benefit from any other recognized approach for the modelisation of components behaviour, here Petri Nets.

[1]4-D/RCS is the architecture that is embedded on the Demo III Experimental Unmanned Vehicle (XUV), a project supported by the American DoD. This probably explains the origin of the parallel between robots architecture and military organisation.

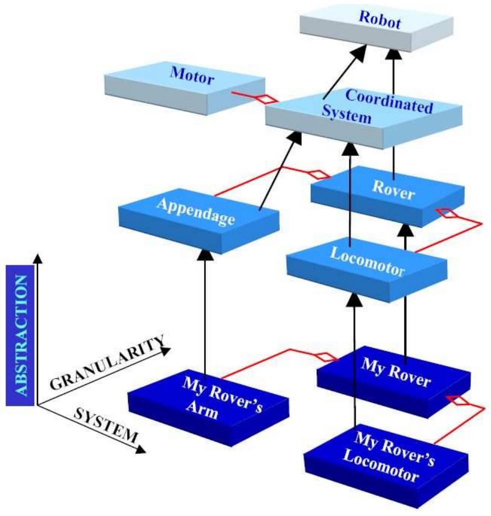

Fig. 5. This example illustrates the object-oriented design of the functional layer of CLARAty. Note that, like all moderns frameworks, it also provides a decision layer, not shown here. But although most other architectures split the executive and planning levels, these functionalities are here gathered within the same layer, for consistency reasons. See [20] for a discussion on this point.

As noted above, the same requirements of modularity and extensibility are needed in the multi-robots context. Here, the main constraint is to enable the adjonction and the removal of agents within the team. If some architectures, like 4-D/RCS, inherently have the capacity to manage several robots, they should also tackle the "fault-tolerancy" problem. That is to say, the lost of a robot or of the communication channel, during the mission, must not induce the failure of the whole team. Most of the time, specific coordination schemes are thus added to the upper layers of the architectures and run all the processes needed to manage a team (especially scalability mechanisms, communication strategies and task allocation). Among all approaches proposed, let us quote ALLIANCE and

TABLE I

TECHNOLOGY READINESS LEVELS

| Low maturity | |
|---|---|
| 1 | Basic principles of technology observed & reported |
| 2 | Technology concept and/or application formulated |
| 3 | Analytical and laboratory studies to validate analytical predictions |
| Medium maturity | |
| 4 | Component and/or basic sub-system technology valid in lab environment |
| 5 | Component and/or basic sub-system technology valid in relevant environment |
| 6 | System/sub-system technology model or prototype demo in relevant environment |
| High maturity | |
| 7 | System technology prototype demo in an operational environment |
| 8 | System technology qualified through test & demonstration |
| 9 | System technology 'qualified' through successful mission operations |

M+. The first one mainly focuses on determining an efficient fault-tolerant scheme: details can be found in [17]. The second one, [6], offers an alternative based on negociations between the robots. It has been successfully integrated in the LAAS-CNRS framework, [1], that hence provides a complete open architecture gathering all functionalities, from reactive behaviours to decisional capacities, for a lonely robot or a team of agents.

Besides, all above quoted works do not remain pure theorical concepts. Many of them have been ported on some vehicles and proved to be relevant potential candidates for real applications.

## B. Introducing Technology Readiness Levels

Technology Readiness Levels (TRLs) were initially created by NASA in 1995 and were officialy adopted by the American Ministry of Defence (MoD) in 2001. This referential aims at assessing the maturity of technologies so as to reduce the risks related to acquisition programmes. It consists of nine levels, of increasing maturity, that apply to individual technologies (not to entire systems). The TRLs grid, copied from [8], is given in table I.

Nowadays results and experimentations show that levels 5/6 can currently be reached. 4-D/RCS framework has thus been ported to the American XUV (Experimental Unmanned Vehicle) and its ability to host functionnal modules could be demonstrated. Besides, Albus' report [3] comforts us in our evaluation of the maturity of this architecture when asserting that "the tests are designed to determine whether the Demo III XUVs have achieved technology readiness level six".

