THE APPLICATION OF OBJECT TECHNOLOGY IN AASHTO'S NEW BRIDGE LOAD RATING AND DESIGN SYSTEMS

Paul D. Thompson, Consultant, USA
James A. Duray, Michael Baker Jr., Inc., USA
Jay A. Puckett, BridgeTech, Inc. and University of Wyoming, USA
Jeffrey J. Campbell, Michael Baker Jr., Inc., USA

Abstract

Funded by the contributions of over 30 states and the US Federal Highway Administration, the American Association of State Highway and Transportation Officials (AASHTO) is nearing completion of a three-year project to develop a pair of related systems for bridge load rating and design, known as Virtis™ and Opis™, respectively. A consulting team led by Michael Baker Jr., Inc. is developing the systems.

Bridge load rating and design are two very different business processes, yet they have much in common, including similar philosophical and mathematical approaches, and similar data. The challenge of the Virtis/Opis project has been to satisfy the complex requirements of each separate system, while saving AASHTO millions of dollars in development and maintenance costs by reusing substantial quantities of software between the two systems. An object-oriented design and development approach has been used to help in meeting these challenges.

Keywords: structures, bridge load rating, bridge design, object-oriented

1. Introduction

The American Association of State Highway and Transportation Officials (AASHTO) is nearing completion of a three-year project to develop a new combined set of bridge load rating and design tools, known as Virtis and Opis, respectively. The project is funded by the contributions of over thirty states and the Federal Highway Administration. Direction of the project is entrusted in two Task Forces, one concerned with load rating functionality and the other concerned with design functionality, each having representatives from five states. A consulting team led by Michael Baker Jr., Inc. is developing the systems, with subcontract support from BridgeTech, Inc., Paul D. Thompson, and Modjeski & Masters, Inc.

Load rating is the process of ascertaining the capacity of each existing bridge in the nation's diverse inventory in order to inform the public of load limits and to develop freight policies and truck routes. Such information is completely lacking for more than two-thirds of the nation's 600,000 bridges. In addition, load rating serves an important operational function in the processing of overload permit applications, to determine whether a specific load can safely travel over a given route. Speed and accuracy are very important in this type of decision making.

Bridge design is a creative process of finding a configuration of future structural components that can satisfy a set of functional and aesthetic requirements at minimum initial and life-cycle cost. Like any complex design process, bridge design depends on the ability to create reasonable potential solutions, evaluate them, and improve upon them until all requirements are met.

Bridge load rating and design are two very different business processes, yet they have much in common. Both have similar philosophical and mathematical approaches to structural analysis, load behavior, and resistance actions, and both require very similar sets of very detailed data describing each bridge. Both activities benefit from visualization.

The twin challenges of the Virtis and Opis projects are to satisfy the very complex analytical, graphical, and data management requirements of each separate system, while saving AASHTO millions of dollars in development and maintenance costs by reusing substantial quantities of software between the two systems. An object-oriented design and development approach has been used to help in meeting these challenges.

Since the structural calculations needed for load rating are already well-understood, the Virtis project has made the decision to reuse an existing software package, called BRASS[1], for this purpose. BRASS, like all industrial-strength load rating packages in common use today, focuses on each bridge individually and does not have database management or graphical features suitable for efficient management of large inventories of bridges. The Virtis project, therefore, has concentrated on building an integrated database and the graphical tools required for visualizing the data.

A great advantage often touted for object-oriented development is the stability of the domain model as a description of the permanent and important aspects of the structure of the problem

---

[1] Bridge Rating and Analysis of Structural Systems (BRASS) is a bridge girderline analysis, design, and rating program that has been in use for many years. It was recently rewritten for the AASHTO LRFD Bridge Design Specifications. It is developed and maintained by the Wyoming Department of Transportation (US).

domain supported by the software. For Opis, the ability to use the same domain model as Virtis has presented the opportunity to build much of the kernel of a new bridge design system by reusing major components of the load rating system. This software reuse was made possible by designing the domain model to accommodate both sets of requirements simultaneously.

2. Background and objectives

The process of creating Virtis and Opis began in 1995. Both systems emerged from a recognized need to retire existing, older generation software packages and to adopt a more modern architecture with an object-oriented structure, a well-organized multi-user database, and a graphic user interface. Several significant drivers led AASHTO to this decision, including software obsolescence and accompanying high maintenance costs, usability considerations, and the need to satisfy new Federal mandates regarding the load rating process.

