MH60S/R Helicopter Multi-Platform & Web-Based Crew Trainer with FLIR

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

Robert A. Richards
Stottler Henke Associates, Inc. (SHAI)
San Mateo, CA 94404, U.S.A.
OMIA@StottlerHenke.com

Abstract—The US Navy's PMA-205, in conjunction with the training and simulation industry, has developed and deployed OMIA: a flexible, multi-platform, Web-based crew trainer for the Navy's new MH-60S and MH-60R helicopters. [12]OMIA is currently in use by HSC-2, HSC-3 and HSM-41 and is available to all crewmembers throughout the Navy. To maximize access to the trainer and to make deployment easier, OMIA is written in Java, and deployed both as a portable application and as a web application. A portable application does not require any special user rights to install; a web application is run over the internet from a browser. OMIA includes a simulation of the MH-60S/R Common Cockpit, including a FLIR capability that includes using an actual hardware FLIR hand-control unit when attached through USB. OMIA has been designed and implemented to be flexible to changing Navy needs, a design aspect which proved itself again in 2009 when OMIA's FLIR was converted for use in the fixed wing EP-3E aircraft. OMIA illustrates solutions to three critical issues in developing low-cost training software for aircraft: quickly responding to the ever-changing demands, re-use of interface components across multiple aircraft, and providing training that can be accessed where and when it is needed.

TABLE OF CONTENTS

1. INTRODUCTION 1
2. OMIA PART-TASK TRAINER 1
3. FLIR ENHANCEMENTS 5
4. EP-3E FLIR TRAINER 6
5. CONCLUSION 7
REFERENCES 7
BIOGRAPHY 7

1. INTRODUCTION

The US Navy has introduced two new helicopters, the MH-60S and MH-60R. 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.(Figure 1). As can be seen in Figure 1 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 eight years the US Navy's PMA-205, in conjunction with Stottler Henke, has developed, deployed, and 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).

Figure 1. MH-60 Common Cockpit

2. OMIA PART-TASK TRAINER

The core OMIA (Figure 2) is a standalone Java program that operates under any standard Windows (XP, Vista, and 7) or Linux computer that includes a Java Runtime Environment (JRE); this includes Navy/Marine Corps Intranet (NMCI) computers. The standalone OMIA provides an introduction to the Common Cockpit, including the Mission Display (Figure 3), the Flight Display (Figure 4), the Center Console's Fixed Function and Programmable Keys (Figure 5), and several helicopter control panels. A major benefit of the standalone core OMIA 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 compact disc or the Web. However, the core system also supports a

[1] 978-1-4244-3888-4/10/$25.00 ©2010 IEEE
[2] IEEEAC paper#1067, Version 2, Updated 2009:11:17

number of optional extensions to meet additional training needs.

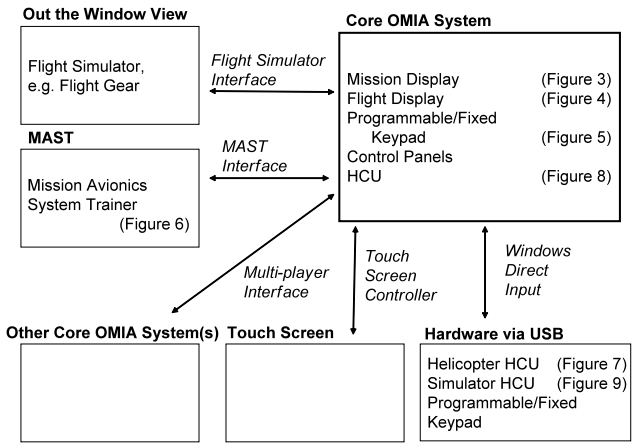

Figure 2. The Core OMIA System and Optional Extensions

The core OMIA can be used to teach both the Sierra and Romeo versions of the helicopter. A different executable is created for each configuration. Presently there are two for the MH-60S (armed and non-armed) and two for the MH-60R (pilot and sensor operator). In addition, the user can also run in standalone mode (the default) or in network configuration. In a network configuration, 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 has to be the server so that the second person can designate himself or herself as the client; the program will search for a server for them to join on the network. To start in network (client/server) mode, the OMIA executable is started with the *–multi* option.

