Training Benefits of Java-Based Part Task Trainers: Lessons Learned from MH-60S/MH-60R Helicopter Training

Robert A. Richards
Stottler Henke Associates, Inc. (SHAI)
San Mateo, CA 94404, U.S.A.
IEEEAero2008.R.RichardsPhD@Neverbox.com

Jeremy Ludwig
Stottler Henke Associates, Inc. (SHAI)
San Mateo, CA 94404, U.S.A.

Abstract—The US Navy's PMA-205 in conjunction with Stottler Henke has re-designed and re-implemented the partial task trainer (PTT) for the Common Cockpit of the new MH-60S and MH 60R helicopters. The tool, called the Operator Machine Interface Assistant (OMIA), is an expandable, easily modifiable low-cost PC-hosted desktop crew trainer. OMIA-JAVA is currently in use for training at HSC-3 and HSM-41 at NAS North Island and HSC-2 at Norfolk NAS; and is continuing to evolve to match the changing helicopters.

OMIA is now implemented in Java and can be run on NMCI (Navy/Marine Corps Intranet) computers. This new implementation allows the software to have all the benefits of a 'portableapp', and a web-based application. OMIA-Java can be delivered as a Java web-start application or users can download a simple zip file. The contents of the zip file are extracted and then the product can be run directly on the user's computer without any further installation or administrator privileges (aka a 'portableapp'). This allows the software to run on any Windows machine that already has a Java virtual machine installed, such as standard NMCI computers. This also allows OMIA-Java to run from external USB "thumb drives".

Formerly, OMIA was used in dedicated classrooms and on personal machines. In both cases, administrator access was required to install OMIA. However, most aviators have access to NMCI computers everywhere they go so this is the most convenient platform to deliver to. Many training solutions have utilized a web-based solution to provide ubiquitous access, but many applications suffer performance problems because they're running over the web. This paper demonstrates the benefits of developing training or other software in the same manner as OMIA is developed so that it is easily available and efficient; as well as describes some of the most recent advances to OMIA .[1] [2]

Another major benefit of recasting OMIA as a Java program is the relative ease of creating and modifying user interfaces to match the evolving helicopter interface. Code reuse was also an influencing factor as the latest version of OMIA-JAVA needed to be client/server based and Stottler Henke already had an extensible body of code to provide this functionality as part of the SimVentive product. So in less than six months a new version of OMIA-Java was created that matches the current interfaces of the helicopters while supporting all of the original OMIA-functionality that took several years to develop.

[1] 1-4244-1488-1/08/$25.00 ©2008 IEEE.

[2] IEEEAC paper #1064, Version 1, Updated January 15, 2008

TABLE OF CONTENTS

1. INTRODUCTION

The US Navy has introduced two new helicopters, the MH-60S and MH-60R, see Figure 1. Both of these helicopters utilize Lockheed-Martin's Common Cockpit design. The Common Cockpit includes all the flight and mission instrumentation in both of the helicopters and enables both the pilot and co-pilot to share workload through dual flight and mission instrumentation, see Figure 2. As can be seen in Figure 2 the pilot and copilot each have two LCD screens, one of which is the Mission Display (MD) and the other is the Flight Display (FD). The pilots interact with these displays primarily through a set of bezel keys around each display and a keypad located in the center console. This keypad contains a set of fixed function keys (FFK), a set of context-dependent programmable keys (PK), and a small joystick known as the "hook". For more than seven years the US Navy's PMA-205 in conjunction with Stottler Henke has developed/deployed/updated a flexible, low-cost PC-hosted crew trainer for the Navy's new MH-60S (Sierra) and MH-60R (Romeo) helicopters called the *Operator Machine Interface Assistant* (OMIA-JAVA). Until 2007 this software was written mostly in C++, which had been the best language for the challenges. A description of the C++ version is provided by in [1] & [2]. However, due to various circumstances OMIA was converted to Java in 2007.

Figure 1. MH-60S

OMIA-JAVA has core functionality that may be enhanced via optional software and hardware. The core of OMIA-JAVA provides a partial-task trainer (PTT) of the helicopter software and hardware. The trainer includes the flight and mission displays as well as the programmable & fixed function keypads, the hook, and the RCU (Radio Control Unit), CMP (Control Monitor Panel), and CCU (Cockpit Control Unit) panels.

