

PC Rapid Modification Tool for Aircraft Experimentation & Training for the MH-60S/MH-60R Helicopters

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

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

Abstract—The US Navy's PMA-205 in conjunction with Stottler Henke has developed and deployed a tool that can be used for training and AOP experimentation for the US Navy's new MH-60S and MH-60R helicopters. The tool, called the Operator Machine Interface Assistant (OMIA), is primarily an expandable, easily modifiable low-cost PC-hosted desktop crew trainer. OMIA is currently in use for training at HSC-2, HSC-3 and HSM-41. However, since both helicopters have been constantly evolving through the development process of OMIA and continue to evolve now and in the future, OMIA had to utilize flexible, low-cost, rapid development methods.^{1,2}

This reality has resulted in a product that is not only valuable for training, but is also useful for Avionics Operational Program (AOP) experimentation, design and testing. For example, both the MH-60S and MH-60R helicopters include Lockheed-Martin's Common Cockpit that includes a programmable keyset (PK) as part of the center console. Changes to the PK have been a major component of the helicopter's evolution. To more rapidly deal with these changes, we have developed the separate KeyBuilder tool, which allows a non-programmer to update the programmable keyset via a drag and drop interface. That is, any user can open the KeyBuilder and move keys within a layer and between layers; once the new configuration is completed the user saves out a text file that contains the information describing the new configuration. When OMIA is restarted it will find the new keyset definition file and use it for determining how to display the programmable keys.

The present method requires Lockheed-Martin to be involved which requires many weeks to see proposed changes. With OMIA and KeyBuilder a test pilot or a human factors engineer could easily make a potential improvement and then conduct experiments to test and verify the changes as an improvement. Only after quantifiable evidence shows that the change is beneficial would the change be requested of Lockheed-Martin. For example, the CogTool cognitive architecture has been used, in conjunction with OMIA, to compare two different alphanumeric key layouts used in MH-60S and MH-60R. [3]

Other aspects of the user interface can be changed rapidly also. For example, another major user interface component is on-screen menus. A menu generation and modification tool has also been developed. The more complex cockpit components have been developed with rapid-development tools; these are DiSTI's GLStudio and Microsoft Flight Simulator (MS FS). These tools make it easier for developers to move or change major elements. The addition of a head-tracking head-mounted display allows MS Flight Simulator to provide a 3D view of the cockpit; one may look around the virtual cockpit quickly just by moving one's head. An engineer or pilot might wish to have the cockpit modified in some way; this could be performed in a day by a programmer.

This paper shows not only the direct benefit to the MH-60 program that having a rapidly-modifiable tool provides, but also lessons learned to help demonstrate how other aircraft could also benefit from such tools to help with more efficient aircraft evolution. That is, none of the techniques shown here are unique to the MH-60 helicopters so they could be applied to other aircraft bringing more power to those closest to the actual flight testing and evaluation.

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. OMIA CONFIGURATIONS	3
3. INTERFACING WITH FLIGHT SIMULATOR.....	4
4. KEYBUILDER	5
5. MENUBUILDER.....	6
6. MISSION AVIONICS SYSTEMS TRAINER (MAST) 7	
7. PLANNED ENHANCEMENTS.....	8
8. CONCLUSION	8
REFERENCES	8
BIOGRAPHY	8

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

¹ 1-4244-0525-4/07/\$20.00 ©2007 IEEE.

² IEEEAC paper #1341, Version 3, Updated December 20, 2006

(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”. The US Navy's PMA-205 in conjunction with Stottler Henke has developed and deployed 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-60S

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.

OMIA 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 emphasizes some of the flexible, low-cost, rapid development methods used more commonly by industry than the military to date in order to reach the present deployment and how OMIA is also useful for Avionics Operational Program (AOP) experimentation, design and testing. These methods include; re-use of COTS software, design for evolving COTS hardware/software, and design for evolving requirements. COTS software utilization includes the integration of Microsoft™ Flight Simulator for many aviation aspects of OMIA, and the use of two- and three-dimensional COTS libraries by DiSTI to develop the user interface and cockpit.

The foundation that permits the flexible and rapid use of different components is the underlying design for evolving requirements that the Navy has required. 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 are not at the same version. The Navy supports these differences in OMIA. When OMIA starts, it asks which platform and seat it should emulate, see [Figure 3](#), and [Figure 4](#).