The AASHTO Bridge Analysis and Rating System (BARS) has been in use in over 25 states since the early 1970's. This system is built using vintage FORTRAN and is difficult and expensive to maintain. It is also difficult to update the software to keep it current with annual bridge design and rating specification changes. Both BARS and the AASHTO Bridge Design System (BDS) use a structured, formatted input, command language input or text based menu system to allow the user to interact with the program. In today's modern bridge office environment, designers and bridge raters demand a graphical and intuitive software interface in order to work efficiently and productively. Neither BARS nor BDS enabled the user to interactively enter and review input data and output results in a friendly, productive manner. When the Federal Highway Administration (FHWA) mandated that the states begin to use the Load Factor method (LFD) instead of earlier methods for load rating, this implied major changes to the AASHTO systems, especially in the amount and type of data to be collected. This made it necessary to decide whether to attempt to update the old software, or to take the opportunity to build a more modern system.

A modern system architecture was proposed to support both the rating and design processes of the bridge engineer. A Functional Requirements document was written to address the bridge engineers' business processes, functional and analytical requirements, system architectural requirements and the major design elements of the system. Following this step, a functional prototype of the new system was developed. Based on comments on the prototype system the final system was designed and is currently under development. Virtis and Opis will be delivered in modules that address the prioritized needs of the states' bridge design and rating engineers. Software to support the most common bridge types will be delivered first followed by modules to support the lesser common structures until the system is in place to design and load rate most structure types and configurations representing approximately 80% to 85% of the nation's bridges.

For Virtis and Opis, the primary goal has always been, and continues to be, to develop the most accurate, complete, rigorous, reliable, and durable software package for bridge load rating and design, respectively, according to state and Federal standards. The users, Task Forces, and developers have recognized that a state-of-the-art approach to standards, technology, licensing, and development are necessary to accomplish this goal. The primary objectives which have been defined in response to this goal are:

- Adopt an object-based architecture to enhance the reliability, efficiency, and maintainability of the system.

- Maximize designer creativity and productivity, and the economy of new bridge designs, by facilitating the swift creation and evaluation of a large number of design alternatives.

- Maximize load rating engineer productivity in reducing the huge backlog of unrated bridges in the US, by speeding data entry and analysis.

- Broaden the business process models of load rating and design for better integration with other systems and related business processes. This includes the more effective use of load rating data in related systems such as routing and permit management, and includes the support of management use of load rating data at a more aggregate level of analysis than the single bridge.

- Satisfy Federal requirements to rate bridges using the Load Factor method, unless the requirement is changed to support the new Load and Resistance Factor specification.

- Provide a high degree of user-friendliness for both rater and designer, recognizing the differences between these disciplines in their needs.

The Virtis/Opis project is an ambitious undertaking that promises to deliver state-of-the-art systems and high value to its licensees.

3. System overview

The Virtis/Opis architecture is comprised of four primary layers shown below in Fig. 1: User Interface, Domain, Data Management, and Database. Each layer has responsibilities for communication and data exchange with adjacent layers and is insulated from other layers. Each layer encapsulates data and functionality with clear, well-defined responsibilities.

The User Interface (Ui) layer is responsible for interacting with the user, or the outside world in the case of interacting with third-party software. This layer contains three components:

1. Virtis/Opis GUI module - This is the visible Graphical User Interface with which the user will directly interact for describing bridges for storage in the BridgeWare database and for performing rating, design and analysis.

2. Analytical (Import/Export) modules - These modules are responsible for: (1) importing bridge description data files from other applications into the BridgeWare database, (2) exporting data for analysis programs such as BRASS, (3) controlling analysis programs such as BRASS. This component also includes future analysis, design, and rating modules.

3. Third-Party modules - These modules are software components that are integrated with the Virtis/Opis GUI to perform special analysis (such as to replace or supplement BRASS for rating or design) or may be completely independent of the Virtis/Opis GUI module but use the Virtis/Opis Domain for accessing the BridgeWare database.