OMIA-JAVA is currently in use by HSC-3 and HSM-41 at NAS North Island and HSC-2 at NAS Norfolk, as well as being available to anyone in the Navy with a PC. This paper describes why it became beneficial to convert OMIA-JAVA to Java in order to increase its capabilities, and flexibility. There were 4 main driving factors that lead to the decision to convert to Java:

1) Continual helicopter software updates that require corresponding updates to OMIA-JAVA,

2) Java version of DiSTI libraries that are used for parts of the user interface and cockpit,

3) NMCI compatibility,

4) Need for a client server version of OMIA-JAVA to support multiple seats and multiple helicopters working together.

Continual helicopter software updates: The flexible design for evolving requirements is necessary because the Common Cockpit has and continues to evolve. Even though the MH-60S and MH-60R both use the Common Cockpit, the helicopters have different capabilities and missions, thus many operations are different on the two platforms. However; a programmable keyset (PK) supports the differences. In addition, the software for the two platforms is not at the same version. The Navy supports these differences in OMIA-JAVA. Since this process will be continuing for years it is always best to have the software in the most flexible language for this task. Advances in the Java language and Java tools have now made it a better choice for rapid modification.

Figure 2. Common Cockpit

OMIA-JAVA has been able and must continue to work with and control Microsoft™ Flight Simulator when it is available, to use COTS and/or custom hardware when attached, and to still function as a complete standalone application. In addition, the C++ version of OMIA-JAVA software was the software component of the MH-60S Mission Avionics Systems Trainer (MAST), this has been replaced by the Java version

Java version of DiSTI libraries: The option to go to Java was facilitated when DiSTI released a Java version of their tools that is used in part of OMIA-JAVA. Without this option, the rest of OMIA-JAVA could have been converted to Java and some of its benefits could have been realized but it would still not be able to run on NMCI machines.

NMCI compatibility: One of OMIA-Java's goals has always been its availability on as many computers as possible both on land and at sea. Thus even though it can be enhanced by optional hardware and MS Flight Simulator, a very functional standalone version has always been available. The default computer configuration in the Navy is referred to as NMCI (Navy/Marine Corps Intranet). These have restricted access and certain aspects of the C++ version of OMIA-JAVA could not be used easily on NMCI machines, including DiSTI's libraries. But once Java versions became available, NMCI compatibility could be provided.

Need for a client server version: Besides OMIA-JAVA keeping up with the helicopters' changes, OMIA-JAVA is

constantly being enhanced. One of the recent enhancements is the client-server design change so that multiple users of OMIA-JAVA can interoperate. That is, OMIA-JAVA can be run by multiple users, so that a set of users can match different seats in one helicopter, and multiple helicopters can also be handled so everyone is playing in the same world. The change was simplified because Stottler Henke already had a general client-server capability built into one of its Java based tools. This was leveraged in the OMIA-JAVA Java version..

2. OMIA-JAVA CONFIGURATIONS

As mentioned above, OMIA-Java has many configurations. The core OMIA-Java is a standalone Java program that operates under any standard Windows 2000 or Windows XP computer that includes a Java Runtime Environment (JRE); this includes NMCI computers. The standalone OMIA-Java provides an introduction to the Common Cockpit, including the Mission Display, the Flight Display, the Center Console's Fixed Function and Programmable Keys, and the CMP, RCU & CCU units. The CCU, alternately known as the Head-Mounted Display Control Unit, is currently only available on the Sierra so this panel is only available on the Sierra.

A major benefit of the standalone core OMIA-Java product that the Navy requires is that it requires no external licensing, and therefore it can be distributed freely to anyone in the US Navy via CD or via the Web.

The core OMIA-Java can be used to teach both the Sierra and Romeo versions of the helicopter. A different executable is created for each configuration (presently one for the MH-60S and two for the MH-60R since the R has a back seat Sensor Operator). In addition, the user can also run in standalone mode (the default) or in network configuration. In a network configuration, in which one operator can be the pilot and another operator can be the co-pilot or sensor operator. In this scenario, both operators will see the same world, including changes made by each other. To do this, you have to state whether you are the server or the client. The first person to start OMIA-Java has to be the server so that the second person can designate himself as the client and the program will search for a server for them to join on the network. To start in network (client/server) mode the OMIA-Java executable is started with the *–multi* option.

If optional hardware is attached OMIA-Java and Windows discovers it and works correctly with it automatically. The simplest example is multiple monitors, by attaching two displays the Mission Display, shown in Figure 3 and the Flight Display, shown in Figure 4, can be displayed on separate monitors, with one of the monitors also displaying the Center Console. Another option is to have one or more of the screens made a touch screen as is done in the Mission Avionics System Trainer (MAST), described below, in which the bezel keys on the flight display and mission display are operated using finger pushes on a touch screen to more accurately emulate the ergonomics of the actual helicopter. Of course, the third screen containing the Center Console panels could also be a touch screen so the user could push the buttons in a more similar way as is done in the aircraft instead of using the mouse.

