

MODEL-BASED ENGINEERING FOR LASER WEAPONS SYSTEMS

Malcolm Panthaki, Chief Technology Officer & Founder,
Comet Solutions Inc., Albuquerque, New Mexico, USA

Steve Coy, President & Founder,
Timelike Systems LLC, Albuquerque, New Mexico, USA
(formerly Principal Scientist, MZA Associates Corporation, Albuquerque, New Mexico)

ABSTRACT

The Comet Performance Engineering Workspace is an environment that enables integrated, multi-disciplinary modeling and design/simulation process automation. One of the many multi-disciplinary applications of the Comet Workspace is for the integrated Structural, Thermal, Optical Performance (STOP) analysis of complex, multi-disciplinary space systems containing Electro-Optical (EO) sensors such as those which are designed and developed by and for NASA and the Department of Defense. The Comet™ software is currently able to integrate performance simulation data and processes from a wide range of 3-D CAD and analysis software programs including CODE V™ from Optical Research Associates and SigFit™ from Sigmadyne Inc. which are used to simulate the optics performance of EO sensor systems in space-borne applications.

Over the past year, Comet Solutions has been working with MZA Associates of Albuquerque, NM, under a contract with the Air Force Research Laboratories. This funded effort is a "risk reduction effort", to help determine whether the combination of Comet and WaveTrain™, a wave optics systems engineering analysis environment developed and maintained by MZA Associates and used by the Air Force Research Laboratory, will result in an effective Model-Based Engineering (MBE) environment for the analysis and design of laser weapons systems.

This paper will review the results of this effort and future steps.

Keywords: *Model-Based Engineering, laser weapons R&D, integrated simulation environment, systems engineering, multi-physics simulation, multi-fidelity simulation, engineering analysis templates*

1.0 INTRODUCTION AND MOTIVATION

We are currently in the midst of project to extend a commercial multi-physics simulation software framework to support model-based engineering (MBE) of laser weapons systems. MBE has been identified by the National Defense Industry Association (NDIA) [Bergenthal 2011] and by the Office of the Director, Defense Science and Engineering and as one of four potentially “game-changing” technologies that could bring about revolutionary advances across the entire DoD research and development and procurement cycle [Boehm 2010]. To be effective, however, MBE requires robust underlying modeling and simulation technologies capable of modeling *all* the pertinent systems, subsystems, components, effects, and interactions at any level of fidelity that may be required in order to support crucial design decisions at any point in the system development lifecycle. Very often the greatest technical challenges are posed by systems involving interactions that cut across two or more distinct scientific or engineering domains; even in cases where there are excellent tools available for modeling each individual domain, generally none of these domain-specific tools can be used to model the cross-domain interactions.

In the case of laser weapons R&D these tools need to support modeling of systems involving combined interactions among structures, thermal, and optical effects, including ray optics and wave optics, controls, atmospheric effects, target interaction, computational fluid dynamics, and spatiotemporal interactions between lasing light and the laser gain medium. A variety of tools and techniques have been developed that address different parts of this problem - e.g., there are commercial tools like Nastran, Thermal Desktop, and Simulink, to address structures, thermal effects, and controls, as well as other tools developed specifically for this application, such as *WaveTrain*TM for wave optics, *SHaRE*TM and others for intermediate fidelity modeling of direct attack and relay systems, and *GASP*TM used to model the detailed multi-physics inside a laser resonator. In order to fully support MBE for laser weapons we need to bring all these simulation capabilities into a single integrated highly-extensible framework, *utilizing a single unified representation of the system under development, so that design information can be shared across the project, and the engineering teams can implement effective concurrent engineering practices* [Geis, et al. 2009]. This unified data model approach to integration of various physics tools is described in greater detail in Section 2 and is compared with a currently widespread approach called the Federated approach.

To address these challenging requirements, we are developing a “hybrid” software architecture providing two distinct and complementary interfaces for integrating new software components for simulation. One is better suited for smaller scale and/or more tightly coupled components, and the other is better suited for larger scale, more loosely coupled components. The interface is provided by *Tempus*TM, a component-based software framework for multidisciplinary modeling and simulation that provides the foundation for *WaveTrain*, which is a widely used tool for laser weapons design. The second is provided by the *Comet Engineering Workspace*TM, an MBE framework that addresses many of the requirements for optical sensors and laser weapons [Panthaki 2008]. Components integrated using *Tempus* exist and interact with other components within the same computer process (the same address space), with a deterministic and predictable flow of execution. *Tempus* provides an API which is simple, flexible, programmer-friendly, and efficient; this is an appropriate strategy for developing small to moderate scale components, systems, and models, providing highly extensible systems simulation capabilities. Components integrated using the *Comet* interface are implemented as independent computer processes exchanging information by means of serialized data captured and managed within a single integrated data model, with their execution and interactions coordinated by *Comet*. This is a very appropriate strategy for integrating multiple existing software tools developed by multiple organizations within a single unified framework.

Tempus and *Comet* have been designed for maximum flexibility and extensibility to allow functionality to be easily added. A wide variety of software has already been integrated into both frameworks. For example, tools for modeling wave optics (*WaveTrain*), ray optics (*Code V*), structures (*Nastran*, *Abaqus*, *Ansys*, *Adams*), thermal (*Thermal Desktop*), multi-physics coupled solvers (e.g. *COMSOL Multiphysics*, *GASP*), controls (*MATLAB/ Simulink*), meshing, optimization tools, tools for automatic generation of fast “surrogate”

models, and many others. This list is not complete - there are many additional tools that are needed, but we have demonstrated that Tempus and Comet, taken together, are clearly up to the task.