In France, the same encouraging conclusions can be drawn. All along SYRANO[2] project, industrials have shown their ability to deploy a modular architecture (although proprietary) on a prototype tank and their capacity to incrementally add new functionalities when they become available. Demonstrations with this vehicle took place in the military camp of Mourmelon, in a quasi-operational context, as related in [16]. Because only teleoperated functions have been used, not all possibilities of the architecture have been extensively tested, and these demonstrations "only" correspond to the TRL-5.

However, a vehicle such as Syrano is TRL-6-ready as, in theory, advanced autonomous modules can be added the same way. Moreover results achieved by some laboratories comfort this point of view. For instance, experiments conducted by LAAS-CNRS in autonomous navigation demonstrated the ability of a robot to automatically explore an unknown area. During these tests, all functionalities were used: planning, supervision, autonomous modules management as well as human-robot interfaces (at least to send mission reports and receive high level orders from the operator). The multi-robot scheme is itself under 'validation' as, in some current projects, additional aerial information is provided by a Blimp UAV to assist the robot in its navigation task.

Since 2004, this laboratory has also been running a robot equipped with the same architecture, [1], in Cité de l'Espace, Toulouse. This robot serves as a guide for visitors, interacting with them to retrieve their questions, then conducting them to the desired point. Since people are obviously neither roboticians nor technicians, the environment of the robot can be considered operational. This experiment can maybe not claim the TRL-7 title, which would require huge reliability, but definitely proves that this level is now achievable.

Of course, qualitatively judging the architecture, on an experiment that takes into account the whole system, is quite difficult and its exact contribution to the overall performance is hard to deduce. But at least, this means that services expected from the software are functional and that internal communications between framework components work. When, moreover, the architecture has been ported on heterogeneous systems (the SYRANO one was previously integrated on a robot jeep, called DARDS, and is reused for a future demining system), it can be argued that modularity and portability have been validated. In conclusion, based on the observation of above quoted experiments, giving TRL-6 as the current achieved maturity level seems relevant.

Some complementary methods also exist to assess the maturity of a whole system, System Readiness Levels. A brief review of SRLs ([9] and table II) confirms that these vehicles have reached levels 6-7, meaning that demonstrations have been conducted successfully in representative environment.

[2]SYRANO is a teleoperated prototype vehicle, with simple autonomous behaviours, for reconnaissance and detection of potential adverse targets. It aims at evolving in open areas.

TABLE II
SYSTEM READINESS LEVELS

| | |
|---|---|
| 1 | User requirements defined |
| 2 | System requirements defined |
| 3 | Architectural design refined |
| 4 | Detailed design is nominally complete |
| 5 | Sub-systems verification in laboratory environment |
| 6 | Sub-system verification in representative integration environment |
| 7 | System prototype demonstration in a representative integration environment |
| 8 | Pre-production system completed and demonstrated in a representative operational environment |
| 9 | System proven through successful representative mission profile |

Nevertheless, the previous section raises a major concern, as it shows that numerous architectures have been develop *concurrently*: if each one, separatly, does satisfy the modularity requirements, components developed for one of them cannot be ported to another. A higher level of specification is still missing that would ensure interoperability, a property of real importance, especially in a military context. Consequently, as upcoming programmes can not systematically rely on a standard, they only correspond to SRL-2. It is thus quite urgent to emphasize the research effort on this point, as will be discussed in section III. We will first focus on a relevant American example, JAUS, then present some French research programmes that tackle the issue.

## III. TOWARDS STANDARDISATION

### A. American proposals

*1) History:* In the last 20 years, a large number of unmanned systems have been developed by US companies in response to the American DoD, but most of them are task-specific and non-interoperable. Therefore, in 1994, JAUGS, an effort to avoid the pitfalls of "eaches" in an expanding domain, was initiated. It was still limited to ground systems (the "G" actually stands for "ground").

In 1998, OUSD (Office of the Under Secretary of Defence) chartered a working group, consisting of members from the government, industry and academia, to develop an architecture for unmanned systems. It set itself *five targets*:

- support all classes of unmanned systems ;
- advocate rapid technology insertion ;
- provide interoperable Operating Control Units (OCUs) ;
- provide interchangeable/interoperable payloads ;
- provide interoperable unmanned systems.

