Application of EuroSim in the F-35 Joint Strike Fighter Embedded Training solution

Leon Bremer[(1)], Erwin Schreutelkamp[(2)]

[(1)] *Dutch Space, Mendelweg 30, 2333 CS, Leiden, The Netherlands. Email:l.bremer@dutchspace.nl*

[(2)b] *NSpyre, Herculesplein 24, 3584 AA, Utrecht, The Netherlands. Email:erwin.schreutelkamp@nspyre.nl*

The F-35 Lightning II, also known as the Joint Strike Fighter, will be the first operational fighter aircraft equipped with an operational MultiShip Embedded Training capability. This onboard training system allows teams of fighter pilots to jointly operate their F-35 in flight against virtual threats, avoiding the need for real adversary air threats and surface threat systems in their training. The European Real-time Operations Simulator (EuroSim) framework is well known in the space domain, particularly in support of engineering and test phases of space system development. In the MultiShip Embedded Training project, EuroSim is not only the essential tool for development and verification throughout the project but is also the engine of the final embedded simulator on board of the F-35 aircraft. The novel ways in which EuroSim is applied in the project in relation to distributed simulation problems, team collaboration, tool chains and embedded systems can benefit many projects and applications. The paper describes the application of EuroSim as the simulation engine of the F-35 Embedded Training solution, the extensions to the EuroSim product that enable this application, and its usage in development and verification of the whole project as carried out at the sites of Dutch Space and the National Aerospace Laboratory (NLR).

1 INTRODUCTION

Since the mid-'90s Dutch Space and the National Aerospace Laboratory (NLR) have been working on developing the concept of Embedded Training (ET) for fighter aircraft. Several projects have been performed successfully like paper studies and implementation of several demonstrators. The two most notable are the Embedded Combat Aircraft Training System (E-CATS) demonstrator in a Royal Netherlands Air Force (RNLAF) F-16 ([1]) and the Multi-Ship ET demonstrator on NLR's Fighter 4-Ship simulator under contract of the F-35 Program Office ([2]). The successful E-CATS demonstration flights on the RNLAF F-16 have boosted the interest from the fighter pilot training community. The knowledge has been supporting the F-35 program and enabled others to expand on this base. Currently, Dutch Space and NLR are, under contract of Lockheed Martin, implementing the core functionality of the F-35 ET system. This comprises main parts of the E-CATS extended with specific training features for pilot training at the F-35 Integrated Training Center (ITC). When fielded, this system will be the first to incorporate multiship embedded training in operational fighter aircraft, allowing multiple aircraft to participate in a synchronized exercise.

Throughout the various embedded training projects, the European Real-time Operations Simulator (EuroSim) framework has played a major role in development, execution and verification of the embedded simulation software. This real-time simulator tool is well established in the space domain, where in particular it has established itself in the area of engineering and AIT. Space systems as the European Robotic Arm, the Automated Transfer Vehicle, the Herschel and Planck satellites and most recently the Gaia satellite have been engineered and verified with support of EuroSim simulators. With the application in Embedded Training, EuroSim has not only been applied in a different domain, but also in a different role. In the space domain the real-time simulator supports the project in establishing a space system, but it is not part of the system once it is deployed. In Embedded Training however, the onboard application is a EuroSim based simulator that operates as an embedded component of the mission software in flight during trainings missions. As the F-35 Embedded Training solution is capable of supporting multiple aircraft in a joint exercise, the EuroSim based embedded components must synchronize to form a distributed real-time simulation.

This paper first provides a summary of the concept of multiship embedded training as well as of the EuroSim framework to picture the challenges and needs of the Embedded Training project. A description of developed processes, toolchains and infrastructures follows. Thereafter, new technological innovations in EuroSim are elaborated that have been established in support of the Embedded Training project. The paper concludes with observations made in the project on the usage of EuroSim and lessons learned for future application of the developed technology.

2 MULTISHIP EMBEDDED TRAINING

By definition, Embedded Training provides training capabilities built into or added onto operational systems, subsystems or equipment to enhance and maintain the skill proficiency of personnel. ET enables the operator to use the "real" weapon system in situations for which it was designed, even if these situations are not available in every day life.

The ET System (ETS) is thereby an integral part of the overall training system and can be used on a daily basis at the operational and training squadrons. The ETS can be used in specific lessons in the syllabus at the schoolhouse with tailored scenarios. At the operational squadrons it can be used almost daily in flights in a variety of training mission types, either as the sole support of training or as a force augmenter, overlaying the real threat environment with virtual threats at no cost.