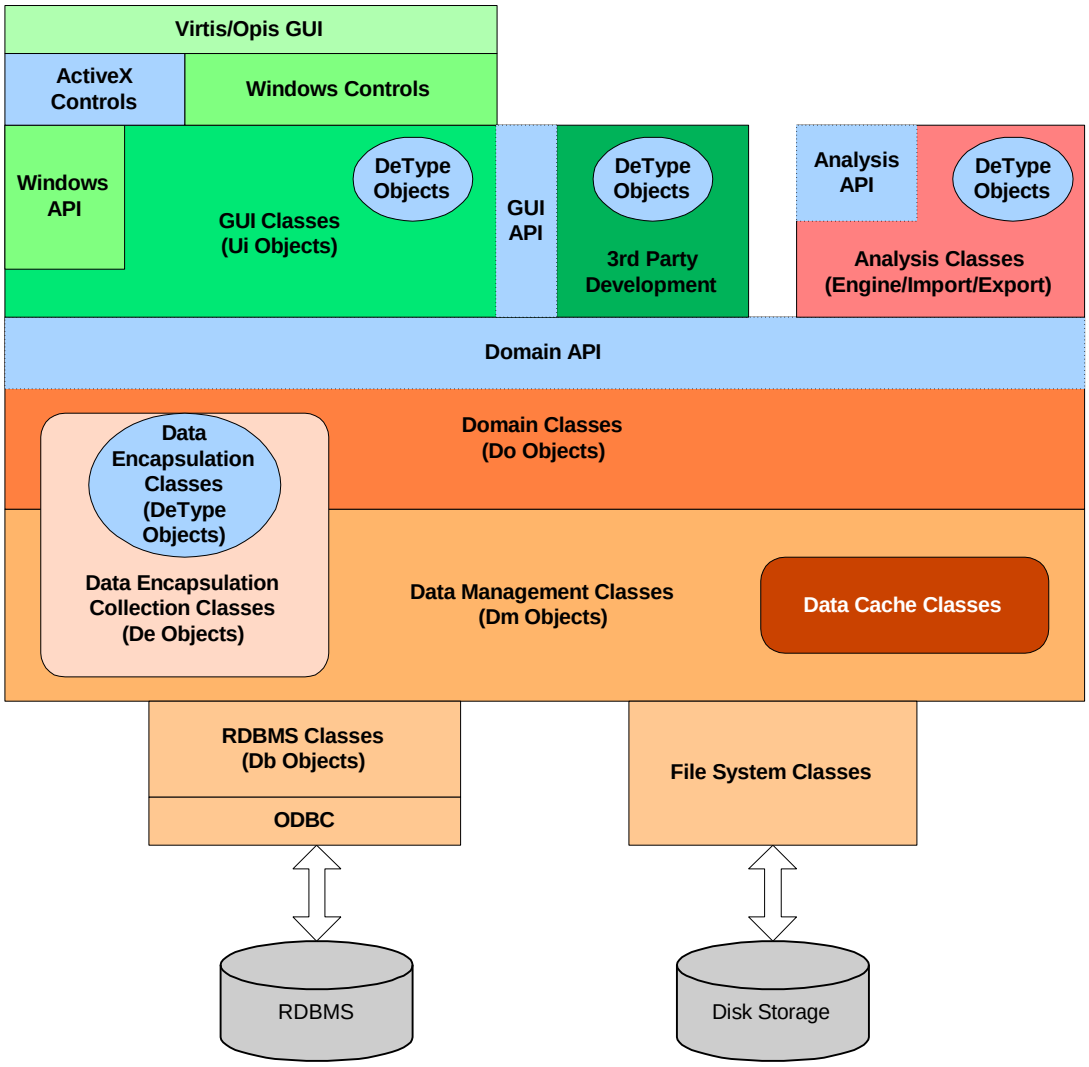


Fig. 1. System Overview

Certain parts of the GUI module, indicated in the diagram as "GUI API", may expose functionality to the outside world for use by third-party developers. This API enables software communication between Virtis/Opis and other applications such as Routing programs.

The Domain (Do) layer is responsible for organizing the bridge data and presenting it to the User Interface layer as a bridge with all the correct relationships among the bridge's components. The domain enforces consistency and "business rules" for all bridge data. Since it is designed to provide convenient navigation according to the semantic structure of the data, the domain is the primary source of data for all GUI and analytical objects. The next section of this paper describes the domain model in more detail.

The Data Management (Dm) layer is responsible for controlling data source access and management of Data Encapsulation (De) objects, which provide attribute values and associated data dictionary information for use by any layer of the system.

The system as a whole is comprised of several thousand C++ classes. Part of the GUI and all of the Domain implement a programming interface compliant with Microsoft's Component Object Model (COM). This interface can be used within any programming environment that supports COM and Automation. This includes Visual Basic and any product that supports Visual Basic for Applications such as Microsoft Excel and Word.

4. Integrated domain model

Both the database and a part of the graphic user interface are organized according to a logical model describing the aspects of a structure which are important for rating and design. Using the methods of object-oriented analysis (Booch, 1994), this logical model takes the form of a domain model. The domain model is not only a software design tool, but it also has evolved into an important part of the system itself, as the collection of objects responsible for enforcing the functional, geometric, and analytical integrity of the data describing each bridge.

Fig. 2 shows a small portion of the domain model, focusing on the static description of steel bridges and the load rating event. The full domain model also includes decks, culverts, prestressed and reinforced concrete beams, substructures, appurtenances, geographic location, security, bridge management, analytical results, and libraries.