Figure 2. Common Cockpit

The flexible design allows OMIA 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. Continuing enhancements of the trainer are allowing it to teach an increasingly broad variety of aviation and mission tasks; for example, OMIA software driving the MH-60S Mission Avionics Systems Trainer (MAST). In addition, being PC-based OMIA can be placed on any number of machines, both on land and at sea.



Figure 3. OMIA Splash Screen

R
F
R
D
R
F
R
D

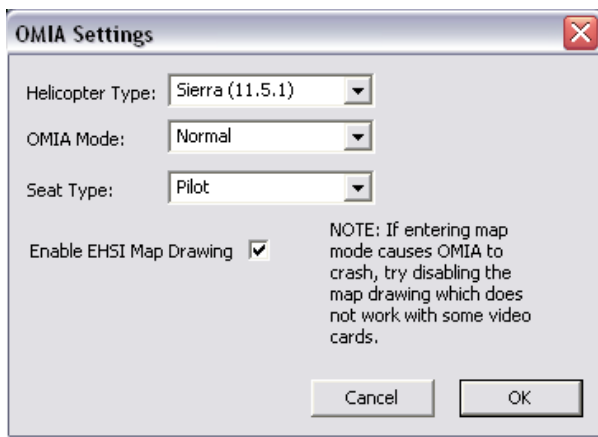


Figure 4. OMIA Splash Screen Configure Settings

The various components of the OMIA software have been implemented in a distributed manner to provide greater flexibility for future enhancements to the software. To deal with this reality, the OMIA components use the concept of an external system interface (ESI) to communicate between modules that may be distributed in the future. Presently, the operator system interface PTT and the Simulator (and the communications package) all communicate with the rest of the components via the ESI.

To allow for more rapid updates of the AOP, additional tools have been developed so developers and non-developers can make many types of modifications to the OMIA interface. The two aspects of the Common Cockpit that have changed the most over the AOP evolution are the layering and arrangements of keys in programmable keyset, as well as the user interface menus displayed on the mission display.

To more rapidly deal with these changes, we have developed the separate *KeyBuilder* tool, which allows a non-programmer to update the PK via a drag and drop interface. That is, any user can open the *KeyBuilder* and move keys within a layer and between layers; once the new configuration is completed the user saves out a text file that contains the information describing the new configuration. When OMIA is restarted it will find the new keyset definition file and use it for determine how to display the programmable keys.

We have also developed a menu generation and modification tool, *MenuBuilder*, in order to more rapidly update the evolving menus. This permits non-programmers to create or modify existing menus.

2. OMIA CONFIGURATIONS

As mentioned above, OMIA has many configurations. The core OMIA is a standalone product that operates under any standard Windows 2000 or Windows XP computer; this standalone product 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 and 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 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 can be used to teach both the Sierra and Romeo versions of the helicopter. When the program is started, the user has the option to change the present configurations. They can select either the Sierra or Romeo helicopter and once they select the helicopter they can select the seat. They can select the pilot or co-pilot seat for the Sierra or the pilot, co-pilot, or sensor operator seat for the Romeo. In addition, the user can also state whether they are flying solo, as in standalone training, or if they are in a network configuration, in which one operator can be the pilot and one operator can be the co-pilot. 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 as the client and the program will search for a server for them to join on the network.

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 5 and Figure 7, and the Flight Display, shown in Figure 6, 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.

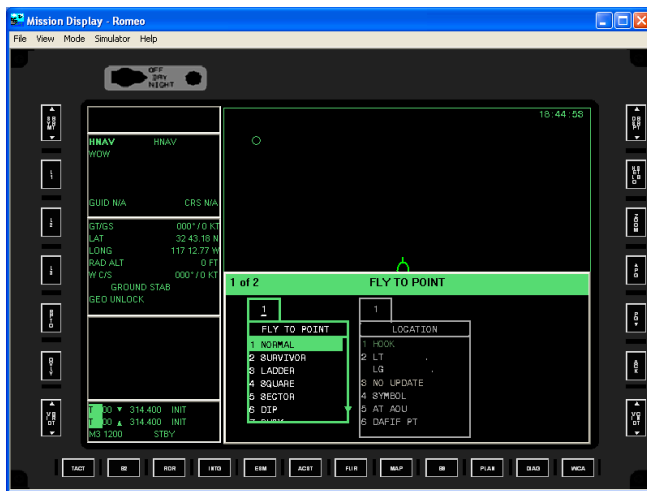


Figure 5. Mission Display with FTP menu



Figure 6. OMIA Flight Display

At this time, software additions for OMIA consist only of Microsoft Flight Simulator. Every time OMIA starts it checks to see if Microsoft Flight Simulator is already running. If it is running, OMIA 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. 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 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.