An important question is whether Tempus and Comet can really be integrated – it is no small matter to integrate two component-based frameworks, designed and developed independently, to meet different sets of requirements. To answer this, we recently conducted a “risk reduction experiment” that was recently completed - it was generally successful, with some caveats. We integrated Tempus and WaveTrain into Comet, and then used them, together with a variety of commercial software tools already available through Comet, to demonstrate that the combined toolset can be used on an example problem relevant to laser weapons R&D - a simple telescope with coupled structural/thermal/optical interactions. The example model was based upon *WtDemo*, one of the standard examples included with WaveTrain, but modified slightly, so that it can accept as input a time-varying optical path difference computed outside WaveTrain - using a set of commercial tools from within Comet. The resulting WaveTrain model, *WtDemo_with_OpdMap*, was integrated into a Comet “process”, Figure 1, where other commercial tools were used to respond to updates in the CAD model of the telescope, generate the meshes, model the thermal and structural effects, and input the resulting optical path difference into the WaveTrain model. Figure 2 shows the WaveTrain wave optics model that was invoked through the Comet process, with the OPD map as input. Based on the success of the risk reduction experiment, we are persuaded that the approach we are pursuing, developing a hybrid MBE framework based on Comet and Tempus, is a viable strategy for creating a comprehensive toolset to support MBE of laser weapons systems.

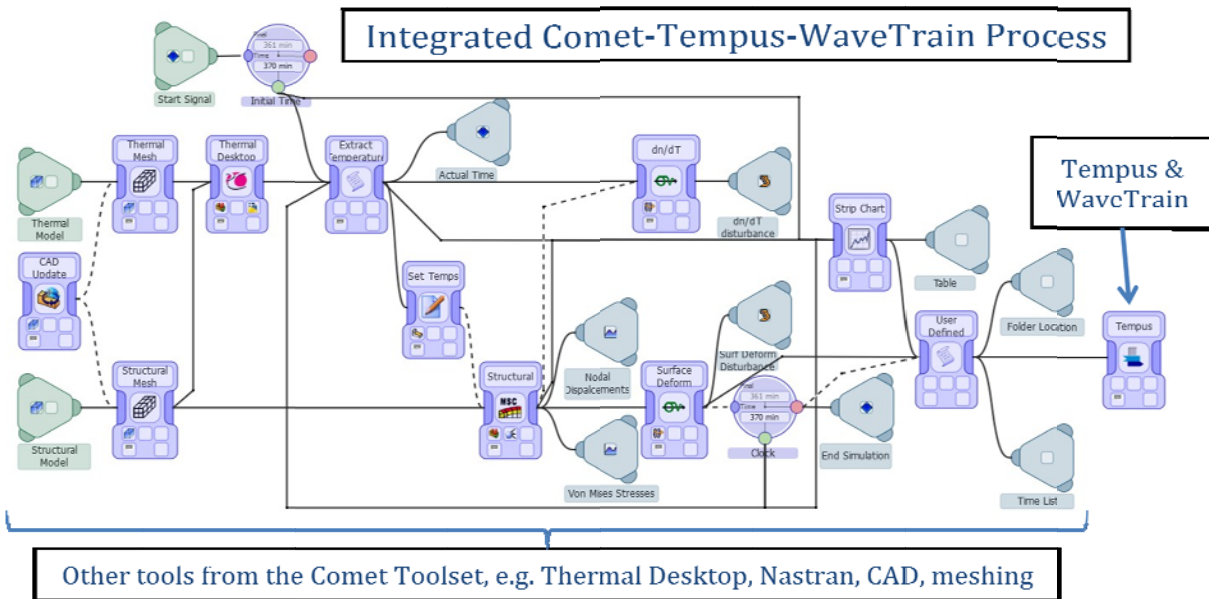


Figure 1. An integrated Comet-Tempus-WaveTrain process, implemented as part of this effort.

2.0 THE ROLE OF COMET

There are two fundamentally different approaches to engineering analysis process capture and reuse for multi-physics and multi-fidelity processes, such as the ones that are required for complex engineered systems such as laser weapons. The most prevalent approach, known as the Federated approach, has "blind" file transfer at its core, with high-level metadata capture. The second approach, used in Comet, adopts a unified data model for capturing processes, models and results. In the next two subsections we will compare and contrast these two approaches.

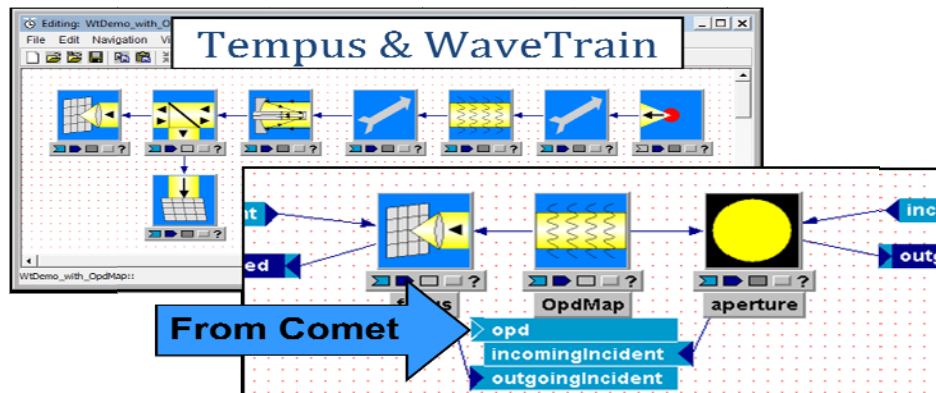


Figure 2. The Tempus/WaveTrain model used in the risk reduction experiment.

2.1 The Federated Approach (used in many other MBE frameworks)

There currently exist a number of software environments that have implemented the Federated approach for engineering analysis process capture and reuse. Some examples are Phoenix Integration's Model Center (http://www.phoenix-int.com/software/phx_modelcenter.php), Dassault Systemes' iSight (<http://www.simulia.com/products/isight.html>), and Esteco's modeFrontier (<http://www.modefrontier.com>). All of the tools listed here are primarily used for what is commonly known as Design Space Exploration - e.g., Parameter Studies, Design of Experiments, and Design Optimization - they have also been referred to as PIDO (Process Integration Design Optimization) tools. As a necessary part of these calculations, the user is required to specify an analysis process that represents the simulation of the product whose design is being explored (the "plant" model), while being compared to the required performance specifications.