As is evident from the diagram, the domain model results from a thought process of disassembling a bridge into the smallest relevant components. Care is taken to organize the components in a way that reflects their roles and structural behavior. AASHTO bridge design specifications were especially useful in classifying components and in the choice of standard terminology. Other existing systems were also consulted to ensure that the new system would be logically compatible with them as much as possible. The objects of each class in the domain model have their own specific role to play in the calculations for structural analysis and specification checking. Most of them also have a visual representation in the schematic graphic diagrams presented in the user interface, and most have their own data attributes to be stored in the system's relational database.

As the diagram indicates, each bridge is a collection of structures, and each structure is a collection of members. There are many different types of members, and each has its own role to play in the structural system's effort to resist the many kinds of applied loads. There are also many forms of physical implementation of members, some of which have a further breakdown of member components, such as the plates and stiffeners on a built-up steel beam.

An important feature of the domain model is a location-definition pattern. The functional requirements and location of an object are stored and managed separately from the physical definition of the object. For example, the location of each girder is represented in the domain model in the girder class, which inherits from superstructure member, a part of the superstructure definition. The physical characteristics of the girder are provided by steel beam definition, which inherits from spanning member definition. This separation of function and implementation allows the definition of a physical girder to be re-used in many places within the same structure definition. For bridge design, this same feature also allows the storage and comparison of multiple alternative definitions for the same member.