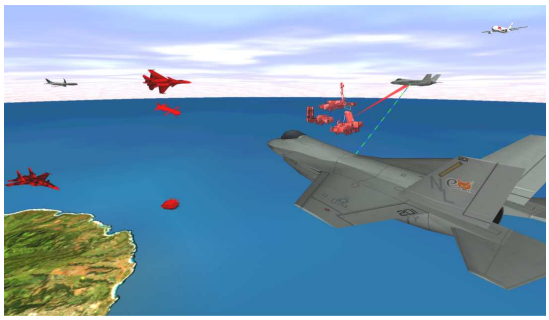

Figure 1 Virtual threat overlay

The ETS for fighter aircraft injects virtual air and surface-to-air (SAM) entities into an aircraft mission system. This provides fighter pilots more intensive training in training missions populated or augmented with virtual hostile, friendly and neutral scenario players, with a debrief playback capability on-ground.

For this purpose the ETS comprises a *Virtual World Simulation* module which is the source of the virtual environment being superimposed into the cockpit. It simulates the subsystems and behavior of virtual aircraft and SAM. For obtaining a realistic overlay an *Own-ship Simulation/Stimulation* module includes sensor simulation of the ownship in order to determine which virtual players are detected. Both modules include simulation of offensive and defensive subsystems (missiles, jamming, chaff, etc.) which cause mutual interaction. As supporting modules the ETS includes a *Safeguarding* module that monitors safety limits and halts the scenario execution when limits are exceeded and a *Data Logging* module that records data to be used for debrief on the ground.

The ETS system as described would allow a single fighter pilot to train against the onboard generated virtual threats. However, in modern day warfare, fighter aircraft are employed in groups, utilizing extensive communication features and team concepts to accomplish their mission. To train as they fight, an Embedded Training solution must therefore incorporate support for MultiShip operation.

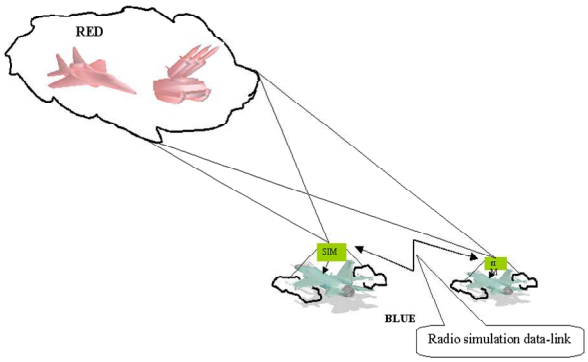

Figure 2 ET synchronization over datalink

A *Simulation Data Link* is added to the ETS to provide synchronization of the scenario across multiple participant aircraft. The main functionality of the Simulation Data Link module consists of:

- Exchange of data for the Simulation Management module in order to realize the coordinated startup and termination of scenarios across multiple participants.
- Synchronization of the Virtual World Simulation modules on each participating aircraft.

Where SingleShip Embedded Training is an embedded real-time simulator application, the inclusion of datalinking makes MultiShip Embedded Training a distributed embedded real-time simulator application. Both Simulation Management and Virtual World Simulation synchronization functions have to deal with latency, bandwidth constraints and temporary data link loss.

Besides the immediate cost benefit as a result of minimizing the need for real air and surface threat systems, the ETS is also expected to reduce the need for dedicated training areas as well as lower the environmental impact of fighter training. As most sorties are training sorties, these benefits come in large quantity.

3 EUROPEAN REAL-TIME SIMULATOR

The European Real-time Operations Simulator (EuroSim) is a tool to support development and application of real-time simulators. The product is developed and commercially made available by the EuroSim consortium, a collaboration of the companies Dutch Space, NLR and Nspyre. EuroSim was initially targeted at the space domain, where it is applied throughout the project life cycle from space mission analyses to Assembly Integration and Test. The key feature of EuroSim is that it is capable of meeting the hard real-time performance and reliability demands of space system hardware in the loop tests as for instance required for Attitude Orbit Control Systems (AOCS). Developing hard real-time solutions on general purpose computers requires in depth knowledge and experience in the area of operating systems, drivers and computer hardware. A EuroSim user however, only has to express real-time constraints supported by graphical editors. Simulator developers can thus add their application specific knowledge as software models without being distracted about the low level details of real-time software engineering.