The resulting Joint Architecture for Unmanned Systems (JAUS), available for use by defence, academic and commercial sectors, is an upper level design for the interfaces within the domain of unmanned vehicles. It aims at being independent from technology, computer hardware, operator use and vehicle platforms, and isolated from mission. It is a component based, message-passing framework that specifies data formats and methods of communication between computing entities of unmanned systems.

Its final goal is to reduce development and integration times, ownership cost, and to enable an expanded range of vendors by providing a framework for technology insertion.

*2) JAUS specifications:* to date, two documents describe the JAUS architecture: the Reference Architecture specification (RA), [13], and the Domain Model (DM), [12].

*a) Domain model:* the analysis conducted in the DM on the five above targets, along with the study of military contracts constraints, urged to define five main requirements on messages within the architecture. Indeed they need to be independent from: 1. vehicle platform, 2. mission, 3. computer resource, 4. technology and 5. operator use. This document is also a tool with which customers/users can define both near and far term operational requirements, for unmanned systems, based on mission needs; and by defining far-term capabilities, the JAUS Domain Model can actually be considered a "road map" for developers to focus research and design efforts to support these future requirements.

In a word, the domain model is a common language which contains three distinct elements : functional capabilities (FC), informational capabilities (IC), and device groups (DG). The first ones, all documented in DM so that to ease dialog between users and developers, are a set of capabilities with similar functional purposes. Eleven categories are identified, that permit to describe the abilities of an unmanned system: command, manœuvre, navigation, communication, payload, safety, security, resource management, maintenance, training and automatic configuration. In parallel with the functional description, Informational Capabilities provide a representation

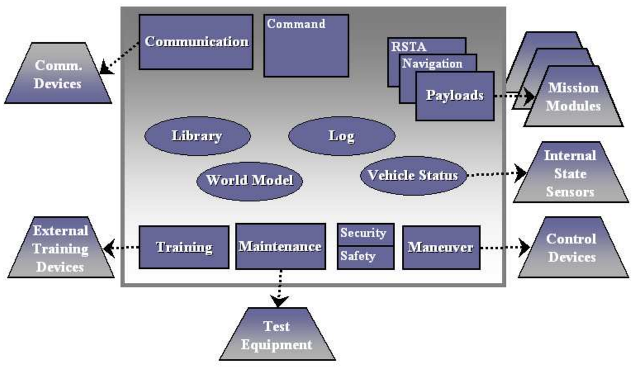

Fig. 6. JAUS domain model representation.

of what unmanned systems (should) know. They are groupings of similar types of information. Five categories are depicted in DM: vehicle status, world model, library, logistics, time/date. Finally, device groups are a classification of sensors and/or effectors that are used for similar functions. Functional and informational capabilities may interface with device groups, but the JAUS domain model does not define these interfaces. Figure 6 summarises the DM representation.

*b) Reference architecture specification:* in the development cycle, the specification of capabilities described in the DM will always precede those that appear in the RA, whose main purpose is to detail all functions and messages that shall be employed to design new components. All currently defined messages as well as rules that govern messaging are also depicted in this second document. It is worth noting that messaging is the sole accepted method to communicate between components.

The RA specification comprises three parts. First of them, the architecture framework provides a description of the structure of JAUS-based systems. Actually, unmanned systems are seen as a hierarchical topology, shown on figure 7. A system is a logical grouping of one or more subsystems, which are independent and distinct units. A node, in such a topology, is a ‘black-box’ containing all the hardware and software necessary to provide a complete service, for example a mobility or a payload controller. A component is the lowest level of decomposition in the JAUS hierarchy: it is an executable task or process. All the components, which may be found within an unmanned system, are listed in this first part of the RA. The above defined topology is very flexible since the only requirement is that a subsystem be composed of component software, distributed across one or more nodes. Interoperability between intelligent systems is achieved by defining functional components with supported messages. Therefore, the only constraint to be JAUS-compliant is that all messages that pass between components, over networks or via airwaves, shall be JAUS-compatible messages. No other rules are imposed to system engineers. Besides, messages coming from or/and sent to non-JAUS components can have their own protocol.