The Federated approach treats each calculation tool in the analysis process as a black box. The user has to provide the already generated input files for each tool - these represent the model to be analyzed and the necessary instructions for what analysis is to be performed. The federated tool inserts data into user-specified slots within these input files, runs the underlying tool with the modified files, and extracts the user-specified results from the results files. These values are then utilized in downstream analysis tasks, as needed.

A Federated environment has the following characteristics:

- Setting up connections (or wrappers) to various calculation tools is relatively simple as there is no semantic knowledge of the data that is being sent to and returned from these tools - the tools are treated as black box calculation engines.
- There is no semantic understanding of the data associated with each of the tools. The input and output data are defined and end up in silos, one for each tool specified in the analysis process.
- There is no single, unified representation of the entire engineering model being analyzed. All the data are maintained in their respective silos, in disparate data file formats. The users are responsible for manually consolidating the data, as required.
- As the input files for each tool need to be generated in advance, this approach does not easily support changing the geometry during a set of iterations. Hence, the types of design explorations are limited to low fidelity models - with high fidelity models (where geometry is required) geometric parameters cannot be modified. This is a serious limitation of these environments.
- There is usually no support for managing configurations of the models and results. Each run results in its own separate piles of data, unrelated to previous runs. Phoenix Integration has recently introduced a product called Analysis Library that manages these disparate files better.
- These environments, like most engineering analysis tools today, encourage engineers to continue to work in their respective silos, and to *not* engage in collaborative, concurrent engineering processes. Concurrent engineering processes have been proven to be much more effective [Malcolm Panthaki

2008, Jason Geis, et.al. 2009, David Thomas 2010], especially for the design of complex engineered systems that span multiple disciplines and time scales, such as laser weapons systems.

These are severe limitations and will not support the requirements for laser weapons design.

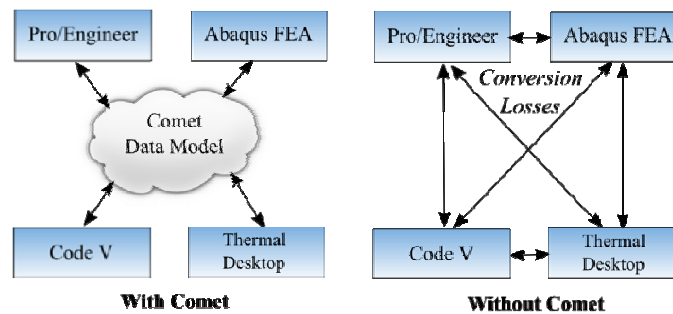
2.2 The Unified Data Model Approach (used in Comet)

We believe that the Unified Data Model approach has many advantages over the Federated Approach, especially for the design of complex systems. The Comet Engineering Workspace adopts this approach.

The heart of Comet is the Comet data model, the Abstract Engineering Model (AEM), which is designed to provide a universal set of data abstractions to represent physical models in a manner that is independent of any particular CAD/CAM/simulation tool. A Comet adaptor for a particular tool must implement two operations - import and export. The import operation converts data from a tool-specific format to the AEM. The export operation converts data from the AEM to a tool-specific format.

The AEM solves two major problems.

1. First, it is scalable and extensible. By using a universal format, a set of n tools only requires n adaptors. Without a universal format, and this hub-and-spoke data architecture, a set of n tools would require n^2 conversion operations, which is clearly not feasible for a large number of tools.
2. Second and more importantly, the AEM eliminates the information losses often associated with the conversion of data from one format to another. Comet stores all of the properties associated with a model, including thermal, structural, and optical properties, in a single unified format. When an optical analysis tool like Code V is invoked, its adaptor extracts only the optical properties, and passes them to Code V. When multiple analysis tools need a particular datum, the single stored value is used for all of them, ensuring data integrity across the various tools.



The Comet workspace has the following characteristics:

- Is built on a robust and uniquely extensible “functional data model” (the Abstract Engineering Model™ or AEM) that can rapidly support evaluation of new subsystem and component types and alternative design configurations. It supports analysis of models at mixed levels of fidelity, as well as complex cross-domain effects and interactions (e.g., structural, thermal, optics, durability, turbulent airflow, etc), across product engineering disciplines (costing, mechanical/MCAD, electronics/ECAD, embedded software, controls, manufacturing).

The AEM unifies all the engineering analysis data from concept models through full 3-D CAD/mesh-based models, with support for all levels of model fidelity and the ability to easily extend the data model to support all required physics. The main features of the data model are:

- is highly extensible and largely independent of the data formats of the underlying codes used for the engineering calculations,
- supports the ability to manage multiple representations of each system component, each used by different analysis codes, often at different levels of fidelity,
- is component-centric, not geometry-centric - supports systems models, and

Model-Based Engineering for Laser Weapons Systems

- captures all data including their units and automatically manages all unit transforms and coordinate system transforms when communicating with the various external tools.
- Facilitates the use of all the "blessed" engineering tools, spanning from spreadsheets, MATLAB and special-purpose, in-house calculation codes to full 3-D CAD and mesh-based codes. The data model is easily extensible to support new data, new physics domains and new types of systems components, as required. There are a number of existing engineering tools that implement a variety of models across various physics domains and levels of model fidelity. It is critical that the MBE environment be able to support any of the tools that might be required for the analysis of the complex system being designed.

Comet integrates with commercially available domain-specific modeling and simulation tools such as various 3-D CAD and Finite Element Analysis (FEA) tools, Excel and MATLAB/Simulink used for new product design and development. As part of this development effort, a first version of an adaptor was created for WaveTrain/Tempus, an Air Force Research Labs (AFRL) funded wave optics systems environment. Other AFRL-specific systems engineering frameworks/tools such as *SCALE* and *ABLPAT*, developed by the primary author, could also be integrated into the Comet environment in the future, using adaptors.