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. The MAST is further described in a section below.

3. INTERFACING WITH FLIGHT SIMULATOR

MS FS is itself an extensible program with Microsoft providing the Microsoft Flight Simulator SDK specifically for this purpose. In addition, third parties provide many extensions to MS FS, one being a more powerful version of the SDK called


FSUIPC (www.schiratti.com/dowson.html).

The FSUIPC Dynamic Link Library (DLL) allows external programs to communicate with (and in many situations control) MS Flight Simulator. Many companies and hobbyists have interfaced extensions, including external hardware, to MS FS using FSUIPC.

Upon startup, OMIA silently searches for a running instance of MS FS using FSUIPC, and if found, the interface is automatically established. Otherwise, OMIA will run in its non-MS FS mode and work properly. If the interface is established, OMIA utilizes FSUIPC to get the:

- wind heading
- wind speed
- ground speed
- true heading
- altitude, and
- latitude and longitude.

This information is used to provide correct readings for the OMIA interface that shows the user all of the above information. An update is performed once each second.

Currently, OMIA only gets information from FS, but successful experiments have been conducted in setting information as well. An example of the interface in action is represented in Figure 7. In the figure, the MH-60S is being flown from North Island Naval Air Station, near San Diego California. The top of the figure shows the out-of-window view provided by MS FS, below is the mission display of OMIA, and to the right is part of the center console. In the mission display there is a hexagonal icon  representing

latitude (Lat),
longitude (Long),
ground track (GT),
ground speed (GS),
radar altitude (RAD ALT)

This level of integration greatly enhances the realism provided by OMIA. For example, many of the training exercises require the pilot and/or copilot to establish a set of fly-to points (FTPs) and then fly to (capture) the FTPs (while performing other operations). The establishment and management of FTP points are performed via the mission display and center console. Figure 5 actually shows the menu for establishing a FTP. With flight simulator incorporated, OMIA can have the operator fly to the FTPs and then the mission display and the rest of OMIA will be updated properly when the operator captures the FTPs.



4. KEYBUILDER

KeyBuilder (2005.01.06): c:\projects\OMIA\OMIA_OST\keys_romeo.txt

File Check for Problems

Layers

ACST
ACT 2R
AH
ALPHA
APPS
AQUA CTRL
ATAK
ATAP
AUD
AUD CTRL
BBO
BUOY CTRL
BUOY INV
BUOY PING
BUOY PRCS
COMM
COMM PSET
DAL CTRL
DCLT CTRL
DCSS BUOY
DCSS DIFR
DEF FRIN
DEF FTP
DEF LEGS
DEF WYPT
DEF WYPT

Keys

Sort: C by Name C by Type

1
2
3
4
5
6
7
8
9
<<
>>
ABC 1
ABRTH

TACT B2 RDR INTG ESM ACST FLIR MAP B9 PLAN DIAC WCA

RBC 1 DEF 2 GHE 3
JKL 4 MNO 5 PQR 6
STU 7 VWX 8 YZ 9
SPAG

Change ATO Box 11.5 Menu Layer

Using the *Run* mode also allows users to simulate navigation through the programmable keyset layers, again in either SO, on the lower left, or CO/ATO, on the right.

A *layer* is a list of 32 keys that will be displayed whenever the layer is selected. There are eight different types of keys available in the KeyBuilder program, each of which has a different effect.

- 5

“^”. Pressing the same layer key again will remove the mark and navigate back to the previous layer.

6. Toggle Key – Toggle keys are used to toggle a system variable (e.g. landing gear). They are marked with a “*” when toggled.
7. MultiStateToggle Key – A MultiStateToggle key is used to toggle between 2-12 items. For example GRD1 T/R to GRD1 -/R to GRD1 -/- to GRD1 T/R. Another example is RAD1 AM to RAD1 FM. Each state is completely customizable. The name chosen for the toggled key must be the same name as its initial state. Furthermore, each state may have a layer associated with it. When the state of the MultiStateToggle Key changes, so does the sub layer. A good example of this is the key entitled RAD1 V/U on the COMM layer. When pressed, it cycles through four states and four layers.
8. LinkedMenu Key – A LinkedMenu key brings up a menu and a new layer, and the value selected in the menu determines the state of the key. A LinkedMenu key has 2-12 different labels, like a MultiStateToggle key. However, the currently displayed label is selected via a menu, rather than through round-robin button selecting. Also, unlike MultiStateToggle keys, all states of a LinkedMenu key link to the same layer. (See “Known Problems” below for issues with LinkedMenu keys).