Figure 3. OMIA Mission Display with Menu Visible

If optional hardware is attached, OMIA 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 that the user could push the buttons in a way more similar to how it is done in the aircraft instead of using the mouse.

A software addition for OMIA is connecting it to a flight simulator. Every time OMIA starts it checks to see if a compatible flight simulator (e.g. Microsoft Flight Simulator, Flight Gear) is already running. If it is running, OMIA attaches itself to the flight simulator and then gets its position, speed and other flight information from it.

Figure 4. OMIA Flight Display

In this configuration, the user could have the external view be completely generated by the flight simulator, and the Flight Display, Mission Display and all of the other panels still being used from the core OMIA. However, any other information such as ground speed, latitude/longitude location, or motion is all being read in from the 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 a flight simulator, other hardware can be used if desired. One can plug in a joystick, or a head mounted display with head tracking may be added to improve the means for emulating the full field of view.

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 do not match the exact characteristics of the actual helicopter. More information on the details of interfacing with Microsoft Flight Simulator is provided in [1].

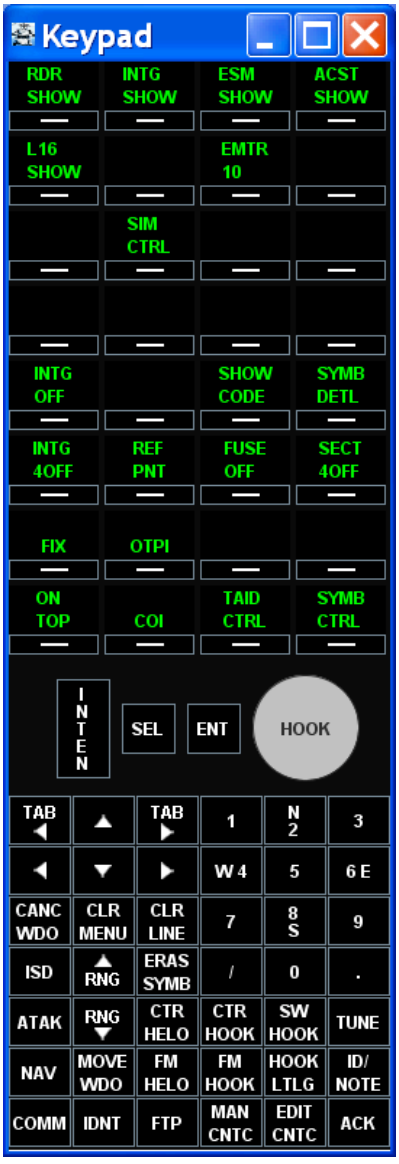

Figure 5. Programmable/Fixed Function Keysets

Converting from C++ to 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 in [1] & [2]. However, due to various circumstances OMIA was converted to Java in 2007. This work is described in more detail in [3].

OMIA has core functionality that may be enhanced via optional software and hardware. The core of OMIA 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.

There were five main driving factors that led to the decision to convert to Java. First, the flexible design for evolving requirements is necessary because the Common Cockpit 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 always at the same version. The Navy supports these differences in OMIA. 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 tools have now made Java a better choice for rapid modification.

OMIA has been able to, 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 software was the software component of the MH-60S Mission Avionics Systems Trainer (MAST); this has been replaced by the Java version

Second, the option to go to Java was facilitated when The DiSTI Corporation (www.simulation.com) released a Java version of their tools that is used in part of OMIA. Without this option, the rest of OMIA 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 Navy/Marine Corps Intranet (NMCI) machines.

