Towards a Robust System-of-Systems Control Architecture for Robotics

Guy Burroughes, Rob Skilton, Jean-Jacque Honore, Rob Buckingham

Abstract

This paper presents the methodology and results of developing and implementing a system-of-systems architecture for controlling many robotic systems within the context of industrial facilities, where the life of the facility will be far longer than the life of the robotics, due to software updates and accelerated digital obsolescence. The use cases are the JET and ITER fusion research reactors. Remote maintenance is both device defining and mission critical for a future fusion reactor and we foresee the need to utilise a considerable number of robotic devices within an overall digital control architecture. The paper will explore the challenges of long term management of digital complexity using the real experience of JET over the last 20 years.

Introduction

For large-scale experimental devices like ITER to maximise their science value per dollar, they inevitably require a high level of flexibility and future proofing so that they can track the science that they advance. However, as learnt from JET these devices have a natural tendency to become larger, more complex, and more hazardous, they inescapably require increased levels of remote maintenance [1]. The key problems at the heart of remote maintenance are:

- Dealing with changing maintenance requirements following the changing needs of the experimental device throughout its lifetime.
- Managing obsolescence in long-life remote maintenance systems, and the impact of obsolescence mitigation activities, such as replacing subsystems with newer, more readily available systems.
- Allowing for the integration of many items from different suppliers, possibly from dozens of different collaborating nations, into a functioning integrated capability including managing interfaces, operators and training requirements. ITER, for example will require more than 8 highly complex and bespoke remote maintenance systems to be provided from contributors worldwide including Europe, Japan, India, China, and Korea [2].

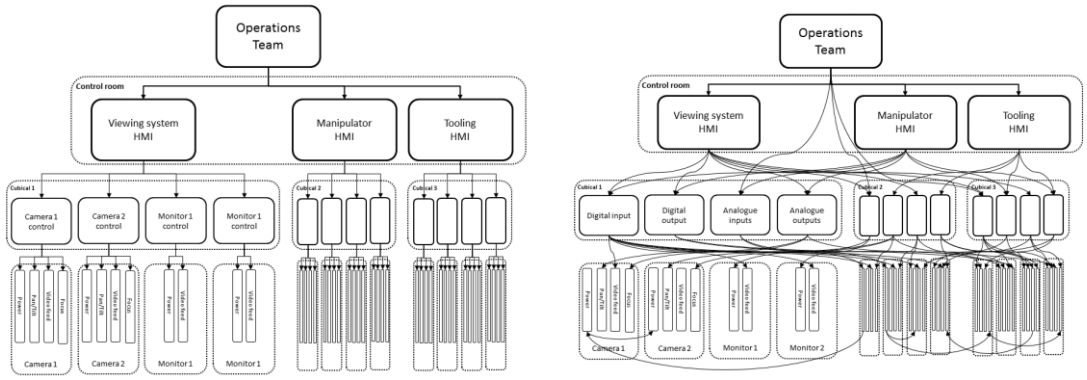

Figure 1:The idealised (left) and actual (right) JET remote maintenance system architecture.

Figure 1 shows a comparison between a simplified and idealised JET RM architecture compared to an approximation to current structure. The idealised hierarchical structure that has clear separations and

encapsulation of functionality. For example, the viewing system Human Machine Interface (HMI) communicates with modules within a hardware Cubical and the Cubical controls the physical cameras.

These are the same issues that the robotic community are facing in general. These issues are limiting the penetration of robotics into fields that require them. What will follow is a presentation of the current architectural issues limiting robots in this space and causing the aforementioned issues. Following this, a short discussion into currently available potential solutions. Finally, a discussion and conclusion will be drawn about the topic.

Issues for the Next Generation Robotics

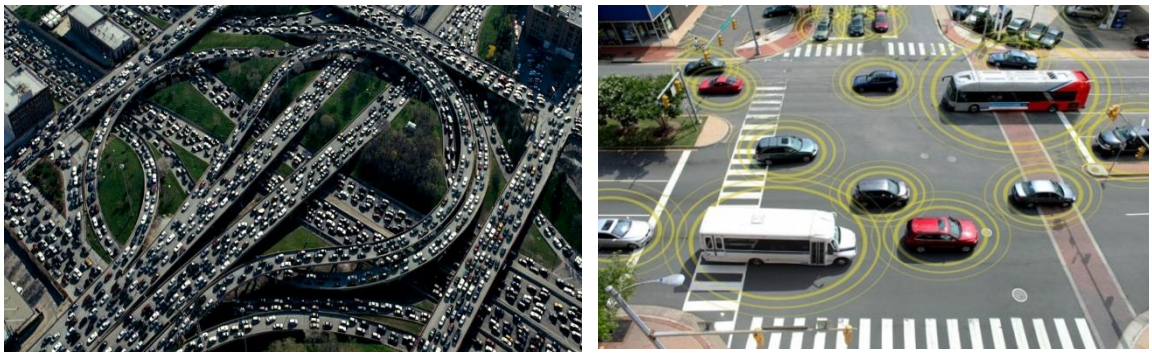

Figure 2: Nationwide Systems of Systems.