Figure 3. Mission Display with Menu Displayed

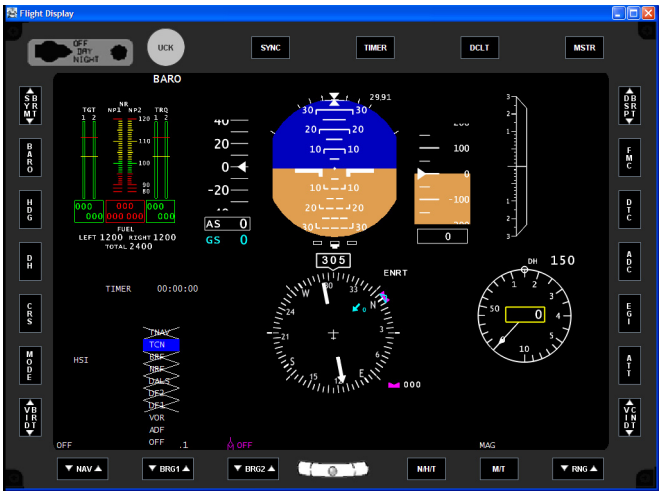

Figure 4. OMIA-JAVA Flight Display

At this time, software additions for OMIA-Java consist only of Microsoft Flight Simulator. Every time OMIA-Java starts it checks to see if Microsoft Flight Simulator is already running. If it is running, OMIA-Java attaches itself to Microsoft Flight Simulator and then gets its position, speed and other flight information from Microsoft Flight Simulator. In this configuration, the user could have the external view being completely generated by Microsoft Flight Simulator, and the Flight Display, Mission Display

and all of the other panels still being used from the core OMIA-Java. However; any other information such as ground speed, latitude/longitude location, or motion is all being read in from Microsoft Flight Simulator. This is very beneficial if you wish to fly or see the terrain while navigating a search and rescue pattern. As one navigates, the helicopter may be guided along the search and rescue pattern on the Mission Display, and as search and rescue points are reached or captured the pattern will update appropriately. When using Flight Simulator, other hardware can be used if desired. One can plug in a joystick, or a head mounted display with head tracking can be added to improve the means for emulating the full field of view. Both options are handled seamlessly by OMIA-Java and Microsoft Flight Simulator.

Flying can be performed solely using a joystick; or a joystick and a separate control for the collective, or COTS pedals could be added. Microsoft Flight Simulator also provides an automatic pilot, as well as the Slew Mode, so one can move the helicopter without having to concentrate on the flying. Since the flying performance will not actually be realistic for an MH-60S or MH-60R helicopter, it is normally better to use the Slew Mode. This feature allows for moving the helicopter in any desired direction without having to fly a helicopter whose characteristics are not going to match the exact characteristics of the actual helicopter. More information on the details of interfacing with Microsoft Flight Simulator is provided in [1].

A two seat simulator, the Mission Avionics Systems Trainer (MAST) essentially combines all the above mentioned capabilities that can be added to the core OMIA-Java. The MAST is further described in a section below.

3. OMIA AS PORTABLE & WEB APP

A portable application, or 'portableapp' is a software program that does not require any kind of formal installation onto a computer's permanent storage device to be executed, and can be stored on a removable storage device such as a CD-ROM, USB flash drive, flash card, etc., this enables it to be used on multiple computers. The portableapp reads its configuration files from the same storage location as the software program files.

Most software for Microsoft Windows is not portable, because they use the Windows registry, etc. That is, if one installs an application that is 'installed' in a folder and one copies the folder to another computer, usually the software will not run on the second computer (where in contrast a portableapp would).

By making OMIA-Java a portable application, the Navy receives many benefits. (First a caveat, since OMIA is written in Java in some regards it is NOT completely portable because there needs to be a Java Runtime Environment (JRE) on the computer that OMIA runs on. The JRE is freely available and many machines already include it, including all NMCI machines. However, the default JRE is not portable. However, there is work in making a portableapp JRE. For the rest of this discuss OMIA Java will be referred to as a portableapp.)