Third, one of OMIA'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 could not be used easily on NMCI machines, including DiSTI's libraries. But once Java versions became available, NMCI compatibility could be provided as the required Java Runtime Environment is already installed on the NMCI machines.

Fourth, besides keeping up with the helicopters' changes, OMIA is constantly being enhanced. One of the recent enhancements is the client-server design change, so that multiple users of OMIA can interoperate. That is, OMIA 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 version.

The fifth advantage of the new OMIA is that it is easily ported to other platforms (e.g., Linux). The Navy requested a Linux version of OMIA in 2009 as part of an acoustic systems trainer enhancement. The fact that OMIA was written in Java allowed us to take advantage of this unexpected opportunity.

Mission Avionics System Trainer (MAST)

The MAST, as shown in Figure 6, includes actual hardware in the Center Console that is exact aircraft hardware or a very close facsimile of it. 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 6. Mission Avionics System Trainer (MAST)

The MAST is a medium resolution trainer driven completely by OMIA software and MS FS software. Again, there is only one version of OMIA, and 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 that 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 software, they can be quickly reconfigured as Romeo stations via restarting the programs in Romeo mode.

OMIA as a Portable and Web Application

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

By making OMIA 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. Nevertheless, work is being done to make a portableapp JRE. For the rest of this discussion, OMIA Java will be referred to as a portableapp.)

A major advantage that has already proven 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, a USB drive can be plugged in and the OMIA software run directly from it . 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 that location. . This has made distribution to training classrooms trivial compared to the previous C++ OMIA, which required an installer, especially when automatically pushing installs to networked computers.

NMCI compatibility, as already mentioned, is a huge advantage provided by OMIA's 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 desks. There is a huge convenience factor in making OMIA available at a user's desk.

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 a portableapp, it was a small task to make it a WEB-based application.

3. FLIR ENHANCEMENTS

OMIA continues its earlier history, where each version not only progressively matched the evolving helicopter software, but functionality was expanded; the current OMIA has added a FLIR (Forward Looking Infrared) capability.

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 helicopter's HCU, but connects to a USB controller with a USB connector. This portable training HCU has been interfaced to OMIA.

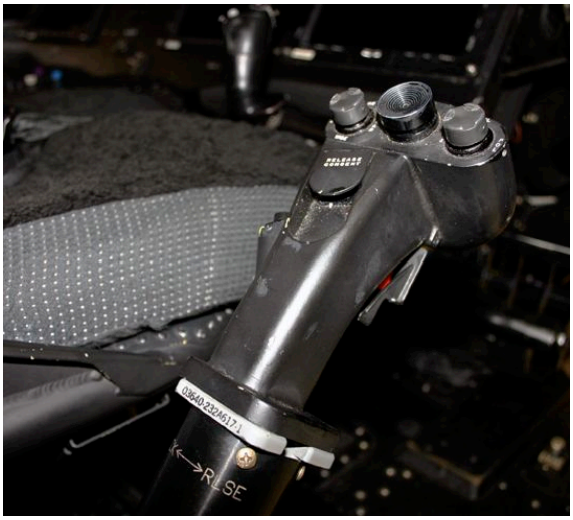

Figure 7. FLIR Hand Control Unit (HCU)

OMIA reacts the same way to the HCU hardware as it does to the presence/absence of other hardware units; when OMIA 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, then a software equivalent is provided (Figure 8).

However, neither of these HCU solutions meets all of the training requirements. The actual hardware is too expensive per HCU unit, while the simulated hardware does not provide the tactile feedback and muscle memory of a physical hand control unit. To address these issues, we have integrated OMIA with a second, low-cost, HCU based on technology originally developed for simulators rather than using an HCU from the helicopter (Figure 9). Due to the modular design of the OMIA software, adding support for an additional USB HCU input device required relatively little development effort.

Figure 8. Simulated HCU in OMIA.