- Enables tool-neutral process automation via GUI-based templates that capture engineering best-practices for reuse across multiple engineering disciplines, across programs over time and with support for external suppliers. These templates are created by the domain experts and can be easily used by systems engineers and non-domain experts.
- Supports the need for mixed fidelity simulation. An engineering analysis process in Comet can be defined to access any of the calculation tools, both commercial and in-house. Each external analysis tool can utilize one of various representations of the system, at the desired level of fidelity. With Comet, the engineers are no longer constrained to pick either low fidelity or high fidelity calculations, but can select the appropriate level of fidelity required to get the desired system performance answers at the current stage of the design process.
- Enables bi-directional flow of data between concept and detailed models, integrating product and process data across multiple physics domains, from low to high fidelity stages of the process.
- Comet templates work effectively across a wide range of different conceptual design topologies or geometry configurations enabling engineers to easily explore the concept design space for rapid analysis of trade spaces and determination of the complexity/risk for each major design alternative. Comet's Abstract Modeling technology is unique and allows engineers to create templates that are independent of the geometry and configuration of the concept being analyzed.
- Enables project team collaboration (including suppliers) and decision-making based on continuous access to key design and simulation performance metrics. These metrics are compared against current system requirements via an interactive, project-centric Decision Dashboard that reflects the current state of the system's cross-domain design performance, throughout the entire product development lifecycle.
- Manages the frequent, work-in-progress model and results variations easily and enables rapid navigation to all project configurations and performance simulation results generated across physics domains and across time for ensuring the data (design and results) pedigree and data traceability (audit trail) through the entire process. This configuration management support is provided with very little overhead to the day-to-day simulation tasks performed by the engineers.

The Comet workspace is a general-purpose environment for MBE, but it does not currently integrate all of the simulation tools required for laser weapons systems analysis and design. However, Comet Solutions has already demonstrated, with major U.S. Aerospace and Defense (A&D) customers over the past several years, that the current implementation of the Comet MBE software environment can be extended to address a number of the fundamental “knowledge and process disconnects” in the classical Systems Engineering “V” approach. These customers have also demonstrated that Comet addresses the knowledge reuse and product team collaboration roadblocks that exist in the current sequential “functional domain silo” product development approach that is still so prevalent throughout the current U.S A&D industry. Several application white papers written by Comet users at both the Aerospace Corporation and General Dynamics Land Systems demonstrate the expected impact from extending and implementing the core capabilities of the Comet environment as a next generation model-based systems engineering platform for new product design and development- see www.cometsolutions.com.

Below is an image of the GUI of the Comet workspace, with its key components annotated:

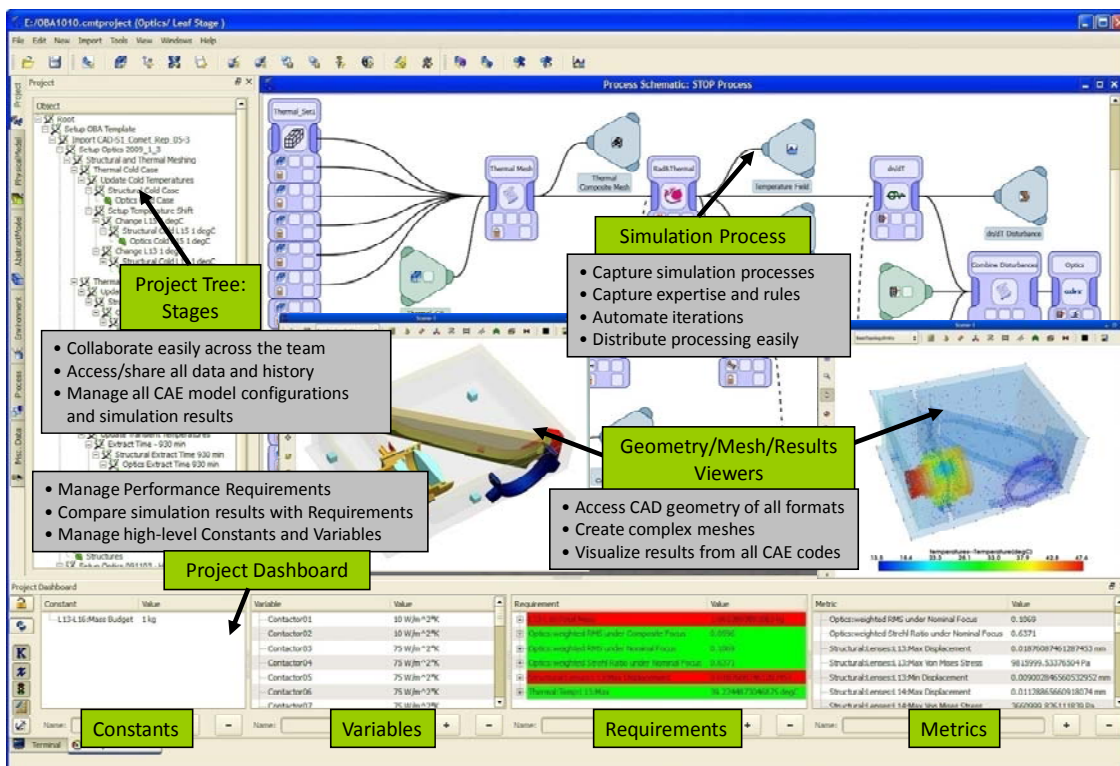


Figure 3. Comet GUI showing key elements such as the Project, a Process and the Dashboard.

2.3 Comet's "Intelligent Templates"

In order to capture the engineering knowledge and best practices of domain experts, the Comet environment provides the ability to create “intelligent” process templates that can grow and evolve over the course of a product development program. The variations of these templates over the life of a project are automatically archived within the Comet project tree.

These process templates can be defined to execute a number of Tasks or steps that involve single or multiple physics domains using different design and simulation tools at different levels of model fidelity. Each Task in a process will often use an external tool (such as WaveTrain, MATLAB, Code V or Abaqus) to perform its calculations. The input files for each external tool are generated each time the Task is executed - this is

contrary to what happens with the Federated approach, where the input files need to be predefined. *Since the inputs are generated each time, the geometry and/or configuration of the model could be changing for each run of the Task.* The related domain expert creates a Task (e.g., the Nastran expert creates and owns the Nastran Tasks) and puts the relevant rules into it for knowledge capture and reuse purposes - most of this is done graphically within the Comet environment, making it easy to create, enhance and maintain these templates.