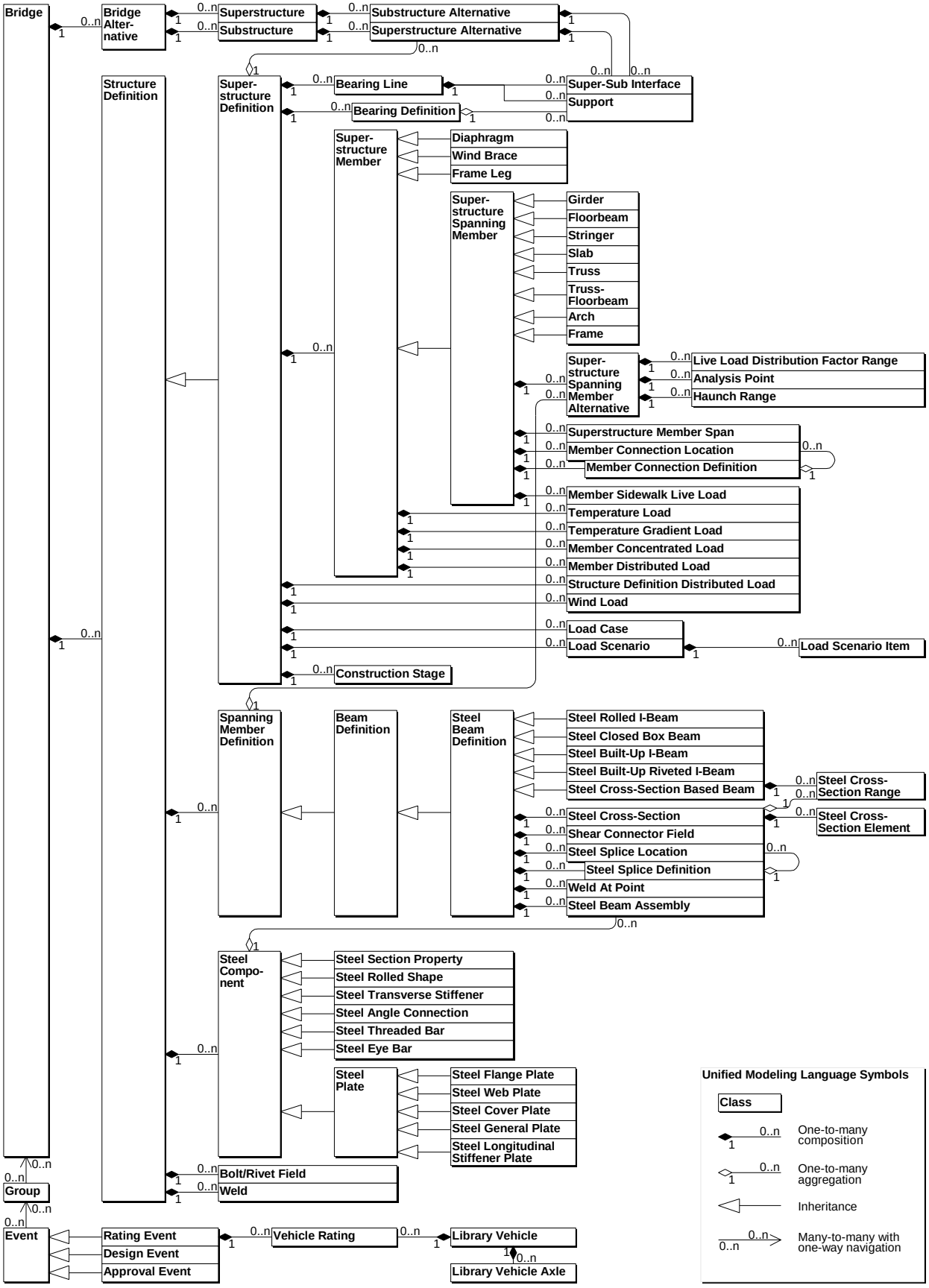


Fig. 2. Domain Model

Classes representing function and implementation are related to each other by means of associative classes, which normally have "alternative" or "assembly" in their names. Often these classes have relatively few input data items, but relatively many output items. For example, the superstructure spanning member alternative class makes it quick and easy for a user to apply a spanning member definition to a given spanning member with very little work. However, features are provided for the definition of distribution factors[2] and points of interest[3], both allowing the user to analyze in as much detail as desired, how the member definition can be expected to perform when used in the indicated position.

Another important feature of the domain model is its ability to represent either of the two most common ways of describing a bridge superstructure. Most existing load rating packages support *cross-section-based input*, the ability to describe the cross-section of a beam at each of a finite number of points. This type of input requires the engineer to abstract the beam — either as built (rating) or as conceived (design) — into these cross sections as a part of data definition. This requires professional judgement and is error prone. When used conservatively, it has a broad range of application. The alternative form is *schedule-based input*, where all the components of the beam are defined individually. Here the information entered into the system is expressed in much the same way as it is represented on the plans. The data requirements and associated graphical interfaces parallel the physical description of the bridge. The importance of this mapping is that the abstraction of the bridge into discrete cross sections is performed within the software; this requires less judgement and is more reliable. Moreover, data may be entered by technical personnel familiar with bridges, but with less training required to provide the abstractions required by the cross-section-based method.

Usually, schedule-based input provides a more precise description of the beam, but requires more data-entry. However, it achieves broad applicability without the need for as many conservative simplifying assumptions. Also, because it is more generic, the schedule-based process is less closely tied to the underlying analytical engine which performs the load rating or design calculations. Both approaches can yield the same results under the same assumptions, but they diverge when assumptions and analytical requirements become more complex, or different analytical processes are employed.

The domain model in Fig. 2 also shows a part of the more dynamic relationship that is used when a bridge is rated. In the Virtis/Opis user interface, users are allowed to combine bridges into groups for any purpose and study the group with respect to one or many vehicles. This combination offers some powerful capabilities, such as:

1. Design or rate a bridge for several vehicles.

2. Study several design alternatives for several vehicles.

3. Rate a set of bridges (grouped along a route) for an overload permit vehicle.

---

[2] Distribution factors abstract a 3-D bridge system into a 1-D mathematical model that represents the behavior of a member in the system. These factors are important to the entire analysis.

[3] Location on a bridge where the effects of load are expected to be critical or of special interest due to construction considerations, specific as-built details, etc.

4. Perform strength evaluation for reporting standardized values to bridge management agencies/entities at the local and national level.

5. Study groups of bridges with particular design details that could be of concern, e.g., weld details.

6. Study bridge management decision-making affecting the performance and life cycles of groups of bridges, such as the effect of a design/rating specification modification, or the effect of policy alternatives on the health of a bridge inventory.

Within the graphic user interface, this feature is supported by allowing users to drag bridges into folders, which are then analyzed as a unit.

