Evaluating ITER Remote Handling middleware concepts

J.F. Koning[a], C.J.M. Heemskerk[b], P. Schoen[b], D. Smedinga[b], A.H. Boode[c], D.T. Hamilton[d]

[a]FOM Institute DIFFER, Association EURATOM-FOM, Partner in the Trilateral Euregio Cluster and ITER-NL, PO Box 1207, 3430 BE Nieuwegein, the Netherlands, www.differ.nl
[b]Heemskerk Innovative Technology, Noordwijk, The Netherlands
[c]University of Applied Sciences InHolland, Alkmaar, The Netherlands
[d]ITER Organization, Route de Vinon sur Verdon, 13115 Saint Paul Lez Durance, France

Remote maintenance activities in ITER will be performed by a unique set of hardware systems, supported by an extensive software kit. A layer of middleware will manage and control a complex set of interconnections between teams of operators, hardware devices in various operating theatres, and databases managing tool and task logistics. The middleware is driven by constraints on amounts and timing of data like real-time control loops, camera images, and database access.

The Remote Handling Study Centre (RHSC), located at FOM institute DIFFER, has a 4-operator work cell in an ITER relevant RH control room setup which connects to a virtual hot cell back-end. The Centre is developing and testing flexible integration of the Control Room components, resulting in proof-of-concept tests of this middleware layer. SW components studied include generic human-machine interface software, a prototype of a RH operations management system, and a distributed virtual reality system supporting multi-screen, multi-actor, and multiple independent views. Real-time rigid body dynamics and contact interaction simulation software supports simulation of structural deformation, "augmented reality" operations and operator training.

The paper presents generic requirements and conceptual design of middleware components and Operations Management System in the context of a RH Control Room work cell. The simulation software is analyzed for real-time performance and it is argued that it is critical for middleware to have complete control over the physical network to be able to guarantee bandwidth and latency to the components.

Keywords: Remote Handling, haptic, virtual reality, master-slave, middleware

1. Introduction

ITER has established procedures to develop validated RH designs and methods [1][2]. This implies that the accessibility and handle-ability of an ITER component by remote methods, needs to be reviewed. Subject to the established RH class of the task, the maintenance procedures need to be detailed before the system enters the operational stage.

Starting from the conceptual design stage, it is necessary to analyze RH compatibility. Virtual Reality (VR) simulation is a cost-effective and flexible method to enable RH Compatibility Assessment (RHCA) [2].

The Remote Handling Study Centre (RHSC) [2] enables VR simulation for RH procedure detailing and validation. Operators can interact individually or as a team with the virtual environment by different means, including common computer interfaces such as keyboards, mice, joysticks and spacemice. Monitors provide visual feedback through configurable views. Additionally, the virtual environment boosts a rigid body dynamics simulator with contact interaction. Advanced interaction with the rigid body simulator is possible through haptic master devices, allowing for user position/force input, and for force feedback to the user.

Currently a desktop solution and a full-scale master arm are available: respectively Sensable Phantom Omni devices and a Haption Virtuose 6D35-45. Various remote scenes are available including a hot cell mockup and a scene used for benchmarking operators [3][4][5], which has also been benchmarked on a hardware test bed [6]. The software components used in the RHSC reflect a number of the functionalities that ITER seeks in the RH Control System (RHCS) architecture, albeit not yet in the modular form that is requested, where a middleware layer will be connecting and servicing the modules with data [7]. The RHSC strives to implement a prototype of the ITER-RHCS.