AOP experimentation can be performed quickly with regard to the programmable keyset (PK) layers. By simply dragging and dropping key names a new arrangement can quickly be constructed. Experiments can then immediately be performed within KeyBuilder. If these initial experiments show potential, then the KeyBuilder text file can be used with OMIA and further experiments can be conducted.

5. MENUBUILDER

The MenuBuilder utility has been developed so non-programmers can build from scratch or modify present menus. The MenuBuilder is not as evolved as the KeyBuilder so it is not as graphical. The MenuBuilding process consists of answering a series of questions. These commands are described below. See Figure 9 that shows the menu used in the example below.

1. Please enter the key name (e.g : *INSR BUOY*):
For example: : *INSR BUOY*
2. Please enter the class name
Use the key name with ‘Menu’ and underscores for spaces (no slashes or spaces).
For example: *INSR_BUOY_Menu*

The program will check if such a class exists already. If it does exist, it will ask you if you want to overwrite. If you do not want to overwrite type “n” and no changes will be done to that menu. If you overwrite, the old menu will be completely erased.

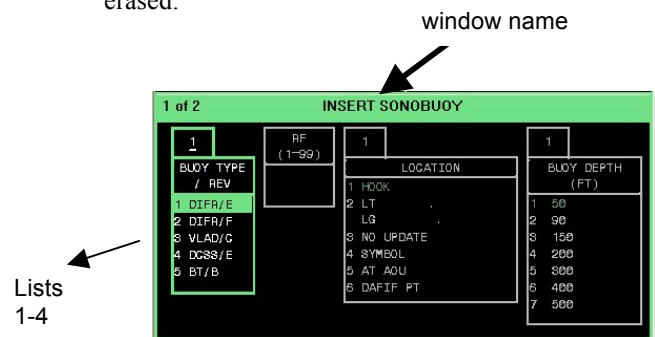


Figure 9. Example Menu

3. Please Enter Window Name
The window name is the text seen in the outer frame of the menu. Enter it identically as seen on the menu.
For this example: *INSERT SONOBUOY*
4. Please Enter Number of List
This is the number of section boxes within the menu. If there are lists inside of lists, only count the “child” lists.
For this example: 4

There are four types of lists:

- a. Location List:
Location lists will be labeled as a location list. It is a common list and the properties are already set so you will not have to identify each line within the list.
- b. List with Selection Indicator:
Selection indicators are the top small box above a list.

Lists 1, 3, and 4 (left-to-right above) have selection indicators.
- c. List without Selection Indicator:
Lists without selection indicators have no upper small box.

List 2 (second from left) is a list without a selection indicator.
- d. Multi-list combined in a single list:
These lists are found inside of other lists. (See example in Figure 10).

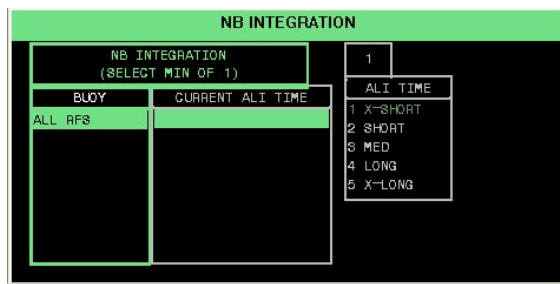


Figure 10. NB Integration Menu

NB INTEGRATION has three lists. The first two lists are type 'd', and the third one is type 'b.' The first two lists are "multi-lists" inside of a bigger list.

5. List Type
Beginning with the first list on the left, identify its type: *a*, *b*, *c*, or *d*.
6. Please Enter List Label for List 0.
The script numbers lists starting at 0. You may have four lists; but the script will identify them as 0, 1, 2, and 3.
The list label is the name in the top box. For the INSERT SONOBUOY example: BUOY TYPE\r\n / REV

Note: The characters \r\n are in place for 'next line' character. If you have a title/label spanning two lines, be sure to insert \r\n for the next line command. The \r\n must be in all lower-case, otherwise it will not compile.