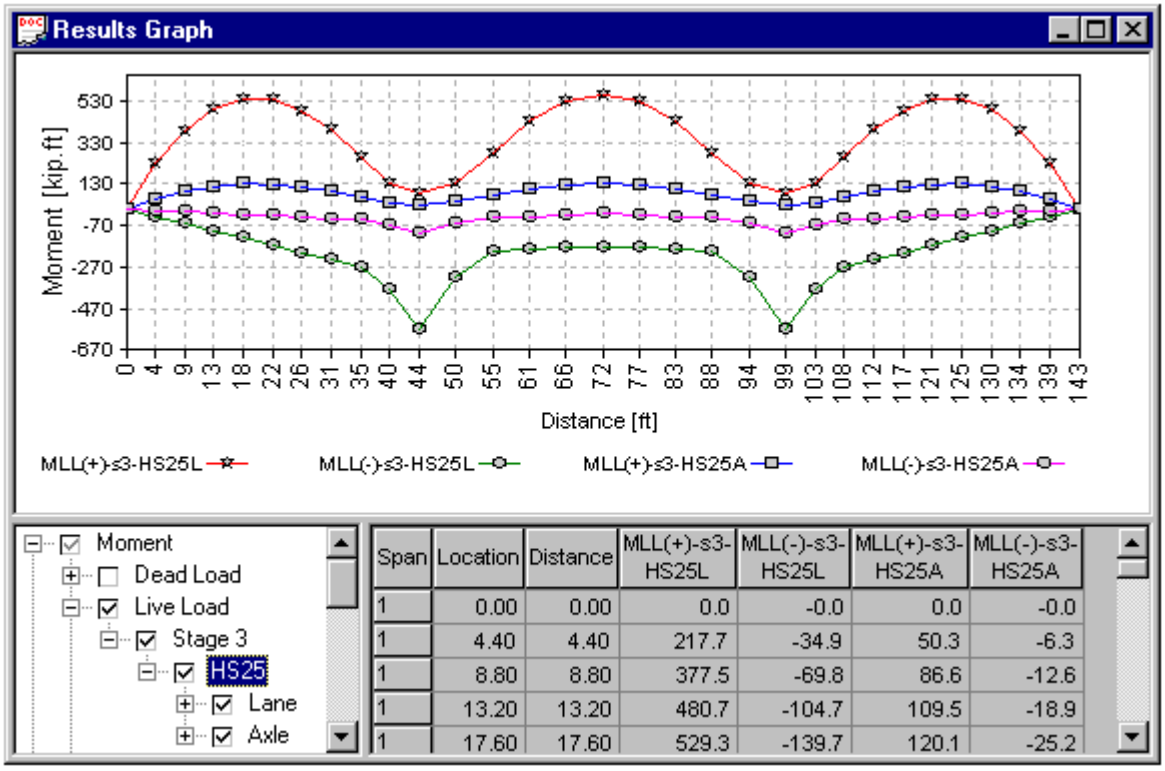
5. Graphic user interface

Load rating can use a visual representation of a bridge as a way of detecting data entry errors or modeling deficiencies, while bridge design can use the same visual images to help the designer to understand the full implications of his creations. Both activities can also use visualizations of structural behavior to understand how an existing structure or a new design can be improved. Finally, both applications are a part of the same life cycle: once a new bridge is properly modeled in the design process, the same data remain available for subsequent load rating after the bridge is built.

An important Virtis project objective is to make load rating as easy as possible, especially the time-consuming initial creation of the structural model. Libraries of standard vehicles, loads, steel and prestressed shapes, load and resistance factors, materials, parapets, and other bridge components allow bridge models to be built quickly in a drag-and-drop manner. As a bridge model is constructed, a graphical schematic framing plan, elevation view, cross-section view, and other schematics provide feedback and make common types of errors more apparent. All or part of a bridge can be copied to another bridge. The ability to copy and paste bridges, members, and other components at any level is especially powerful in reducing data entry time.