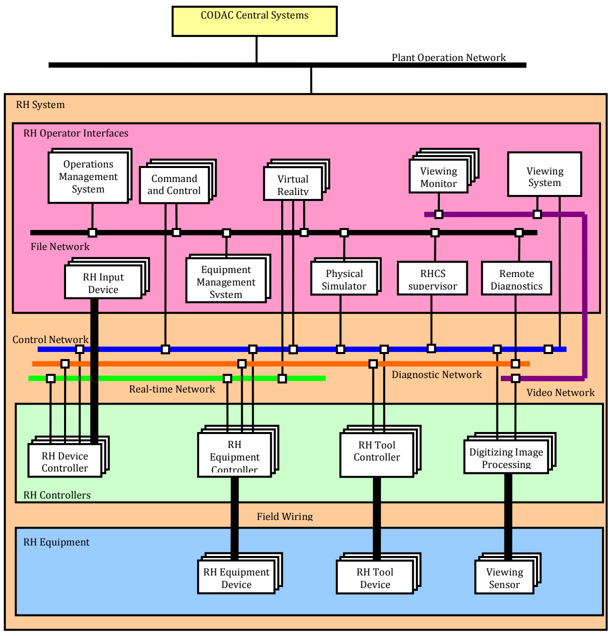


Figure 1.1 RHCS layout concept with RHCS supervisor added

author's email: j.f.koning@differ.nl

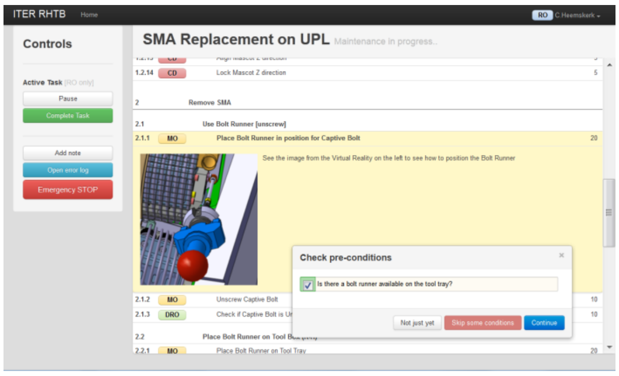
This paper describes steps taken to come to a RHCS middleware prototype based on the available software components at the RHSC. The driving requirements are presented and reviewed. Several generically available middleware solutions and an ad-hoc solution are conceptually implemented. Compliance with requirements and performance are analyzed wherever possible.

2. RHCS required functional components

The RHSC is developing functionality roughly along the lines of Figure 1.1 [7]. Most modules do not yet implement the full set of features required for future operations. Some software overlaps several modules, and some modules are not relevant for the RHSC.

2.1. Operations Management System

RH Procedures will be stored in an Operations Management System (named ODS at JET). The RHSC developed a web server component, called the RH Task Browser (RHTB), for accessing the task information through any web browser. The tasks are stored in a backend Java server running a PostgreSQL database.



SMA Replacement on UPL Maintenance in progress..

1.2.14	CD	Lock Mascot Z direction	5
2		Remove SMA	
2.1		Use Bolt Runner [unscrew]	
2.1.1	MO	Place Bolt Runner in position for Captive Bolt	20
2.1.2	MO	Unscrew Captive Bolt	10
2.1.3	DRO	Check if Captive Bolt is Ur	10
2.2		Place Bolt Runner on Tool Box	
2.2.1	MO	Place Bolt Runner on Tool Tray	20

See the image from the Virtual Reality on the left to see how to position the Bolt Runner

Check pre-conditions

✓ Is there a bolt runner available on the tool tray?

Figure 2.1 RHTB with procedure to replace Steering Mirror Assembly component on Upper Port Launcher plant

The RHTB allows RH procedures to be input in a programmatic way, using branching and sub-functions, steps with situational pictures and descripts, user rights management, and procedure execution by a team of operators. The operators have different roles and can work together on this system to successfully complete procedures and annotate steps along the way.

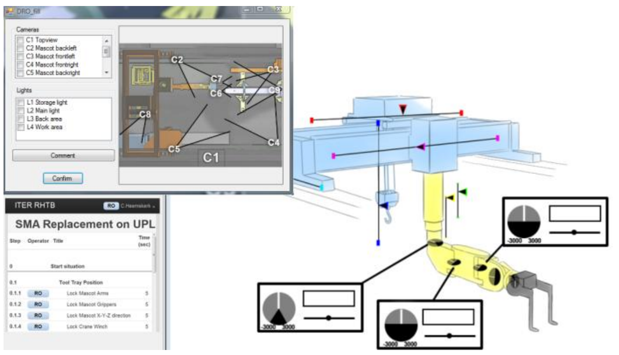
A structured language for defining and describing RH steps remains to be developed.