Robotics will and are already causing the next Industrial Revolution. However, before complete robotic dominance is achieved 9 major architectural issues need to be addressed. These issues are present today in JET and ITER today, and will be present in all systems in the coming years. A platform for robotics needs to be designed solving the following 9 issues:

1. The future robot architecture will mix advanced mechatronics and cutting edge deep-learning and AI techniques effectively and safely, and thus need to be resilient and supportive to **mixed real-time** constraints (i.e. soft and hard) and enable easy mixed communications and interactions.
2. Future robotic systems will be a mix of wired, Wi-Fi, and 5G systems and thus be able to support enable effective communication between all of these domains on a **Realistic network** configuration.
3. One of the biggest issues evident, is the scale of future robotic systems. The entirety of the road network (cars and infrastructure) will become a massive robotic **system of systems**, capable of controlling every element at every level. A manufacturer might want to inspect the functioning of one wiper blade on one car for a defect or ambulance might want to request a clear route on traffic lights all this possible, all this achievable now but for a lack of architecture. Thus, a platform should infinitely **scalable**, whilst also making the complexity manageable for developers and users alike.
4. Now consider this Nation-wide system of systems framework in control of 1 tonne 70 mph projectiles. Clearly, for the platforms **Cyber-security** will be paramount. This will need to be beyond mere communication encryption; authentication, verification of actors, identification and elimination malicious elements should all be considered. Stuxnet proved that an air gap is not enough, and Cyber-security must be taken extremely seriously.
5. Similarly, **Functional Safety** must be ingrained into the system. Robot operation should have underlying layer of safety that cannot be avoided or circumvented.

6. Now, consider a road network or nuclear waste management facility, they will both inevitably filled with advanced robotics soon. However, these facilities will have a service life of decades and must maintainable remembering the accelerating pace of development. These plants should be upgrade-able to whatever the future might deliver and require. Thus, the platform should be backwards-, cross-, and **forward-Compatible.**
7. One of the more important issues for Industry is the platform must be **Stable, maintainable, and sustainable.** Industry is not willing or capable of support a constantly morphing platform, especially when applications will have lifetimes in the decades. However, this requirement on a platform should not limit upgradability.
8. Similarly, the platform should be compose-able or **Plug and Play** by both expert and executive users, whilst not limiting functionality. It cannot be expected that all users of the systems will have PhDs in Robotics and AI.
9. Finally, the platform must be **AI-friendly.** It must enable the Big Data contained within to be used in Deep Learning algorithms and must support Autonomous control in a safe and constrained manner.

The 9 issues are readily solvable today. All that is needed is a centralised effort to create a platform that solves all problems at once. It is worth noting that "Done" is better than "Perfect". Another benefit from a centralised effort to design a platform is the possibility of creating a new, open marketplace for robotics.

ROS

In academia, there has been a robotic revolution, since 2008 researcher have primarily developing their robotic solution on ROS (Robot Operating System) [3]. ROS is a Robotics middleware, it provides services designed for heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Despite the importance of reactivity and low latency in robot control, ROS, itself, is not a real-time OS (RTOS), though it is possible to integrate ROS with real-time code it is not inherently supportive.

For academia the open-source solution has allowed for rapid development and collaboration of both hardware and software. Groups and individuals actively distribute their work allowing for academics to reuse and create cutting-edge robotics out of the community modules. However, this inevitably involves complex integration efforts.

Originally ROS was not intended for industrial use, but rather started in 2007 as a project at the Stanford Artificial Intelligence Laboratory in conjunction with more than twenty institutions that collaborated on the development model. It was not until 2011 that Southwest Research Institute (SwRI), Willow Garage (now disbanded), and Yaskawa-Motoman Robotics expanded on ROS with ROS-Industrial to bring advanced robotics software to the industrial automation domain. The outcome was ROS-I, a software architecture based on ROS, controlling an industrial manipulator. Once the ROS-I framework was in place, the extensive ROS community was opened to industrial robot hardware. However, this did not deal with issues in reverse, what would convince industry to start using ROS?

ROS has been amazing for the Robotics community; however, there has been concerns raised that it will not be sufficient to serve academia and industry in the future [5]. Primarily, ROS does not solve any of the 9 issues outlined in the previous section.

ROS 2.0 is the next iteration that will focus on multi-robot interaction, Small embedded platforms, real-time systems, and Non-ideal networks. The biggest change is move from a home-rolled communications platform to DDS. DDS (Data Distribution Service) is a real-time middleware produced by OMG, that aims to enable scalable, real-time, dependable, high-performance and interoperable data exchanges using a publish–subscribe pattern. This step is the first step to solve issues 1, 2, and 3; however, it is not a complete solution. ROS 2.0 does not offer a framework for mixed real-time, only the capability to support real-time systems. ROS 2.0 thanks to DDS, ROS2.0 can operate in Realistic networks, although it does not offer a framework for mitigating their issues. Similarly, DDS offers encrypted communications, but this is not synonymous with cyber-security [4]. Additionally, ROS2.0 will support multi-robot interaction; however, it is still to prove scalability and in no way intended to work on the scale described in issue 3.