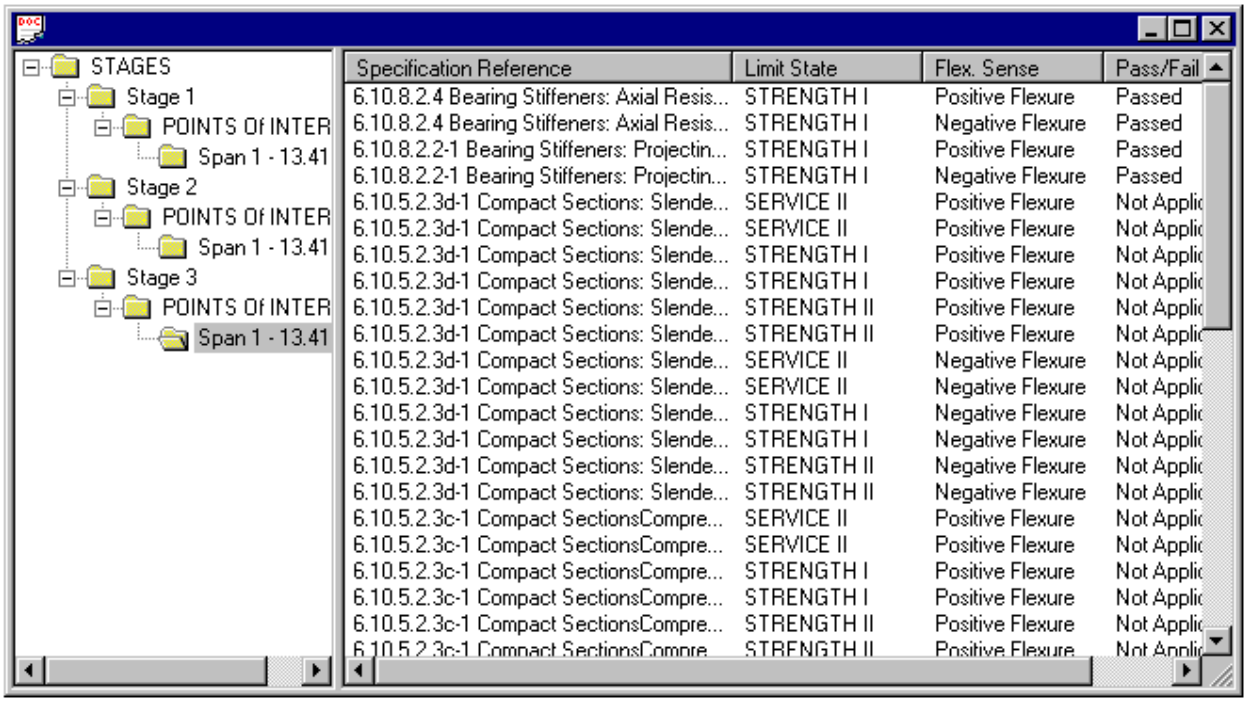
In addition to the features described above for Virtis, Opis will provide a set of output reports to help the designer understand the performance of a new bridge. A tree-structured graphical representation of the AASHTO Load and Resistance Factor Design specification indicates whether each article is passed or violated, and provides access to the detailed calculations for the bridge as well as the specification text. A suite of X-Y plots shows moments, shears, deflections, actual vs. capacity envelopes, influence lines, and other valuable information.

Fig. 3 is a typical X-Y plot that illustrates the results from the structural analysis. Fig. 4 illustrates the results from detailed structural specification checking. The structural specifications that guide bridge design in the US are approximately 1100 pages. These specifications involve a detailed prescription for structural analysis and performance requirements. Virtis and Opis provide the computations that are performed at every level and permit the user to drill down or query conveniently for all relevant details, in a format similar to what could be performed with hand computations.



Span	Location	Distance	MLL(+)-s3-HS25L	MLL(-)-s3-HS25L	MLL(+)-s3-HS25A	MLL(-)-s3-HS25A
1	0.00	0.00	0.0	-0.0	0.0	-0.0
1	4.40	4.40	217.7	-34.9	50.3	-6.3
1	8.80	8.80	377.5	-69.8	86.6	-12.6
1	13.20	13.20	480.7	-104.7	109.5	-18.9
1	17.60	17.60	529.3	-139.7	120.1	-25.2

Fig. 3. A typical Opis plot of structure performance



Specification Reference	Limit State	Flex. Sense	Pass/Fail
6.10.8.2.4 Bearing Stiffeners: Axial Resis...	STRENGTH I	Positive Flexure	Passed
6.10.8.2.4 Bearing Stiffeners: Axial Resis...	STRENGTH I	Negative Flexure	Passed
6.10.8.2.2-1 Bearing Stiffeners: Projectin...	STRENGTH I	Positive Flexure	Passed
6.10.8.2.2-1 Bearing Stiffeners: Projectin...	STRENGTH I	Negative Flexure	Passed
6.10.5.2.3d-1 Compact Sections: Slende...	SERVICE II	Positive Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	SERVICE II	Positive Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH I	Positive Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH I	Positive Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH II	Positive Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH II	Positive Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	SERVICE II	Negative Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	SERVICE II	Negative Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH I	Negative Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH I	Negative Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH II	Negative Flexure	Not Applic
6.10.5.2.3d-1 Compact Sections: Slende...	STRENGTH II	Negative Flexure	Not Applic
6.10.5.2.3c-1 Compact SectionsCompre...	SERVICE II	Positive Flexure	Not Applic
6.10.5.2.3c-1 Compact SectionsCompre...	SERVICE II	Positive Flexure	Not Applic
6.10.5.2.3c-1 Compact SectionsCompre...	STRENGTH I	Positive Flexure	Not Applic
6.10.5.2.3c-1 Compact SectionsCompre...	STRENGTH I	Positive Flexure	Not Applic
6.10.5.2.3c-1 Compact SectionsCompre...	STRENGTH II	Positive Flexure	Not Applic
6.10.5.2.3c-1 Compact SectionsCompre...	STRENGTH II	Positive Flexure	Not Applic

Fig. 4. Navigating to the details of specification checking

6. Interface with BRASS

Another objective of both Virtis and Opis is to be able to check a load rating calculation by applying more than one software package to perform the calculations independently. An obvious way to accomplish this is to define a standardized interface through which third-party load rating calculation packages can withdraw bridge data and deposit the results. BRASS, the analysis application that is being delivered with Virtis and Opis, is the first such third-party package and provides a working example of how the interface is intended to be used.