The processes defined in Comet are captured as an integral part of the data model, thus ensuring rigorous associativity between the process being executed and the underlying models, environments and results that are generated with each of the calculation tools. The use of a common abstract data definition of the product ensures that consistent data are used across the various analysis codes (contrary to what happens in the Federated approach), keeping the models consistent and always in synch for each domain or discipline working within the same Comet project. This has been identified as a key requirement for effective Model Based Engineering [Boehm 2010, Bergenthal 2011].

A key enabler for this multi-physics and cross-disciplinary process execution within Comet is the concept of “adaptors”. Each distinct type of Task (e.g., a WaveTrain Task or a Nastran Task) requires an adaptor - adaptors are typically bi-directional and perform the following operations:

- take the appropriate design data and analysis information that is resident in the Comet functional product definition and create compatible input files for the required underlying external tool,
- execute the software tool with the appropriate input files, either on the local machine or a remote machine,
- capture the desired results output from the files generated by the external tool,
- save the original results files back into the Comet project (either on the local disk or a remote, shared drive), and
- display the values of the pre-defined design variables vs. design requirements and pass/fail metrics in the Comet Decision Dashboard.

An adaptor essentially generates the required multi-physics models and results for a domain-specific tool, every time it is encountered within a Comet process template. This is a “hub and spoke” unified data model approach, contrary to the file-centric (black-box) transfer of data that occurs in the Federated approach. The input and output files associated with each tool are available to the expert, as required.

Comet currently provides adaptors for a number of CAD and analysis tools such as Pro/Engineer, NX CAD, and SolidWorks; Excel and MATLAB; Thermal Desktop, MSC/Adams, MSC/Nastran, Ansys, Abaqus, SigFit and CODE V - across the thermal, structural and optics analysis domains.

3.0 THE ROLE OF WAVE TRAIN AND TEMPUS

In addition to the set of modeling tools for which Comet adaptors already exist, there are a number of other modeling tools that are needed to support MBE for laser weapons R&D. These include wave optics modeling tools such as WaveTrain, WaveProp, ACS, and OSSIM, various intermediate fidelity tools for modeling direct HEL systems performance such as *SHaRETM*, *SCALE*, *HELCOMES*, *HELMUSE*, *HELSEEM*, *HELLEOS* and other specialized modeling tools. such as *GASPTM*, which can be used to model the detailed multi-physics phenomena inside a laser resonator. To use these tools in combination with the other tools within the Comet environment, adaptors will need to be developed for these tools.

Implementing a Comet adaptor for a new modeling tool can be relatively easy or quite challenging, depending on the complexity of the physics/engineering information to be exchanged, and also the degree to which the requirements for the new adaptor resemble those for previously implemented adaptors. For example, the first time a Comet adaptor was implemented for a ray-optics optical design tool, Code V, a large portion of the

effort went into defining the relationships and commonalities between the information used in that domain and that used in others, such as structures and thermal modeling. However, now, if we were to implement an adaptor for another ray-optics optical design tool, such as ZEMAX, that part of the effort is reused, and the part that remains to be done is relatively straightforward, following the same pattern as the Code V adaptor.

Implementing Comet adaptors for these additional modeling tools would represent a significant investment of time and resources, and therefore entails a significant risk. To reduce this risk, the Air Force Research Labs (AFRL) decided to carry out a risk reduction experiment, in which we implemented a simplified Comet adaptor for one of the required modeling tools, WaveTrain. We chose WaveTrain for two reasons: first, it is widely used in laser weapons R&D, and second, it poses an especially stressing test case - if there were going to be any showstoppers, the AFRL wanted to discover them sooner rather than later. Our reasoning was that if we can implement a Comet adaptor for WaveTrain, then it should be possible to implement adaptors for any of the other required modeling tools.

There were three main reasons we expected that WaveTrain would prove to be an especially stressing test case:

1. It was the first wave optics modeling tool to be integrated into Comet, so we had to deal with a new modeling domain, with no prior examples to follow.
2. This was the very first adaptor implemented by anyone outside of Comet.
3. It is not simply a stand-alone modeling tool for wave optics, but is an extension of a general purpose multi-disciplinary modeling and simulation framework called Tempus. The additional capabilities and flexibility provided by the Tempus framework contribute a great deal to the usefulness of WaveTrain in laser weapons R&D, so it is important that the adaptor should provide access to not only WaveTrain's wave optics capabilities, but all of Tempus' multi-disciplinary modeling and simulation capabilities as well. This considerably increases the semantic complexity of the engineering information to be exchanged, and therefore makes the design and implementation of the adaptor that much more challenging. At the same time it will make the resulting integrated tool that much more useful.

To illustrate how WaveTrain and Tempus work, consider the following example: Figure 4 shows the top level block diagram for the model of a laser beam projection system involving closed loop tracking and adaptive optics, using one or more illuminator lasers (e.g. a "TILL" and a "BILL") and a weapons laser. Both the platform and target may be moving, and the effects of propagation through distributed turbulence are included.

The above example also shows how the usefulness of WaveTrain is augmented by the general purpose modeling and simulation capabilities provided by Tempus. All but two of the connections shown represent optical interfaces modeled at high fidelity (wave optics) using WaveTrain, but the two connections at the top and bottom of the diagram (with right angle bends) represent sensor outputs and control signals in the track and AO loops. These interactions are much simpler, but they are crucial to understanding and predicting the behavior of the system. However they don't involve wave optics, so they don't require WaveTrain. Instead they can be modeled using the basic mathematics capabilities of Tempus directly, or by using another modeling tool, such as Simulink, which Tempus can interface with.