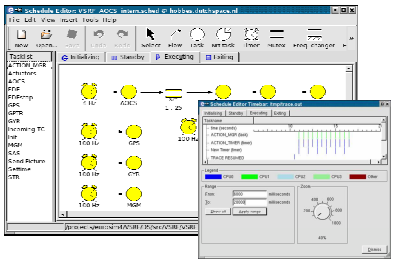

Figure 3 EuroSim real-time schedule definition

A distinct difference is made by EuroSim between simulator development and its application in test campaigns. The (model) developers construct the simulator executable and schedule definition. This is still a highly iterative process with many tests to assure that the simulator performs correctly. This process however has a white box nature and EuroSim provides extensive support to look into the simulator in real-time, non intrusively, as well as for debug support. When the simulator construction process is completed, the transition to verification is made. In the verification phase, users write scripts and add logging and visualization specifications to verify the correct operation of the simulator or space system, possible in conjunction with hardware in the loop in various missing item simulator configurations. The focus thus shifts to a black box testing approach, and tooling in EuroSim shifts to another set of user interfaces.

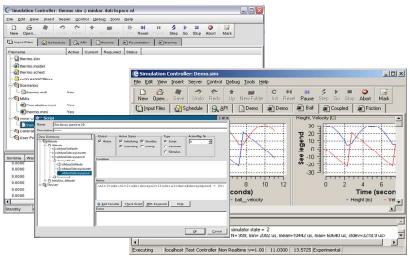

Figure 4 EuroSim Runtime tooling

4 EUROSIM IN THE PROJECT LIFE CYCLE

In past ET projects, most notably the demonstration of E-CATS on the RNLAF F16, EuroSim has proven itself as a highly valuable asset. The demonstrator on the F16 used EuroSim as a development, application and Analysis platform and achieved a flawless performance in ten successive sorties, after a remarkably quick development time of 8 months. The F-35 ET project however aims at developing an integrated operational MultiShip capability that was to become part of the F-35 mission software itself. The project would have a significantly larger team, specialized target environment and many new requirements, of which MultiShip synchronization would be the new and thus highest risk factor. A fairly short project life time and a consequently large group of developers combined with highly specialized and limited availability of target hardware, means that the key factor for success is a seamless flow from design through application to verification on an easy general purpose fast computer. An automated porting process then supports the transition to the target hardware.

An analysis showed the needs for a tool environment supporting the following:

- Modeling in UML with a single automatic code generation phase producing as much as possible code automatically.
- Programming in C++ and utilizing the Eclipse Development Environment
- Developers of software models and scripts working on general purpose servers whereby EuroSim provides the abstraction (and port to) the specific target hardware platform
- Many developers working in parallel, preferably on a server setup for security reasons
- A constantly ongoing Integration process to avoid any major integration problems as time progresses
- Development, integration and qualification test systems supporting single ship and multi ship

configurations as well as faster than real-time simulation and extensive debugging support

- Nightly build and tests that can build and run the integrated system, as well as perform metrics, coding rule checks, coverage analysis etc.

Figure 5 describes the automated tool chain to support development, verification and target generation as established in the project.

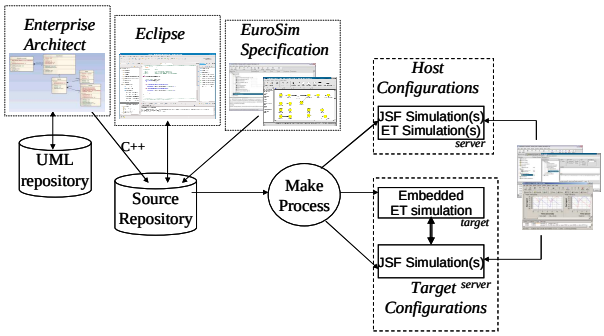

Figure 5 Tool chain in the ET project

In the top left of Figure 5 is the ET design defined in UML using the tool Enterprise Architect. EuroSim provides enhanced code generators for this tool to produce C++ code from the class diagrams and extends that with code to connect the software to EuroSim. The C++ code is entered in a code repository and the developers use Eclipse to add functionality to the generated code. EuroSim tooling is used to specify the model build and real-time execution definitions. This specification is made once and hardly changes throughout the project as it relates to higher level design.

From the code base, a make process can be activated to create various simulator configurations. Host configurations are simulators that run ET on the server, supporting developers with the quick program and test cycle. Target configurations support verification processes where the ET simulation runs as an embedded simulator on the target hardware. However from the perspective of the control interface to the user the interface for host and target configurations is the same.