The definition and the format of those messages are the objects of the second and the third parts of the RA. In the second one, message definition, different classes of messages (command, query, inform, event setup, event notification) and message composition (classically, a header and a data fields) are defined. Messaging protocol is also depicted with the routing strategy, the way to send large data messages, the way to establish a connection between two components, as well as some various messaging rules. The third and final part of the RA, Message Set, presents the details of command code usage for each message (the command code is an information included in the message header).

*c) Other documents:* additional documents support the JAUS standards: a Document Control Plan (DCP) and the Standard Operating Procedures (SOP), [14]. The first one defines the process to update the JAUS DM and RA whereas the second one establishes the charter and organisation for the JAUS working group. A compliance plan, [11], transport plan and user’s handbook are under development.

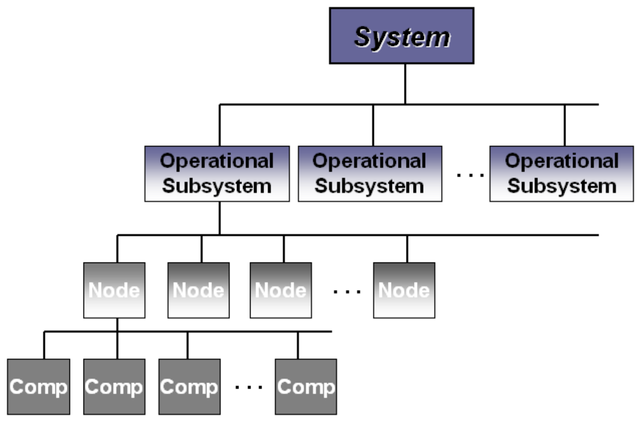

Fig. 7. The reference architecture from JAUS.

*3) Conclusion on JAUS:* JAUS is not an architecture, as defined at the beginning of this article. It is rather a process to ease *communication between users and developers* and to *standardise exchanges of data* within software embedded in an autonomous system. However, nowadays, JAUS is mandated for use by all of the programmes in the Joint Robotics Program (JRP), [15], and numerous American manufacturers begin to follow the requirements. For example, the EOD[3] Man-Transportable Robotic System (MTRS), PackBot, produced by IRobot, is JAUS-compliant. Moreover, JAUS is now recognized as a technical committee within the SAE, Aerospace Council’s Aviation Systems Division (ASD), which name is AS-4 Unmanned Systems.

## B. French government effort

For several years now, French DoD has been preparing a number of studies to get standards to emerge, so that future architectures embedded in military demonstrators be ready for interoperability needs. In the ground robotics field, two major research programmes have been, or are about to be, launched.

At the end of year 2005, the “Démonstrateur BOA” programme was notified to an industrial group (TGS: Thales, GIAT Industries, SAGEM). Roughly, BOA is the equivalent of the American Network Centric Warfare. It aims at proposing new organisations for ground forces (including aerial devices operating to their profit) with a high degree of interoperability between the different units, through advanced communication means. UGVs and mini-UAVS will naturally be part of this structure since they represent a privileged way to retrieve information, even during high-intensity actions, without exposing soldiers, enabling new combat strategies such as indirect firing, see figure 8. Missions which they should respond to are manifold and heterogeneous, from urban fighting, to logistics or reconnaissance. Hence, the challenge:

[3]explosive ordnance disposal

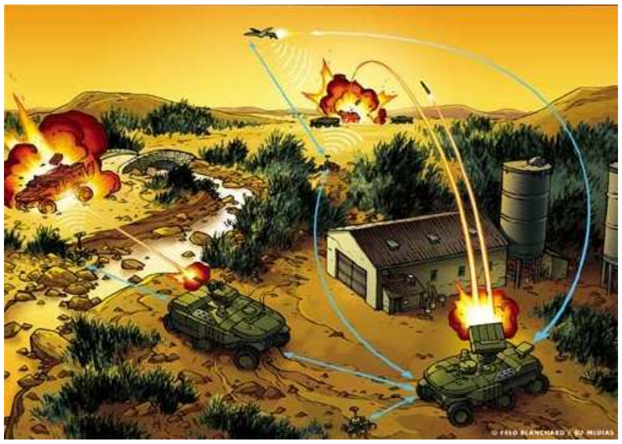

Fig. 8. Official illustration of the BOA concept, showing candidate systems, data exchanges between them and consequent achievable missions.