A major advantage that has already proved itself very valuable is the ease of distribution and 'installation'. Since a portableapp does not require formal installation it can be used directly from the distribution media as long as the media is writable. That is, USB drives can be plugged in and the OMIA-Java software can be run directly from the USB drive. This is not the case for a CD disk, since the disk is read only. However, in both cases the OMIA directory can be simply copied to anywhere on the computer and then run from the location on the computer. This has made distribution to training classrooms trivial compared to the previous C++ OMIA that required an installer. Automating the installation on a large number of computers was difficult, resulting in a delay in the distribution of OMIA C++ updates. The new process is simply a directory copy. In addition, the OMIA-Java directory can simply be placed on a network drive that all the computers can see and run from the network drive. Running from a network drive has resulted in slower start-up times but once loaded the software's responsiveness is about the same as running locally.

NMCI compatibility, as already mentioned, is a huge advantage provided by OMIA-Java being a portableapp. Previously, users would have to go to a lab when they wanted to utilize OMIA even though most have an NMCI computer at their desk. There is a huge convenience factor in making OMIA-Java available at a user's desk.

Portableapps provide essentially all the advantages of WEB-based applications, plus something that a WEB-based applications don't or can not. Since a portableapp runs locally it is (almost always) much faster than a WEB-based application. In addition, the portableapp (in installed locally or on a USB drive) is available even when the internet is not.

Another advantage of OMIA being written in Java is that it could be run as a Java web-start application. That is, not only is OMIA-Java a portableapp, it is also a small task to make it a WEB-based application. Due to the many advantages of being a portableapp, there has been no need to make it a web-start application.

A final advantage of the new OMIA-Java is that if it needs to be ported to another platform (e.g., Linux) , it is written in Java, a language that allows for the easiest portable across different operating systems. Presently there is no desire on the part of the Navy to have OMIA run on any other platform.

4. MISSION AVIONICS SYSTEMS TRAINER (MAST)

The MAST, as shown in Figure 5, includes actual hardware in the Center Console that is exact aircraft hardware or a very close facsimile. The MAST hardware was procured by the Navy from JF Taylor, Inc. One can actually push physical buttons, change actual knob positions, feel feedback, open covers, etc. There are two seats, the pilot and co-pilot. Each seat has two screens just as in the helicopter, one for the Flight Display and one for the Mission Display. There are individual screens for the pilot and co-pilot showing the outside view, generated by Microsoft Flight Simulator. There is a simple cyclic in the MAST and the screens for the Flight Display and Mission Display are actual touch screens. Another feature of the Center Console hardware is the actual hook hardware, used to control the cursor in the Mission Display.

Figure 5. Mission Avionics System Trainer (MAST)

The MAST is a medium resolution trainer driven completely by OMIA-Java software and MS FS software. Again, there is only one version of OMIA-Java, it can work with or without MAST hardware. The MAST can be used for many different types of operations, including coordinated operations because, as described above, the two seats can be used independently; or in conjunction (client/server mode) so the pilot and co-pilot are flying the same mission. The MAST has been in use for a couple of years at HSC-3 at NAS North Island and another MAST is available at HSC-2 at NAS Norfolk. They are mainly used for Sierra training; however, since they are being completely driven by OMIA-Java software, they can be quickly reconfigured as Romeo stations via restarting the programs in Romeo mode.

5. ENHANCEMENTS

OMIA-Java continues the earlier history of OMIA, where each version not only continued to match the evolving helicopter software, but functionality is expanded. OMIA-Java has added a FLIR (Forward Looking Infrared) The FLIR is an external hardware unit, see Figure 6 that provides an infrared view of the exterior on the mission display.

Figure 6. FLIR Hardware

The FLIR user mainly controls the FLIR operations via a Hand-Control Unit (HCU), as shown in Figure 7. The Navy has developed a portable HCU that uses the actual helicopters HCU, but connects to a USB controller with a USB connector. This portable training HCU has been interfaced to OMIA-Java.

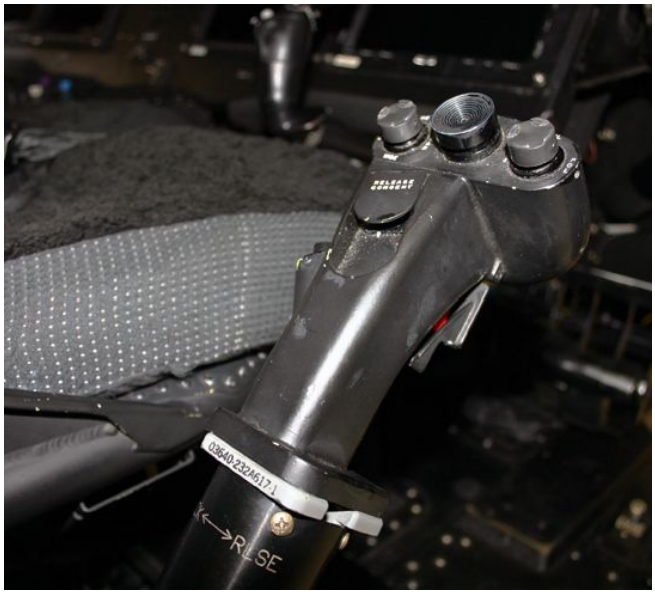

Figure 7. FLIR Hand Control Unit (HCU)