Figure 9. Low-cost HCU built by Metters Industries.

An example of a FLIR, with the MH-60 overlay, as shown on a Mission Display in OMIA is shown in Figure 10. 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.

The FLIR implementation in OMIA uses a 2D FLIR image. Much of the learning related to FLIR concerns the operation of the FLIR menus and other operations that are part of the overlay. Through the combination of the hardware FLIR HCU and the overlay menus and other functions, a great deal of learning is facilitated. For example, users can zoom in and out, slew, adjust image polarity, cycle through camera modes, and navigate through on-screen menus.

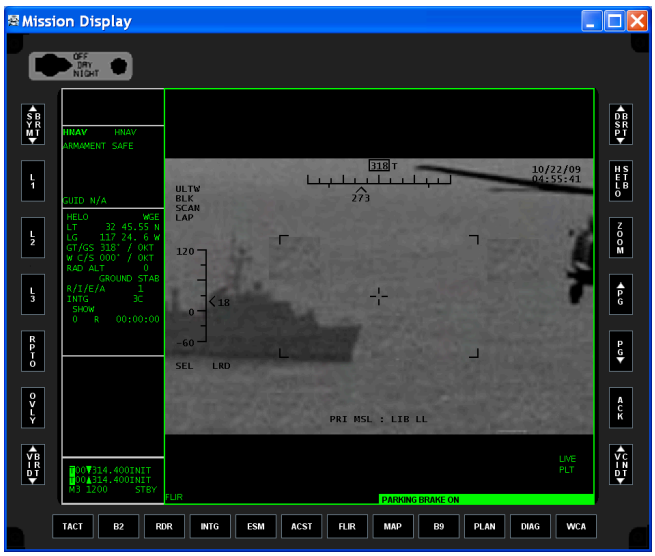

Figure 10. Screenshot of 2D FLIR in OMIA

4. EP-3E FLIR TRAINER

Recently, we developed a prototype version of a FLIR trainer for the EP-3E fixed wing aircraft based on the OMIA architecture, as shown in Figure 11. Two different types of engineering changes were required to support this, which can roughly be divided between *framework* and *flir*.

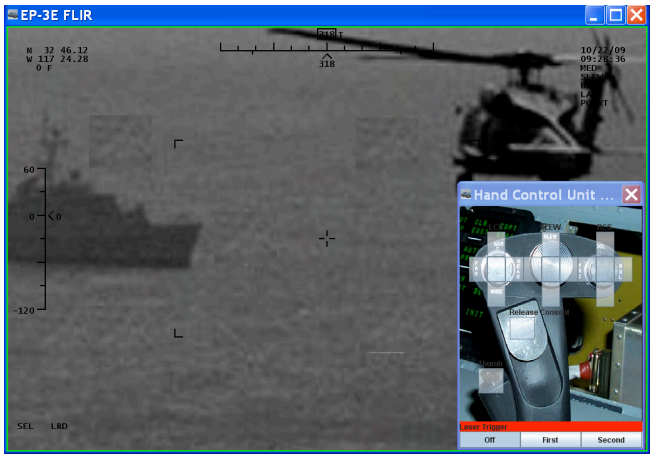

Figure 11. A prototype FLIR trainer for the EP-3E based on the OMIA architecture.

The *framework* changes were aimed at supporting a FLIR-only view, removing all of the MH-60 specific elements with as little programming as possible. The alteration occurs via a configuration property such as a command line argument. The changes include removing the flight display and keypads, changing frame titles and borders, and removing unneeded panels from the mission display. Once this UI adaptation was complete, the main functional change was to automatically enable the FLIR upon startup. This was done by simulating programmable, bezel, and fixed key presses on the hidden UI interaction components. The final modification was to replace the image in the splash screen to show an EP-3E rather than an MH-60. Overall, the framework changes took only a few hours of programming.