Once we had chosen WaveTrain/Tempus for the risk reduction experiment, it soon became apparent that there were two possible implementation strategies for the adaptor; one more straightforward, but in the long run more limiting, and the other clearly superior in the long run, but requiring a greater investment up front. We chose the first approach, because that would suffice for the purposes of revealing potential issues, and we were confident that we could complete the implementation within the allotted time and resource constraints - this was not the case for the second approach. The second approach will be used in the next phase of the development effort.

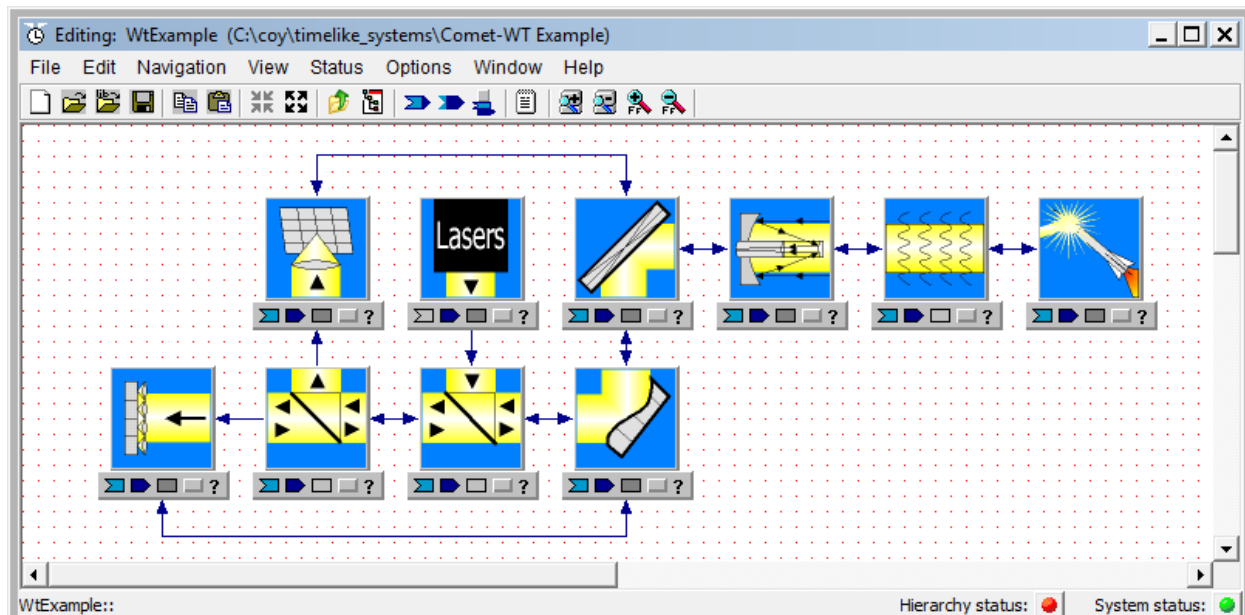


Figure 4. Example optical system modeled in WaveTrain and Tempus.

As noted above, besides WaveTrain, there are many other modeling tools commonly used in laser weapons R&D. All of these tools must eventually be integrated into the Comet/Tempus MBE framework. The resources required for each adaptor will vary, depending on the complexity of the data that needs to be supported.

4.0 RESULTS OF THE RISK REDUCTION EFFORT

The risk reduction effort was completed in March 2011 and spanned a period of eight months. This Air Force Research Labs-funded project included the implementation of a simplified Comet adaptor for Tempus, making it possible to use Tempus and WaveTrain in combination with other tools already in the Comet framework.

Using the new adaptor, we implemented a model of a telescope system which incorporated a tempus/WaveTrain wave optics model, a Nastran structures model, a Thermal Desktop structures model, as well as automated meshing tools, all integrated into a single Comet Process, as shown in Figure 1. The successful completion of this test case demonstrates that WaveTrain can in fact be integrated into the Comet framework, and because WaveTrain was chosen to provide an especially stressing test case, this gives us high confidence that the various other modeling tools used in laser weapons R&D can also be integrated. Furthermore, once the adaptor was completed, we found it very easy to integrate WaveTrain with other modeling tools within Comet. This is crucial, because ease of integration is one of the principal advantages the Comet framework is designed to provide.

In the course of implementing the Comet adaptor for WaveTrain we assessed some of the strengths and limitations of the Comet framework as a platform for laser weapons R&D. This assessment was carried out not by Comet Solutions, but by MZA Associates Corporation, a contractor chosen by the Air Force Research Laboratory to do this independent evaluation.

MZA's assessment of the strengths and limitations of Comet for this application are summarized in section 4.1. As part of this effort, MZA also evaluated the strengths and limitations of the Tempus and WaveTrain software for use in this context, as described in Section 4.2.

4.1 Strengths and Limitations of Comet's Abstract Engineering Model (AEM)

Comet provides several capabilities designed to allow CAD/CAM and simulation tools to interoperate easily. The most important characteristics, strengths and benefits of Comet are detailed in Section 2.2 of this paper.

A goal of this risk reduction effort was to also uncover the limitations of the Comet approach, and in particular, its key technical innovation, the AEM.

Universal Data Representation. While this is the core strength of Comet's data model, developing a single, integrated data representation for all CAE data is not an easy task and has certain limitations. These are not weaknesses of the AEM, but issues that are inherently difficult to solve from a data modeling perspective.

A fundamental problem is that there are many different ways to represent a component, and it is not always possible to come up with a single "universal" representation that works for all tools. Over the course of its development, Comet's software engineers have been able to identify certain representations (data types) that are common to several different tools. For example, rigid body dynamics codes typically model a physical component as a set of rigid parts which are connected by joints, where the joints constrain the motion of the parts. There are several different kinds of joint, such as hinges, spherical joints, universal joints, etc. The set of joint types is relatively well understood, and does not vary much between different tools, so it is possible to construct a universal representations of these joint types that include all of the necessary properties. Similarly, finite element analysis tools deal with meshes, so the AEM has a number of built-in facilities for capturing and manipulating meshes.