7. Please Enter List 1 Item (enter '.' to end)

Note: If your list contains text/names/numbers, enter as they are shown. If the list appears to be an interactive list, where numbers or text are to be inserted, use these place-holding characters to indicate these fields:

- a. NN- for numbers
- b. CH- for characters
- c. NS- for NorthSouth
- d. EW- for EastWest

Each placeholder is for one digit/character. Enter as needed for multiple placeholders.

8. Which item do you want to select (0- NN-):
Indicate the item (list line) number. Keep in mind the list items are numbered from zero, not one.

Note: On most lists there is a selection that is highlighted and often it is the first item in the list. This is where you identify that item.

9. Would you like to highlight this item by default? (y/n)
Indicate *y* or *n*.
10. Would you like to register this key with menu? (y/n)
Enter *y* if this is a new menu, or *n* if this is an existing menu.

After these steps the menu information is written out so that the next time the program is *compiled* the menu changes will take effect.

So in its current incarnation, MenuBuilder is not something that would probably be used by human factors engineers or test pilots directly; but it does allow for very rapid changes to the menus so requests for changes by human factors engineers or test pilots could be fulfilled in a few days.

6. MISSION AVIONICS SYSTEMS TRAINER (MAST)

The MAST, as shown in Figure 11, 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 11. 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, it can work with or

without MAST hardware, demonstrating that flexible software can be created to exist independent of 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 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 software, they can be quickly reconfigured as Romeo stations via restarting the programs in Romeo mode.

In addition, AOP experimentation can now be performed in a more realistic environment. That is, all the changes mentioned above can be made and then used in the MAST. For example, one could use KeyBuilder on another machine, change how some of the layers are laid out and then copy the KeyBuilder output (a text file) to the MAST machine and test the new layers there while pushing physical hardware keys.

7. PLANNED ENHANCEMENTS

An enhancement currently in production is the construction of a low cost version of the PK/FFK/Hook center console (Keyset) hardware unit. This will be used primarily for Romeo training for both the front seats (pilot/copilot) as well as the rear seat (sensor operator). The low cost USB-based reproduction will maintain the same length, width, and layout as the real hardware for training accuracy, but will be much thinner so it can be used in a computer lab environment or a notebook computer at sea. This enhancement will significantly improve the efficacy of the OMIA trainer by having students press the actual key rather than the artificial action of using a mouse to press a key on the screen.

A second, related, enhancement currently under investigation is creating a low cost hardware solution for the MD and FD bezel keys. The goal is to create a low cost PC based training system with two bezel key screens (one for MD and one for FD) and one Keyset unit described above. This complete trainer would allow students to learn the MH-60S and MH-60R systems in a realistic manner while still maintaining the benefits of low-cost and portability.

A separate option being researched is attaching a FLIR hand control unit so FLIR operations can be performed via a PC with the hardware being plugged into the USB port.

OMIA will react the same way to any of these hardware units; when OMIA starts up it will detect which hardware is attached. In the cases where there is a software equivalent shown by default for the attached hardware, the software equivalent will not be displayed. For example, if there is hardware for the Keyset connected to the USB port, then the

software version of the Keyset will not come up automatically when OMIA is started.

8. CONCLUSION

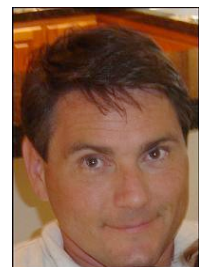
The complexity and number of the sensors under control of the crew on the MH-60S and MH-60R helicopters pose a difficult training task for the Navy. To meet this challenge the US Navy's PMA-205 in conjunction with Stottler Henke and various hardware vendors has developed and deployed OMIA, a flexible, low-cost PC-hosted desktop crew trainer. OMIA is a flexible solution leveraging COTS solutions, designed for evolving needs and able to be integrated with MS Flight Simulator and custom hardware for enhanced functionality. To learn more regarding the past, present and future of OMIA, please visit the project web page at www.StottlerHenke.com/omia.

REFERENCES

- [1] Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.) (1999). How People Learn: Brain, Mind, Experience, and School. Washington D. C.: National Academy Press.
- [2] Stottler, R. H., & Vinkavich, M. (2000). Tactical action officer intelligent tutoring system (TAO ITS). IITSEC 2000 Proceedings.
- [3] 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. OMIA 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 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.