The *flir* changes were more substantial. While the two systems share much in common with respect to the actual functionality, both the HCU interaction and information displays are quite different. As a functionality example, the EP-3E does not have as many sensor modes as the MH-60, so the extra modes had to be hidden. As an example of an interaction change, pressing down on the right control on the HCU while in the main FLIR display brings up a weapons menu in the MH-60R; there is no corresponding function in the EP-3E. Pressing the right control down while looking at the main menu has the same effect in both systems. As an example of information display changes, the two systems have opposite vertical angle indicators, with the MH-60R displaying from -60 to 120, while the EP-3E runs from -120 to 60 for the same physical vertical slewing. More substantially, while the main FLIR displays mainly contain the same information, the locations for much of the information are different across the systems, as are the exact formatting requirements of this information.

These *flir* changes utilized the modular architecture used in OMIA. We were able to easily encapsulate the information display changes as we were already making use of a *model-view-controller* architecture [4]. In this case, the flir *model* contained all of the information about the current state of the simulated FLIR system, and the *view* controls, how to display this information. Switching to FLIR for EP-3E involves substituting the MH-60 view with an EP-3E view – the underlying model remains the same. The changes in FLIR functionality were very limited across the systems, so special cases were included in the FLIR *model* to accommodate this. Likewise, changes in the interactions supported by the FLIR *controller* were also limited, so special cases were added here as well.

There are two significant benefits gained from using the FLIR portion of OMIA for training on both the MH-60 and EP-E3E platforms. First, since the *controller* remains largely unchanged between the two systems, support is provided to both for input from either the software HCU or either of the hardware HCU units in both FLIR systems. Second, since the model is primarily the same for both systems any additional functionality added for one system

will be available to the other. This leverages development resources across the two FLIR trainers.

5. 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, with each iteration, it has become an ever more functional trainer.

The latest version of OMIA demonstrates the utility of designing a flexible system that allows for quickly responding to ever changing demands. As presented in this paper, OMIA was able to adapt to three new challenges with a minimum of additional development costs. The first challenge was an unexpected request to run on Linux; the second challenge was the addition of a new physical hand control unit for FLIR; and the third challenge was to leverage the work done for the existing FLIR trainer for the MH-60 to create a new FLIR trainer for the EP-3E aircraft.

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. Big Sky, Montana.

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

[3] Richards, R., J. Ludwig (2008) "Training Benefits of a Java Based Part Task Trainer", 2008 IEEE Aerospace Conference Proceedings. Big Sky, Montana.

[4] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). "Design Patterns: Elements of Reusable Object-Oriented Software." Addison-Wesley.

BIOGRAPHY

Jeremy Ludwig, Ph.D. is a project manager / research scientist at Stottler Henke, where his research areas include intelligent training systems, machine learning, and behavior modeling. He is leading the software development effort for the OMIA common cockpit helicopter training system for the MH-60S and MH-60R. Dr. Ludwig has also been involved in a number of other research projects that utilize game and simulation technology for training. These projects include a mobile, game-based training system developed for the iPhone to support the training needs of F-16 and F-22 pilots, a game-based second language retention system also delivered on the iPhone, an on-line simulation-based training system with learning adversaries in a multi-player virtual world for desktop training, and SimVentive™, an integrated development environment for the rapid construction of training games and simulations. Jeremy was a co-chair for the 2008 AAAI Fall Symposium on Adaptive Agents in Cultural Contexts and has been involved in a number of tutorials on the use of artificial intelligence techniques in serious games at IITSEC over the past four years. He joined Stottler Henke in the fall of 2000 and holds a PhD in computer science from the University of Oregon.

Robert Richards, Ph.D. is the Principal Scientist and Manager of Stottler Henke's Navy helicoptertraining contract, OMIA. OMIA is a PC-based desktop training system that teaches crewmembers the Navy's new MH-60R and MH-60S helicopters. Dr. Richards has taken OMIA 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.