The interfaces to BRASS (and other applications) exploit a common object-oriented development technique: the use of a "wrapper" or "adapter" (Gamma et al, 1995) class which marries the object-oriented Virtis/Opis software with the decidedly non-object-oriented existing base of structural software. This application-programming interface (API) will be an open and published portion of the AASHTO system.

The interface to BRASS is bi directional. First, the input data are exported to BRASS using an ASCII metafile format. This file structure uses the same BRASS commands that would be used if BRASS were executed outside of the Virtis/Opis environment. The commands are well documented and this intermediate step provides a checkpoint for data definition between the two systems. The input data requirements are very small, usually less than one Kbyte. The typical user is not concerned with this intermediate step.

To accomplish this export step, the application developer creates objects that read the Virtis/Opis domain model and output the data in a form which the third-party package can read. The BRASS example included with the system provides a template for this step. For external systems that are already object-oriented and can access COM interfaces, this intermediate step can be skipped by accessing the domain directly. When necessary, the third-party developer can also provide data entry screens for information specific to the program, which are then available to the user in Virtis/Opis at appropriate places.

When the external analysis is complete, output data from the third-party package are passed to Virtis/Opis, using classes supplied for this purpose. The output is voluminous, typically several tens of megabytes. This includes many X-Y plots, and detailed capacity and specification checking information. Virtis/Opis provides the User Interface features necessary to view the output data. Many of the data may not be usefully stored in the long term and are classified as semi persistent data. Such data are very useful for on-line studies but can be easily regenerated based on the definition of the bridge and vehicle characteristics, and therefore need not be permanently stored. Storage options for these data will be available.

7. International Considerations

The architecture of Virtis/Opis is open and general. The ability to interface with alternative analytical engines extends to the use of design codes from other countries, when needed. The system complies with Microsoft guidelines for internationalization, which facilitate the translation to other languages and the use of local conventions for data representation. All measurements are stored in SI units, with several alternatives for metric and English display conventions.

8. Conclusions and future directions

Virtis and Opis are designed to be powerful tools to serve an existing critical need, exploiting object-oriented technology, client-server databases, and modern graphic user interfaces to significantly advance the state-of-the-practice of design and load rating in the United States. With the generality of the object-oriented design of the systems, they have the potential to become a core infrastructure, a nucleus around which tools that are even more powerful can be built.

For example, having all of its data in a standardized accessible form makes it economical within Virtis to build powerful new features for management of the load rating process and for support of routine business activities such as policy development and overload permit application review. In order to function as a decision support tool for the diverse overload permit management processes that exist in every state, Virtis once again will use object-oriented techniques to build a standardized interface to communicate with outside systems. This interface will support the construction of a model of the candidate truck (from the permit application) and a model of the route the truck will take (from the state's existing geographic information system). It then prepares the load rating information and results in Virtis for review by a load rating engineer, who can then efficiently decide whether to approve the application.

The same tools, which the rating engineer uses to describe an existing bridge, will be used by the design engineer to describe his concept of a new bridge, which can then be evaluated. Eventually, AASHTO hopes to evolve Opis into a true engineer-in-the loop design package, including automated capabilities to search and select design variables which optimize performance requirements. This long-range vision includes a new object-oriented engine for analysis and specification checking, for both superstructures and substructures.

9. References

Booch, G., (1994), Object-Oriented Analysis and Design with Applications, Addison-Wesley, New York, NY, USA, 1994.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, New York, NY, USA, 1995.

10. Authors

Paul D. Thompson, 2425 Hawken Drive, Castle Rock, CO, 80104, USA.

James A. Duray, Michael Baker Jr., Inc., Airport Office Park, Building 3, 420 Rouser Road, Coraopolis, PA, 15108, USA.

Jay A. Puckett, BridgeTech, Inc., 302 South Second, Suite 201, Laramie, WY, 82070, USA.

Jeffrey J. Campbell, Michael Baker Jr., Inc., Airport Office Park, Building 3, 420 Rouser Road, Coraopolis, PA, 15108, USA.

The authors wish to acknowledge the contributions of the Virtis Task Force, chaired by Don Flemming of Minnesota DOT; the Opis Task Force, chaired by Jim Roberts of Caltrans; Dave Pope of Wyoming DOT; Kurt Johnson of AASHTO; and the many engineers who have contributed to the project through their work on the various Technical Advisory Groups.