However, the "universal" representation is at best an approximation, which has been tailored to the needs of whatever set of tools is currently supported by Comet. When Comet adds support for a new tool, there is no guarantee that the existing Comet data types will be sufficiently "universal" to handle the capabilities of the new tool. As a hypothetical example, assume that a more sophisticated dynamics tool has been released, which introduces flexible joints that allow a limited degree of elastic, damped deformation. The Comet data model would need to be extended to support such a tool, because the existing rigid joint types would not be sufficient to describe an elastic model. WaveTrain itself is actually a concrete example of this principle – the existing representation that Comet uses for optical components will have to be extended to support "similar" components that are represented in WaveTrain.

To deal effectively with this inherent aspect of the CAE data model, the AEM has been designed from the ground up to be easily and highly extensible, with a core, "skeleton" of data types and the ability, when required, to specialize the leaf types of the skeleton, as needed. The AEM is constantly being refined and extended as support for new tools is added. As newer versions of the data model are released, users projects may need to be "translated", to allow them to be used in the latest versions of the software. With this translation process, backward compatibility of the users' data is ensured.

Multiple Component Representations. As described above, the extensibility mechanism in the AEM is the ability to specialize leaf data types. The issue is that for certain types of data, a single representation does not suffice. For example, the shape of a parabolic mirror in an optical system can be described both in terms of a radius and focal point, as well as by using a CAD shape representation in the form of NURBS. The former representation might be used by a ray-tracing system such as CODE V, and the latter might be meshed and the mesh then used by multiple finite-element analysis tools. In this case, there is no "universal" representation for the parabolic mirror component. It is possible to convert from a radius/focal-point representation to a mesh, but at the cost of lowering the level of abstraction and possible to fit a parabola to a mesh, but at the cost of fidelity – there is no one universal representation for all cases.

Comet resolves this issue by allowing each component in the model to have multiple representations. The appropriate representation is then selected when the model is exported to a particular tool. In the case of a parabolic mirror, Comet would maintain the CAD part, the associated meshes (there can be multiple meshes per part), and the "optical component representation" that holds the radius/focal point information. One representation would be used for ray-tracing, and the other for finite-element analysis.

Supporting multiple representations is a good compromise, but it does cause some potential problems. First, Comet makes no attempt to automatically verify that the two representations are equivalent, e.g., that the mesh actually has the radius and focal point that have been claimed for it. Second, using two representations may prevent the calculations done by one tool from being used in another, e.g., deformations calculated by FEA will not be fed into the ray tracer if the ray tracer does not use the mesh; some other tool needs to "bridge" these representations - in this case, that tool is SigFit from Sigmadyne.

While Comet does a good job of trying to accommodate many tools, but it is not a magic bullet. There are cases where tools may simply differ too much to be easily integrated. The attempt to integrate WaveTrain into Comet encountered some of these difficulties - more work needs to be done in the next phase to import into Comet both the component type data and the particular system being imported.

Non-standard type system. In our view, the Comet data model suffers from another weakness, which is that it is based on a non-standard type system. In computer science, there are two ways of organizing data that have achieved widespread use: data types for programming languages and relational databases. Over the past few decades, programming languages have converged on a fairly standard set of fundamental data types such as scalars, records, arrays, lists, enumerations, variants, and object-oriented classes. Relational databases present an alternate mechanism, based on entity-relationship (E/R) diagrams, which are optimized for queries on large data sets. Comet uses a hybrid type system, along with some extensions of its own.

Comet data types bear some resemblance to the data structures used by programming languages, but introduces its own terminology. A "SingleValuedDynamicEntity" is a scalar quantity with units (a crucial type for engineering data and not one that is supported by standard type systems), while an "AggregateDynamicEntity" is anything that is not a scalar, such as a "DynamicEntityMapType", which is a record. Like object-oriented languages, Comet supports inheritance between AggregateDynamicEntities, and it uses inheritance extensively in its efforts to build a universal representation for physical models.

AggregateDynamicEntities also differ from object-oriented classes (or indeed, any other data types that we are familiar with) in several ways. First, the data members associated with the entity are not defined as part of the data type, but are defined in a separate "template" definition, another Comet-specific term. Templates have their own inheritance hierarchy, which is usually, but not necessarily, the same as that of the entities that they define. This disassembly of the data associated with a type, while non-standard, has numerous advantages such as reuse of data and extensibility of the type system.

Comet distinguishes between properties of an entity (scalars) and the children of an entity, which can be other AggregateDynamicEntities. Child relationships must be declared with "link" declarations that specify the cardinality of the relationship, resembling the E/R diagrams used in relational databases.

A minor annoyance is that the definition of a new data type is far more complicated than it would be in a typical programming language, requiring modification of around 5 different files. There is clearly room for streamlining this process.

The Comet type system does have some advantages. The use of links with cardinalities, rather than pointers, is a clear advantage over the type systems commonly used in programming languages, while inheritance gives the system much more flexibility than a relational database. However, the combination of unusual terminology, coupled with unusual design decisions, make the type system difficult to understand. Moreover, the template mechanism seems ad-hoc. We feel that the benefits of the Comet type system could have been achieved without deviating quite so far from accepted practice.

It is important to note that the details of the Comet type system are not visible to most users. Only Comet developers and adaptor developers need to interact with the type system directly. From a user's standpoint, these criticisms are largely irrelevant as the type system supports the required functionality.

4.2 Strengths and Limitations of Tempus and WaveTrain

The main advantages of the Tempus modeling and simulation framework are its exceptional flexibility and extensibility, which make it possible to model a very wide variety of component and effects models all within a

single unified framework. This flexibility and extensibility is what made it possible to implement WaveTrain, a specialized tool for high fidelity (wave optics) modeling of advanced optical systems, within the general purpose Tempus framework, thus making it easy to use WaveTrain in combination with other kinds of components and effects models, e.g. structures, thermal effects, and controls.

The main disadvantages of Tempus and WaveTrain relate to their implementation strategy. It relies upon auto-generation of C++ code, which must be compiled and linked before a simulation can be executed. When Tempus and WaveTrain were originally implemented this was the only viable strategy, consistent with the technical requirements, but this strategy has a number of significant disadvantages.