The tool chain is available on a powerful 8-core server with a single target hardware system attached. Two identical setups are available, one at NLR and one at Dutch Space, allowing for contingencies and usage of one setup for qualification test cycles while next release development can occur on the other setup.

5 SUPPORTING ET DEVELOPMENT

For Multiship Embedded Training application development the developers work on two major areas: threat modeling and MultiShip synchronization protocols. In both cases the developer wants quick feedback on the behavior of his code as it progresses over time. Faster than real-time runs, extensive debugging, monitoring, recording and stimuli become essential features. EuroSim has built in support for such cycles, and for single ship simulations development of the ET simulator on the host is already fully supported.

For MultiShip simulations however, an environment has to be established that incorporates the ET simulators of multiple F-35s. Each such ET simulator is capable of a single ship simulation, as well as collaborating with other ET simulators in MultiShip operation. At first thought, it seems logical to apply typical distributed simulation solutions utilizing network connections. However, the configuration needs to support debugging of protocols where internals of the simulators need to be visible of all ET simulators in synchronization. Additionally faster than real-time execution, step by step execution and symbolic debugging are required. Rather than creating extensive tooling and solving synchronization issues, the ET project took the approach of integrating the ET simulators in one real-time simulator. All standard features of EuroSim then become available in the same way as for Single Ship configurations. Figure 6 shows the architecture of this solution.

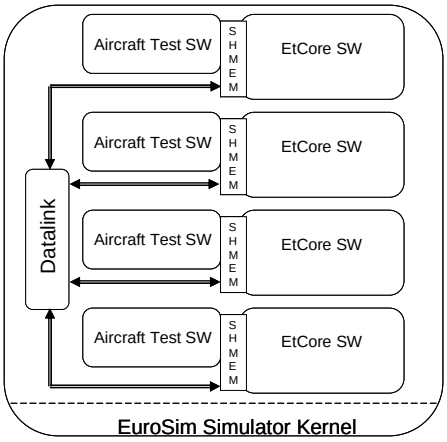

Figure 6 MultiShip Simulator, Host configuration

The key enabling EuroSim feature that makes this configuration work is the multi processor scheduling that allows each ETCore SW component to be mapped on its own processor. When real-time mode is selected these processors must be available, but in non real-time

situations EuroSim emulates these processors. In addition a standard linking feature allows us to wrap the EtCore SW binary library in a shared library without exposing symbols. This automated extra link step in the make process allows us to duplicate the binary code multiple times while each ETCore is completely unaware of the presence of other EtCore components in the same process.

The Aircraft Test Software and data link model are part of the ET project test environment and are a simple generic abstraction of an F-35 provided environment of ET. The data link model simulates the RF link that propagates message that the ETCore SW exchanges. This model can be configured to drop connections or insert delays and errors in order to stress the EtCore SW. The benefit of the availability of host based development support configurations is thus not only limited to EtCore SW developers. The formal system verification of ET is a black box test approach which requires test models as well as extensive scripts to verify the ET behavior. EtCore software developers utilize the graphical user interface to interact with the simulator. The test system developers write such interaction utilizing Python with EuroSim extensions. Every interaction that can be performed with the GUI can also be performed from a Python script, and once at that level functions can automatically analyze and evaluate data on pass fail criteria that relate to the system requirements.

6 SUPPORTING ET TARGET GENERATION

While the EtCore SW developers produce the MultisShip Embedded Training capability on top of EuroSim on a standard Unix server platform, the EuroSim consortium developed its new embedded simulator technology solution to be able to cross build an ET simulator from the host to an embedded target computer The embedded simulator solution consists of the following main features:

- A port of EuroSim core modules as the operating system layer, utility libraries, scheduler and dictionary. (Additional EuroSim modules can be added but for ET would unnecessarily enlarge the footprint),
- Tooling to transform EuroSim specification files (e.g. schedule) into source code that is linked into the application. This avoids the need for a file system on the final embedded target hardware.
- Interfaces to allow user defined connection to streaming interfaces (e.g. logging).

By using this approach the final application runs on the same EuroSim framework as the one on which it was developed. This greatly reduces integration and performance risks when the transition from development system to final target hardware is made. The embedded simulator guarantees hard real time performance. This means that when the same schedule file is used as during development the timing on the target will be identical. Furthermore, EuroSim has been enhanced with improved metrics functionality that allows detailed monitoring and reporting of process loads and memory usage on the final target hardware. This makes it possible to identify any performance issues early on when transitioning to the final target hardware.