- integrating unmanned vehicles communications in an already very constraint electromagnetic environment;
- enabling information sharing between robots and with human units;
- getting highly reconfigurable robots/UAVs so that they can quickly be adapted to the actual mission.

The third point can only be achieved through the use of a modular architecture. Moreover, since BOA is still a prospective concept, all possible missions are probably not exhaustively identified; finally, some specific functions, dealing with autonomous capacities of ground vehicles, will be provided by other programmes, the actors of which are not necessarily involved in BOA. These two supplementary aspects imply that the software framework be also open, to allow the desired extensibility. But the second above point is maybe the most critical. The diversity of information sources, and the number of actors that will access them, then encourages to adopt standards for data exchanges.

As a consequence, the DoD insisted on the architecture part of the robots and vehemently required that modularity be achieved at all levels (software as well as hardware), that interfaces be open and can be communicated to third parties. The interoperability constraints were tackled by explicitely asking for the adoption of a standard for data exchanges: JAUS, if obviously not imposed, was quoted as an acceptable solution, so that to clearly illustrate DoD expectations. Finally, it is worth noting that information provided by unmmanned vehicles are often of the same nature (localisation, detection or intelligence data) than those conveied by Battlefield Management Systems, that will be of primordial importance in BOA. Robots are then natural candidates to feed these systems and analyzing data structures used for the latter can also be a relevant source of inspiration.

Although all previous examples are taken from ground robotics fields, the open modular architecture is actually an important subject for the near future UAV systems. Currently, the French DoD is conducting two studies in parallel. The aim for both is: “definition of an open, standardised, modular and evolutionary architecture for a generic and interoperable UAV system”.

The problematic of an UAV system is large and very complex because of the multiple interfaces including the *onboard* (aircrafts, payloads, airworthiness, air traffic management…), *ground* (command, control and exploitation station, recovery, launch…) and *system* (communications, certification, subsystem interfaces, real time synchronization, critical software…) constraints. The studies cover, from the time being, the three different UAVs system categories: tactical, MALE (Medium Altitude Long Endurance) and HALE (High Altitude Long Endurance). The main technical axes of the studies can be summed up with the following:

- define the best configurable and generic architecture;
- improve the system’s performances regarding new hard or software technologies;
- interchangeability of payloads within the “plug and play” concept;
- obtain secure, certifiable and everlasting architecture.

An “open, standardized, modular and evolutionary” architecture is the challenge for the future French UAV systems programmes. Thanks to this, the interoperability will exist throughout the UAV system’s total cycle life (15 to 20 years).

However, one can still argue that these efforts towards standardisation are limited either to ground or to aerial vehicles, and that one still lacks a federative framework. This is mainly the goal of the OISAU research programme. Initiated by ground robotics experts of the DoD, this study for “open and interoperable autonomous systems” actually introduces no assumptions concerning the type of the candidate platforms, that can either be aerian, ground or even marine vehicles. For the first time, it explicitely gathers within a lone coherent programme all the requirements presented above, asking that the resulting architecture enable :

- platform and hardware independency;
- cost reduction thanks to standardisation (thus allowing acquisition and maintenance savings);
- easy integration and replacement of functional modules;
- ability to incrementally proceed to these integration or replacement to allow systems evolution.

This programme is probably of a primordial importance in an effort to get operational robots, that can be introduced in armed forces.

## IV. Conclusion

Indeed, many reasons, technical, commercial or practical, lead to increase the modularity of robots architectures. Along the past ten years, a number of solutions has been proposed and open extensible frameworks are actually available to robots developers and users. Some of these concepts have even been successfully ported on real systems, demonstrating their relevance and maturity.

However, the robotics community still lacks a real federative standard to get unmanned systems interoperable. As a matter of fact, following the American JAUS example, and since interoperability is of crucial importance for the introduction of robots within armed forces, the French DoD has decided to support the development of new normative frameworks. Besides, this effort concerns all fields of robotics, from ground to aerial vehicles, and is at the heart of current research programmes. A major objective of these works is to increase the technology and system readiness levels, i.e. to get more mature and reliable robots.