1. In order to install Tempus and WaveTrain it is necessary to install a complete make/build environment (Microsoft Visual Studio). This is much more expensive, burdensome, and error-prone than an ordinary software installation.
2. The primary graphical user interface (GUI) components provided by Tempus and WaveTrain support inspection and editing only of the data structures that define the system model in a static sense, for use in code generation. No mechanisms are provided to inspect the system state as it changes over the course of a simulation run. This means, for example, that when errors or unexpected results are encountered, users cannot use the block diagram editor to investigate the source of the problem; instead they need to be able to debug the code at the source code level.
3. In the present implementation, the major functional capabilities provided by Tempus and WaveTrain, such as code generation, can only be accessed via the GUI. In order to implement the Tempus/WaveTrain adaptor for Comet we needed to be able to access these capabilities independently of the GUI, so that Tempus/WaveTrain functionality could be incorporated into a Comet "process" and run in batch (non-GUI) mode. This required a considerable amount of work, possibly the majority of the effort expended in the course of this risk reduction exercise, and could have been avoided if the architecture had clearly separated the GUI from the underlying software that implements the various components in the block diagram.

These limitations of Tempus and WaveTrain have been well known for many years. Early on, most fell into the category of "necessary evils", because they were consequences of the implementation strategy chosen, and at the time the tools were designed that was the only viable implementation strategy. However the process of carrying out the risk reduction exercise has given us a new appreciation of just how severe those limitations are. In addition, subsequent advances in certain technologies, in particular C#, .NET, and Java, now support better implementation strategies that would make it possible to eliminate most of these limitations.

5.0 CONCLUSIONS AND FUTURE WORK

With funding from the Air Force Research Laboratory, MZA Associates Corporation of Albuquerque, New Mexico, performed an independent assessment of the Comet Model-Based Engineering (MBE) framework for laser weapons R&D - author Steve Coy led this effort. MZA's conclusions are as follows:

The Comet Data Model (AEM) is the heart of Comet. It is designed to provide a single unified representation of all system data, independent of any particular modeling tool. The AEM enables the user to use different tools to work more effectively together. We have expressed some concerns about the design of the type system, and about the restriction to directed acyclic data structures. However, the Comet AEM is a sophisticated piece of software engineering, and on the whole it works as advertised, which is an impressive accomplishment. Furthermore, we believe that most of its present limitations are largely a reflection of the fact that Comet is breaking entirely new ground and continues to be enhanced, as needed.

In seeking to design a single common representation for all engineering analysis data, Comet has tackled an enormously difficult problem, one which, in our view, represents a critical enabling technology for effective MBE of complex systems, such as laser weapons systems. MBE for such systems generally requires the use of a relatively large number of disparate engineering analysis tools, across various physics and levels of model fidelity. Ordinarily, each of these tools employs its own scheme for representing its data, which makes it

difficult to share data among different modeling tools, even when the underlying information may be the same. This problem is typically addressed by implementing separate interfaces between individual pairs of tools (see Section 2.1, the Federated Approach), but this approach does not provide a systems view of the engineering product, scales poorly, is difficult to maintain, and hides the underlying information from the user, rather than making the information explicit and accessible.

Comet provides the following significant innovations that enable it to seamlessly coordinate the activities of many different CAD, CAE, and simulation tools.

1. The Comet AEM provides a universal representation of a physical model that allows the different tools to share data, without the losses and complexities of data conversions.
2. Comet processes provide a convenient mechanism for invoking and scripting the various tools in a visual environment - a visual programming language for engineering analysis.
3. Projects, composed of stages, provide a form of version control with special support for exploring various engineering alternatives, while keeping track of the evolution of the design through the life of the project.
4. Finally, Comet has been specifically designed for extensibility, so that users can integrate additional modeling functionality as needed to support their particular application. This is critical for applications such as laser weapons R&D, which rely heavily on the use of domain-specific modeling tools like WaveTrain.

We are not aware of any other commercially available tools for model-based engineering that can offer these advantages. Although we have identified certain limitations of the Comet framework, as discussed in Section 4.1, we conclude that Comet offers a clear and obvious benefit as an environment that facilitates MBE for complex systems such as laser weapons.

Based on the results of the risk reduction experiment, we are now proposing to go ahead with the development of a comprehensive model-based engineering environment for laser weapons systems, based on the Comet COTS MBE framework, and implementing more complete Comet adaptors for Tempus, WaveTrain, and eventually other modeling tools used in laser weapons R&D. We will then pursue one or more pilot projects, where the new MBE framework will be tested, with active cooperation from the project engineers to help us identify ways in which the environment could be improved to better meet their needs.

ACKNOWLEDGEMENTS

The work presented in this paper was funded by the Air Force Research Laboratory under a Small Business Innovative Research grant. The authors would like to express their appreciation for the help, guidance and support of Dr. Dan Marker, AFRL/RDSE.

REFERENCES

[Bergenthal 2011] Jeff Bergenthal, Subcommittee Lead, "Final Report of the Model Based Engineering (MBE) Subcommittee," NDIA Systems Engineering Division, M&S Committee, 10 February 2011.

[Boehm 2011] Barry Boehm, Principal Investigator, "Systems 2020 Strategic Initiative," Final Technical Report SERC-2010-TR-009, Systems Engineering Research Center, University of Southern California, August 29, 2010.

[Thomas 2010] David Thomas, "Causes of catastrophic failure in complex systems," Proc. of SPIE Vol. 7796, 77960K, 2010.

[Geis 2009] Jason Geis, Jeff Lang, Leslie Peterson, Francisco Roybal, David Thomas, "Collaborative design and analysis of Electro-Optical Sensors," Proc. of SPIE Vol. 7427, 74270H, 2009.

[Panthaki 2008] Malcolm Panthaki, "Concurrent Engineering to the Rescue: Do We Have the Software Tools to Support It?" Proc. of SPIE Vol. 7071, 2008.