Similarly, MOOS [6] is affected by the exact same problems as ROS, without the promise of real-time and determinism of ROS 2.0. MOOS does offer the MOOSDB, which acts as a central server for all communicated information in the system. Another ROS-like system is EPICS [7], The Experimental Physics and Industrial Control System, is a software environment used to develop and implement distributed control systems to operate devices such as particle accelerators, telescopes and other large experiments. As with ROS, EPICS uses client/server and publish/subscribe techniques to communicate between the various computers.

GenRobot is the generic low level control system software controller for the ITER RH Control System. It aims to solve the problems of control robots in uniform, reliable, and SIL2 fashion, which is crucial. But is not the same problem as posed in this paper. Similarly, OROCOS [8] falls into the same field of solving the problem of a deterministic middleware for robotics but other than this valuable distinction suffers from the same issues as ROS.

Next Steps

The next step should be the formation of a common neutral committee actively engaging both academia and industry to discuss these issues; all with the aim of quickly delivering a solution to the 9 issues. Whether this forms ROS3.0 or something else doesn't matter; however, timeliness is key. The committee also should agree the form in which the platform takes: open standard, maintained open-source, licensed product held by a neutral party, etc.

For inspiration, the research and development of Internet of Things (IoT) architectures and technologies are showing great promise. IoT is a fuzzy term, roughly defined, IoT is: a culture of integrating imbedded intelligences into a distributed range of objects/systems that are all capable of communicating in a mostly indiscriminate manner over a wider area network. By this definition, IoT outlines a design pattern that inherently has a fine granularity and allows for prevalent status monitoring/control of countless systems. Due to the limitations of the embedded devices, there is also lesser temptation to fit extra, unrelated functionality onto a module; passively encouraging more manageable partitioning with better cohesion. In theory, all IoT devices should be modular, easily replaceable, and easily upgrade-able. Clearly the main issue that IoT addresses is scale, this is one on of the major issues that must be addressed for future remote maintenance systems. A remote maintenance system is not a primary use case for IoT. However, research is being conducted into how include high complex, intelligent, or actuating devices into IoT. One such effort is the Industrial Internet of Things (IIoT) is being researched as a method for manufacturing more flexible, cost effective, and responsive to changes in customer demands, that could be applied to a remote

maintenance facility. However, a major concern surrounding the IIoT is interoperability between devices and machines that function within different protocols and architectures [9]. Several methods have been proposed to cope with the issues of communications, utilizing various middleware based techniques. In particularly, specific issues such as Sensing and Actuating IIoT devices have been discussed; although the problem is still an open problem [10].

Primarily, IoT addresses the key issue of scalability while also providing a fine granularity of control and information gathering. IoT related developments may also prove to provide other great benefits like greater support for mass automation, artificial intelligence systems and obsolescence management. However, there are still many challenges to overcome with regards to standardising these systems and supporting their continued used over long periods of time.

References

[1] Crofts, Buckingham, Rob, and Antony Loving. "Remote-handling challenges in fusion research and beyond." Nature Physics 12.5 (2016): 391-393.

[2] Damiani, C., et al. "The European contribution to the ITER Remote Maintenance." *Fusion Engineering and Design* 89.9 (2014): 2251-2256

[3] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." *ICRA workshop on open source software*. Vol. 3. No. 3.2. 2009.

[4] Morante, Santiago, Juan G. Victores, and Carlos Balaguer. "Cryptobotics: Why robots need cyber safety." *Frontiers in Robotics and AI* 2 (2015): 23.

[5] Cousins, Steve. "Is ROS Good for Robotics? [ROS Topics]." *IEEE Robotics & Automation Magazine* 19.2 (2012): 13-14.

[6] Newman, P. "The MOOS-cross platform software for robotics research." *URL http://www. robots. ox. ac. uk/% 7Emobile/MOOS/wiki/pmwiki. php* (2003).

[7] Lewis, Stephen A. "Overview of the Experimental Physics and Industrial Control System: EPICS." *h ttp://csg. lbl. gov/EPICS/OverView. pdf* (2000).

[8] Bruyninckx, Herman, Peter Soetens, and Bob Koninckx. "The real-time motion control core of the Orocos project." *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 2. IEEE, 2003.

[9] Jung, Jieun, et al. "Design of Smart Factory Web Services Based on the Industrial Internet of Things." *Proceedings of the 50th Hawaii International Conference on System Sciences*. 2017.

[10] Abdmeziem, Mohammed Riyadh, Djamel Tandjaoui, and Imed Romdhani. "Architecting the internet of things: state of the art." *Robots and Sensor Clouds*. Springer International Publishing, 2016. 55-75.