But, if modularity and standardisation are necessary conditions for architectures to meet all the above discussed requirements, they are not sufficient. Until now, robots are not completely autonomous systems and, even in the future, a supervisory control will be at least kept. Nevertheless there still remains work to determine which role humans deserve and which level of autonomy will be granted to robots. And the conclusions of such a work will impact the needed exchanges, data structures and interfaces that have to take place between the framework components. That is, the basic characteristics that enable modularity will deeply depend on the role of the human in unmanned systems.

## Acknowledgment

We would like to thank Agnès Lechevallier, French DoD, for her contribution on the UAV part.

Special thanks go to Jérôme Lemaire, head of our department, for his help and valuable advice on the content of this paper.

Also note that research works quoted in this paper would probably deserve much more attention than the one that could be granted here. They have all bring relevant results, when no breakthrough, to the field of control architectures. Please refer to the original articles for details and in-depth discussions on their specificities.

## References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab and F. Ingrand, *An Architecture for Autonomy*, International Journal of Robotics Research, 17(4), April 1998.

[2] J.S. Albus, *4-D/RCS: A Reference Model Architecture for Demo III*, NISTIR 5994, National Institute of Standards and Technology, Gaithersburg, MD, March 1997.

[3] J.S. Albus, *Metrics and Performance Measures for Intelligent Unmanned Ground Vehicles*, in proceedings of the Performance Metrics for Intelligent Systems (PerMIS) workshop, 2002.

[4] D. Andreu and R. Passama, *COSARC: COmponent based Software Architecture of Robot Controllers*, in proceedings of the CAR'06 workshop, Montpellier, April 2006.

[5] R. Arkin and T. Balch, *AuRA: Principles and practice in review*, Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2-3, pp. 175-188, 1997.

[6] S.C. Botelho and R. Alami, *M+: a scheme for multi-robot cooperation through negotaited task allocation and achievement*, in proceedings of IEEE International Conference on Robotics and Automation, vol. 2, pp. 1234-1239, Michigan, May 1999.

[7] R.A. Brooks, *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation, vol. RA-2, no. 1, pp. 14-23, April 1986.

[8] Future Business Group, *Technology Readiness Levels (TRLs) guidance*, FBG/36/10, January 11th 2005.

[9] Future Business Group, *System maturity assessment using System Readiness Levels guidance*, FBG/43/01/01, v3.2, February 16th 2006.

[10] L. Gillet and Y.L. Lunel, *Des robots pour assister le combattant débarqué en zone urbaine : les démonstrateurs MiniRoC*, L'armement, no. 85, March 2004.

[11] JAUS, The Joint Architecture for Unmanned Systems, Compliance Specification, v. 1.1, March 10th 2005.

[12] JAUS, The Joint Architecture for Unmanned Systems, Domain Model, volume I, v. 3.2, March 10th 2005.

[13] JAUS, The Joint Architecture for Unmanned Systems, Reference Architecture SPecification, volume II, parts 1-3, v. 3.2, August 13th 2004.

[14] JAUS, The Joint Architecture for Unmanned Systems, Standard Operating Procedures, v. 1.5, October 10th 2002.

[15] JRP, Joint Robotics Program, Master Plan, FY2005.

[16] J.G. Morillon, O. Lecointe, J.-P. Quin, M. Tissedre, C. Lewandowski, T. Gauthier, F. Le Gusquet and F. Useo, *SYRANO: a ground robotic system for target acquisition and neutralization*, in proceedings of SPIE Aerosense, Unmanned Ground Vehicle Technology V, vol. 5083, pp. 38-51, September 2003.

[17] L.E. Parker, *ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation*, IEEE Transactions on Robotics and Automation, vol. 14, no. 2, pp. 220-240, 1998.

[18] J.K. Rosenblatt, *DAMN: A Distributed Architecture for Mobile Navigation*, in proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, H. Hexmoor & D. Kortemkamps (Eds.), Menlo Park, CA:AAAI Press.

[19] R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras and H. Das, *CLARAty: Coupled Layer Architecture for Robotic Autonomy*, technical report, Jet Propulsion Laboratory, December 2000.

[20] R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras and H. Das, *The CLARAty architecture for robotic autonomy*, in proceedings of IEEE Aerospace Conference, Montana, March 2001.