OMIA-Java reacts the same way to the HCU hardware as it does to the presence/absence of other hardware units; when OMIA-Java starts up it detects if the FLIR HCU hardware is attached. If it is attached, the software will read input from it, if it is not detected a software equivalent is provided.

An example of a FLIR image, with the MH-60 overlay, as shown on a Mission Display is shown in Figure 8. The generation of FLIR images is a difficult task in real-time. Usually FLIR simulators are very expensive units incorporated into multi-million dollar simulators. For OMIA a simpler solution is created to provide a high level of learning benefit without the cost. Actually there are two solutions, for maximum flexibility for the Navy.

Figure 8. FLIR Image

The simpler solution uses a 2D FLIR image and runs completely in the stand-alone OMIA-Java. Much of the learning related to FLIR concerns the operation of the FLIR menus and other operations that are part of the overlay. So the combination of the actual FLIR HCU and the overlay menus and other functions, a great deal of learning is facilitated.

A more complete solution with the ability to fly in a three-dimensional FLIR environment is created via MS Flight Simulator. A small FLIR world has been created in MS Flight Simulator, that includes an island with bunkers and tanks, as well as some ships offshore. A simulated FLIR image is shown in Figure 9; this solution allows the user to experience most aspects of utilizing FLIR before using more expensive simulator or real flight time.

Figure 9. Simulated FLIR Image

6. CONCLUSION

The complexity and number of the sensors under control of the crew on the MH-60S and MH-60R helicopters pose a difficult training task for the Navy. To meet this challenge the US Navy's PMA-205 in conjunction with Stottler Henke and various hardware vendors has developed and deployed OMIA, a flexible, low-cost PC-hosted desktop crew trainer. OMIA has evolved with the changing helicopter software, in addition it has become an ever more functional trainer with each iteration. The latest version of OMIA was refactored and converted from C++ to Java. This latest version opens up OMIA to run on just about any Navy PC since it functions on the NMCI PCs which are the default computer provided to Navy personnel. OMIA is available for anytime training on both land and at sea. To learn more regarding the past, present and future of OMIA, please visit the project web page at www.StottlerHenke.com/OMIA.

REFERENCES

[1] Richards, R., J. Ludwig (2007) "PC Rapid Modification Tool for Aircraft Experimentation & Training for the MH-60S/MH-60R Helicopters", 2007 IEEE Aerospace Conference Proceedings. 2007 IEEE Aerospace Conference Proceedings. Big Sky, Montana, March 4-9, 2007.

[2] Ludwig, J. (2006). Comparing helicopter interfaces with CogTool. 7th International Conference on Cognitive Modeling, Trieste, Italy.

BIOGRAPHY

Robert Richards, Ph.D. is the Principal Scientist and Manager of Stottler Henke's Navy helicopter training contract, OMIA-JAVA. OMIA-JAVA is a PC-based desktop training system that teaches crewmembers the Navy's new MH-60R and MH-60S helicopter. Dr. Richards has taken the project from a Research and Development SBIR project to a deployed training tool that has been awarded a $4.1 million IDIQ contract. Dr. Richards received his Ph.D. from Stanford University in mechanical engineering with an emphasis on machine learning and artificial intelligence. Dr. Richards is managing and has managed multiple projects for both commercial and government clients, including various intelligent-tutoring-system-based training projects. He is the principle investigator for VERTICAL, a Navy project to develop an innovative analytic test tool that can be used to support vertical takeoff and Visual Landing Aid analysis and testing. He was also the PI for INCOT, an Air Force project that developed automated tools for network layout. These projects exemplify his wide range of research and application area interests, including: training system development; applying automation and artificial intelligence techniques; and decision support tool development for life-critical situations. Dr. Richards has publications in all these areas.

Jeremy Ludwig is the technical lead on the OMIA-JAVA project. He holds a Master's Degree in Computer Science, with a concentration in Intelligent Systems, from the University of Pittsburgh and a Bachelor of Science Degree in Computer Science with minors in Psychology and Philosophy from Iowa State University. Other projects he has been involved with recently include the SimBionic behavior modeling framework and the SimVentive instructional game toolkit.