An embedded simulator has a simpler kernel than its full-fledged EuroSim brother as it requires no scripting and user interaction capability during operation. It comprises only those modules that are needed to execute the final application in a real-time environment on the target hardware. To isolate the application as much as possible from the rest of the system there is only a Shared Memory interface with the outside world. As a consequence all configuration files needed to run the embedded simulator must be embedded in the executable. The schematic of a typical embedded EuroSim is shown in Figure 7

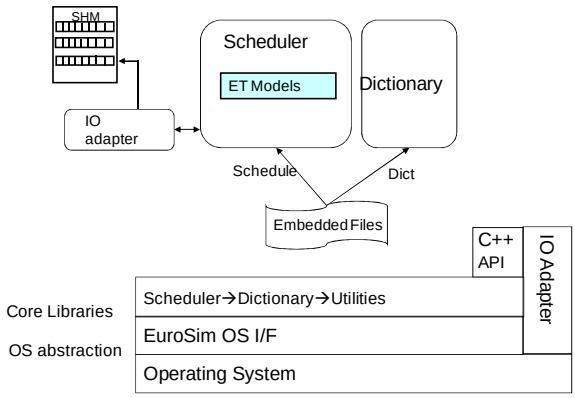

Figure 7 EuroSim Embedded Simulator solution

As illustrated, the modules in the embedded EuroSim simulator are the scheduler, dictionary, system level libraries and the IO Adapter. The latter plugs in a shared memory based interface through which communication takes place. The application level model code together with this infrastructure forms the onboard application. The Operating System layer is an abstraction layer on top of the native operating system.

For the ET project the native operating system on top of which the embedded simulator runs is the safety critical Integrity-178B operating system. For other embedded simulators this may be a different (real-time) operating system such as VxWorks or even be purpose built.

Depending on the system requirements and availability, other hardware interfaces can be added to the simulator.

7 SUPPORTING ET VERIFICATION

With the host based ET simulator configurations, an efficient development environment has been created. With the embedded simulator technology the ability is created to automatically generated an embedded version of a host developed ET Simulator. However, a final interconnection is needed to support the formal system verification process with the embedded ET Simulator executing on the target system. This interconnection is a replicator mechanism to allow the system verification test scripts that were developed against a host based ET Simulation configuration, to be rerun with a target based ET Simulation. Figure 8 shows the replicator solution for a single ship host and target configuration:

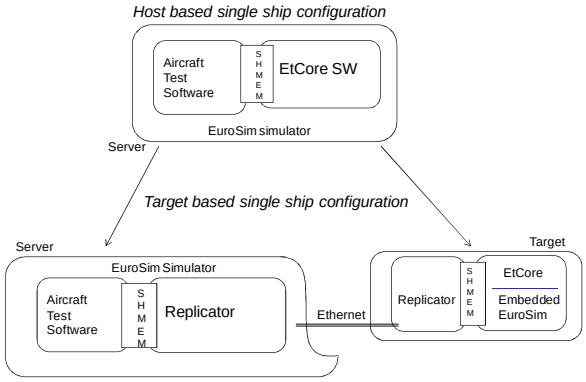

Figure 8 Transition from host to target simulator

As shown in Figure 8 the replicator component replaces the EtCore SW on the host and the Aircraft Test SW on the target. The two replicator components cooperate over Ethernet to create a data mirror between the two shared memory modules. Note that this is a software solution that could be easier implemented with reflective memory cards; however such card is not supported on the target hardware. The replicator allows the extraction of EtCore SW from a host based simulator configuration to a target solution while still allowing the same test scripts to be used for verification.

Where Figure 8 utilizes the single ship configuration to illustrate the transition from host to target configuration, the most challenging environment is the MultiShip configuration. This final configuration is shown in Figure 9.

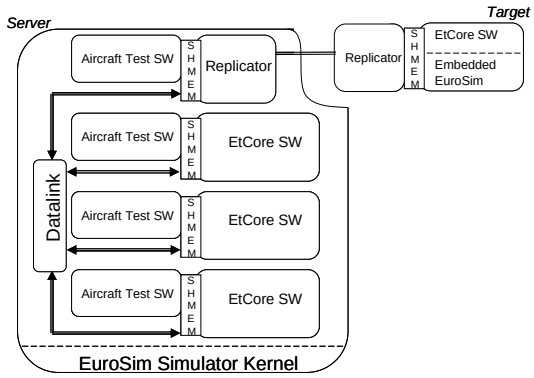

Figure 9 MultiShip verification environment

Additional systems such as real-time visualization have been connected to this configuration for verification purposes. In addition, ideas are discussed whether this configuration can be utilized to support development of ET training scenarios.