2.2. Virtual Reality

In the RHCS context, VR is merely used as an aid to simulate the current state in a scene and to help the operator easily monitor situations which cannot be directly viewed by real cameras.

2.3. Physical simulator

The Virtual Slave Simulator (VSS) is the RHSC implementation of the Nvidia PhysX rigid body simulator with contact interaction. The software is soft-realtime and runs on a Windows platform with refresh rates up to 1kHz. For non-trivial scenes, the update rate is between 250Hz and 333Hz. This is still fast enough to transparently display slave manipulator forces to the master operator.



SMA Replacement on UPL

Step	Operator	Title	Time (sec)
0		Start situation	
0.1		Tool Tray Position	
0.1.1	RO	Lock Mascot Arms	5
0.1.2	RO	Lock Mascot Grippers	5
0.1.3	RO	Lock Mascot X-Y-Z direction	5
0.1.4	RO	Lock Crane Winch	5

Figure 2.2 HMI with identification of manipulator axes and positions, camera and light locations

2.4. Command and Control

This is a category of user interface software components that incorporates generic user interfacing to the VR and other systems, as well as specific interfacing to RH equipment and tooling. The latter are often in form of a Human Machine Interface (HMI, Figure 2.2) and incorporate commands specific to the start, operation and shutdown of these systems.

2.5. Modules irrelevant to RHSC

The RHSC lacks a hardware backend, so it is not yet possible to implement a complete remote viewing system with sensors, image processing, monitors to the operator and remote diagnostics. However, research into a synthetic viewing system is being undertaken [8], employing cameras to geometrically calibrate a virtual scene to a real hardware setup [2].

2.6. RH supervisor role

One item is added to Figure 1.1, being the RH supervisor role [7] – an operator interface and RHCS component that allow setting the operational theatre. Its function is to allocate resources (RH interfaces, RH equipment, network capacity, and operators) to the task at hand and changing task states.

3. Middleware synopsis

Functional requirements based on data analysis and types of middleware are presented below. The view of ITER RH is presented and taken into account to identify a few available middleware frameworks. These are pilot tested. Furthermore, as network performance will be limiting performance of any middleware, it is evaluated whether conventional Ethernet is fast enough for time synchronization and soft real-time control.

3.1. Requirements from data perspective

The combination of data requirements in the various RH networks makes for a challenging setting for middleware. The goal is to use the same middleware across all networks.

Real time data, such as joint position updates, requires small data packets to be sent with little latency. Typical messages are in the order of 40 bytes per RH equipment and to be sent with <0.2ms end-to-end but preferably <0.1ms round-trip delay at a high rate (1kHz). A small loss of data is acceptable only if very infrequent.

Video data requires multiple, continuous streams with, for HD-video, >1Mbit/s bandwidth per channel with <50ms end-to-end latency. Frame drops and data loss are acceptable to about 10% before visual artifacts start to hinder operators.

Audio, if used, is similar to video with a factor 20 lower bandwidth for one channel. Time stamping is important to keep the audio in sync with video.

Text, messages, state changes and other data is much lower in bandwidth and does not have real-time or bandwidth requirements, but normally do require acknowledgement of correct reception. Database transactions will not be too large but do require fast transmission and fast processing, as other clients might depend on the correctness in time of some of the data.

File/model data exchange requires larger amounts of data to be transmitted in a short interval, with integrity guarantees, but no real-time or firm bandwidth constraints. As it is likely that multiple module (instances) will individually load models at the same time the concurrent capacity needed will be quite large.

From these conflicting requirements is derived that at the least a Quality-of-Service enabled network will be necessary. Combining streaming and file access is common practice although tailoring is required for critical data. However, providing real-time guarantees with low latency and jitter is much more difficult [9].