8 RESULTS AND EXPERIENCE

The ET project has recently passed its critical design review. Observations that can be made so far are:

- The code generation from UML provided a start with a complete set of empty classes, whereby the EuroSim enhanced code generators not only added the required EuroSim software, but also a makeup of the files compliant with Doxygen and inserting Doxygen todo labels where functionality needed to be added. As a result we have been able from the start to generate detailed design documentation from the software and keep track on todo items in the code via a to-do list in that documentation,
- Once all object instance creations are manually added, the EuroSim dictionary can start the empty software and reflect the object hierarchy that results from instantiation. However, we did find that getting the instance creation correct was a considerable effort and many improvements in EuroSim have subsequently been added to better support this process,
- Both ET development and test system teams have made extensive use of the host based development environments. Model developers thereby use the single core configuration, protocol developers use the multi core solutions and test developers focus on test software and script development. Throughout the day multiple simulations of different configurations run simultaneously on the server and development is on schedule,

- Remarkably few and minor integration issues have been encountered so far as they are detected early in the project and solved by the original author of the component on the fly and in good cooperation. In addition automated nightly build and test allows us to catch problems within a maximum period of one day,
- The embedded simulator technology has been provided by the EuroSim consortium team, and has been integrated in the build process of the ET project. This build process now creates host and target based ET simulator configurations. Utilizing the embedded simulator technology has prevented that ET developers required any target specific hardware, operating system or development environment knowledge,
- Metrics facilities incorporated in EuroSim provide complete feedback on schedule execution statistics and memory consumption, both on the host and the target. Although the ET development as well as performance analysis is ongoing, no indication has been found yet that could raise performance or memory consumption to a high risk item,
- The MultiShip target configuration as shown in Figure 8 has not yet been utilized at the time of writing of this paper. In principle it will work, however, this solution has tighter timing constraints in terms of clock synchronization, which may pose some challenges due to the limited support on the target hardware for clock synchronization,
- Integration of an early test delivery within the laboratory environment of Lockheed has shown that functionality and performance as achieved by Dutch Space and NLR on representable commercial target hardware can be reproduced on the actual flight hardware. The built in metrics capabilities made this an easy comparison.

9 CONCLUSIONS AND RECOMMENDATIONS

In the F-35 MultiShip Embedded Training project we have successfully applied EuroSim in an entire new application. A EuroSim based real-time embedded simulator will become part of an Operational Flight Program of a production fighter aircraft. It is expected that development of Embedded Training will continue to further increase the capabilities of the virtual threats. On the other hand, this accomplishment in the defense domain has proven the readiness of the technology for application in the space domain, and we hope that this leads to new applications of real-time simulation the space domain.

Besides the new type of application, the integration of EuroSim in development and verification of the Embedded Training project life cycle has allowed the project to be efficient and to prevent the classic integration problems normally encountered in large distributed projects. We recommend therefore looking at the desired tool usage in a project, defining the desired tool chains and following a perspective of support for development and test, based on simulation technology.

Finally, new features and interfaces have been added to the EuroSim environment driven by the Embedded Training project. Features as the C++ code generators, embedded simulator technology and extended metrics capability are made available to all EuroSim users as part of the EuroSim 4.2 release. This illustrated the concept of sharing the benefits when (large) projects request new features that subsequently come available as new features to all users of the EuroSim product.

10 ACKNOWLEDGEMENTS

I would like to thank my co-author Erwin Schreutelkamp for his diligent effort in the development of the EuroSim Embedded simulator technology and my NLR and Dutch Space colleagues for their input and support to me in writing this paper. Special thanks go to the Royal Netherlands Air Force for providing the support in developing the ET knowledge and to Lockheed Martin and the F-35 Program Office for their trust in applying this technology.

11 REFERENCES

1. Krijn, R., & Wedzinga, G. (2004). *Development and in-flight demonstration of "E-CATS", an experimental embedded training system for fighter aircraft*, 35th Annual International Symposium of the Society of Flight Test Engineers, Whichita, KA, USA, September 2004.
2. Lemmers, A. (2007). *An Embedded Training Multi-Ship Demonstrator*, I/ITSEC, 2007.