3.2. ITER RH originating requirements

The RHCS should align to ITER RH middleware requirements. That shall [10]:

- Be able to support synchronous remote method invocation (Sync RMI),
- Be able to support asynchronous remote method invocation with call-back (Async RMI),
- Use a neutral language to define remote object public interfaces (Neutral Language),
- Have no platform, language, network, or application dependency (Independency).

3.3. Types of middleware

In the literature four main types of middleware can be found [11]. The transactional type is geared towards database transactions, with consistency guaranteed, having as drawback that clients could have to wait on other client transactions. Remote Procedure Calls (RPCs) are transparent and supported on multiple platforms, but scalability is limited as there is no support for asynchronous communication, load balancing etcetera. Message Oriented Middleware (MOM) offers a mailbox abstraction focusing on asynchronous transmission. No confirmation of delivery is implemented but other solutions like QoS [12] improve reliability. There's no standardization. Object Oriented Middleware (OOM) evolved from RPCs by adding inheritance, references and exceptions. Transport transformation is automatic, synchronous and asynchronous communication is supported. Scalability is somewhat limited by overhead in locating the implementation and methods by remote objects. The latter type is the most widely used and most evolved, and integrates all functionality necessary.

3.4. Pilot testing available OOMs

On the Windows 7x64SP1 working environment of the RHSC, a number of readily available options were pilot tested for integration. Setting up build environments and producing simple working applications on localhost took for all options a few days.

CORBA (Common Object Request Broker Architecture) is the classical OOM and a widely recognized standard. However, many different versions and many different implementations exist, none of which implements the full scope. In JacORB version 2.3, an open source CORBA implementation, a trivial "Hello world"-like client/server application was programmed. It was found to require quite some environment configuration to get it working.

ICE version 3.4 (Internet Communication Engine, an OOM) by ZeroC has been implemented in the VSS application. One-way latency in data transfer measured over localhost, i.e. both communicating software components are located on one PC, increased from 0.1ms to 0.16ms. Additionally ZeroC has IceStorm available, a DDS (Data Distribution Service, which is a form of MOM).

OpenSplice DDS & OpenSplice RMI (Remote Method Invocation) are respectively a MOM and OOM offered by PrismTech. Version 6 of DDS is recommended by a study for ITER RHCS [10] and was implemented here. In contrast to IceStorm it has QoS available, enabling native support for real-time applications. It performs very fast over localhost, on average some 25μs round-trip, although some spikes do occur. This is much faster than sending an ICMP over localhost (around 250μs), or sending an UDP datagram over localhost (around 50 μs) suggesting that OpenSplice bypasses the network stack for localhost connections.

Unfortunately, implementation of OpenSplice RMI was found to be hard, support from PrismTech turned out to be slow and limited. Manuals are concise and community support is barely available. It is a new product for OpenSplice and seems immature. An ad-hoc RMI method via DDS was not implemented at this time.

EPICS is the middleware of choice for ITER CODAC. However it does not seem to be suited to robotics [13] so this option was not tried.

3.5. Real time control

Closed-loop control needs accurate time synchronization to be an order of magnitude better than the smallest time slice encountered (about 100μs). Network Time Protocol offers accuracy down to one millisecond. NTP is present and enabled in all major operating systems, however in Windows it is a simplified version which does not enable synchronization accuracy better than 2 seconds [14].

As an alternative, the Precision Time Protocol (PTP) is an IEEE standard and allows sub-microsecond accuracy but is a bit tedious as it requires specific network switches, and functions best with specific endpoint Ethernet hardware.

3.6. Ethernet latency

On a local network with Dell i5 Windows 7 PCs with 1Gbit/s cards, all connected via a Foundry FLS 648 switch, Ethernet latency has been estimated by sending ICMP packets.

From: To:	Linux, 32bit Ubuntu 12.04	Windows 7 SP1 64bit
Linux	130µs	860 µs
Windows	600µs	1350µs
Localhost	11µs	250µs

Table 1 Round-trip delay timing (averaged) on local network

Latencies with only a crossover-cable were similar, but more consistent. From Table 1 it seems the Windows endpoints contribute significantly to the overall latency, and a multiple-PC network environment contributes to jitter and spikes. Our test was not extensive enough to report actual jitter values.

Results indicate that Windows should be avoided for soft real-time applications over Ethernet. Other actors on the network will create further performance degradation, both in throughput and in latencies. Therefore, it is important to control the behavior on the network, either by programmatically enforcing correct behavior on the application side, or enforcing application access and behavior on the network switches.

For future work it is recommended to perform further performance testing the middleware options in a representative environment.

4. Conclusions and future steps

The ITER RHCS requires a mixture of Object Oriented Middleware (OOM) and Data Distribution Service (DDS) like middleware architectures. Both the OOM and DDS solutions should preferably be acquired from the same supplier to insure interoperability and support. Candidate suppliers that can deliver both include PrismTech OpenSplice and ZeroC ICE, however the research presented here is not complete to warrant a preferred option yet. The major advantage of PrismTech is native QoS support, with a disadvantage that their RMI product seems immature.

Additionally a small management component overhead is suggested which serves to monitor network usage and performance, and can act as the ultimate escalation level that can be called by individual actors (when require more capacity, lower latency). This component can balance requests and act accordingly.

For the RHSC at DIFFER, the ad-hoc UDP datagram method will be extended to basic real-time functionalities [15] and system time synchronization will be attempted. Prototyping will proceed both with Ice and OpenSplice middleware to evaluate implementation in these environments with real-life applications, and network control will be implemented.

Acknowledgments

This work, supported by NWO, ITER-NL and the European Communities under the contract of the Association EURATOM / FOM, was carried out within the framework of the European Fusion Programme. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization

References

[1] A.Tesini et al., ITER Remote Maintenance System (IRMS) lifecycle management, Fusion Engineering and Design, Volume 86, Issues 9–11, October 2011, Pages 2113-2116

[2] J.F. Koning et al., Analysis of ITER upper port plug remote handling maintenance scenarios, Fusion Engineering and Design, Volume 87, Issues 5–6, August 2012, Pages 515-519

[3] C.J.M. Heemskerk et al.., Verifying elementary ITER maintenance actions with the MS2 benchmark product, Fusion Engineering and Design, Volume 86, Issues 9–11, October 2011, Pages 2064-2066

[4] J. van Oosterhout et al., Haptic Shared Control improves Hot Cell Remote Handling despite controller inaccuracies, this conference

[5] H. Boessenkool et al., Challenges in human-in-the-loop tele-operated maintenance at ITER, this conference

[6] G.Y.R. Schropp, On the Influence of Visual Feedback on Human Task Performance in Remote Handling, Fusion Engineering and Design, Volume 87, Issues 5–6, August 2012, Pages 808-812

[7] D.T. Hamilton et al., An integrated architecture for the ITER RH control system, Fusion Engineering and Design, Volume 87, Issue 9, September 2012, Pages 1611-1615

[8] C.J.M. Heemskerk et al., Introducing artificial depth cues to improve task performance in ITER maintenance actions, this conference

[9] T. Dumitras et al., A study of unpredictability in fault-tolerant middleware, Computer Networks, article in press

[10] P. Soetens et al., Middleware, ITER_D_7554YJ v1.0

[11] H. Pinus, Middleware: Past and Present a Comparison, http://userpages.umbc.edu/~dgorin1/451/middleware/middleware.pdf

[12] ITU, G.1000 : Communications Quality of Service: A framework and definitions, http://www.itu.int/rec/T-REC-G.1000-200111-I/en

[13] P. Soetens, RH Operating Framework, ITER_D_6URDH4 v1.0

[14] Support boundary to configure the Windows Time service for high accuracy environments, http://support.microsoft.com/kb/939322

[15] Mahajan K., UDP Unicast Test, ITER_D_46LRZ4 v1.1