Computers and Concrete, Vol. 1, No. 3 (2004) 303-324
DOI: http://dx.doi.org/10.12989/cac.2004.1.3.303

Framework for a general section designer software component

Naveed Anwar[†] and Worsak Kanok-Nukulchai[‡]

Asian Institute of Technology (AIT), Bangkok, Thailand

(Received February 9, 2004, Accepted July 12, 2004)

Abstract. The Component-Based Software Development (CBSD) has established itself as a sound paradigm in the software engineering discipline and has gained wide spread acceptance in the industry. The CBSD relies on the availability of standard software components for encapsulation of specific functionality. This paper presents the framework for the development of a software component for the design of general member cross-sections. The proposed component can be used in component-based structural engineering software or as a stand-alone program developed around the component. This paper describes the use-case scenarios for the component, its design patterns, object models, class hierarchy, the integrated and unified handling of cross-section behavior and implementation issue. It is expected that a component developed using the proposed patterns and model can be used in analysis, design and detailing packages to handle reinforced concrete, partially prestressed concrete, steel-concrete composite and steel sections. The component can provide the entire response parameters of the cross section including determination of geometric properties, elastic stresses, flexural capacity, moment-curvature, and ductility ratios. The component can also be used as the main computational engine for stand-alone section design software. The component can be further extended to handle the retrofitting and strengthening of cross-sections, shear and torsional response, determination of fire-damage parameters, etc.

Keywords: data communication; structural engineering; structural models; information management; computer application; objects; components; patterns.

1. Component Based Software Development (CBSD)

The mechanical and electronic industries have long embraced the idea of the component-based manufacturing of their main products. For example, several components used in cars are identical across different models or even different brands. A similar concept is used in the manufacturing of computers, where the standard components are assembled and integrated into different computer models, brand names, and configurations.

The component-based manufacturing concept has been extended to the development of software applications in recent years. Component Based Software Development (CBSD) is a logical extension of object-oriented concepts and the corresponding object technologies, which have pervaded in the software industry since the early 1990s. With the almost universal availability of the Internet and other network facilities, it has become viable for application software not only to be

† Associate Director, Asian Center for Engineering Computations and Software (ACECOMS)
‡ Professor, School of Civil Engineering

deployed at locations different from their users' computers, but also to break up and deploy large systems from several locations. The CBSD has also made it possible for parts of the same software to be developed by different organizations, and to assemble new applications from existing specialized or generalized components. In such a scenario, the component-based architecture can significantly formalize, streamline, expedite, and enhance the software development process for structural design applications as well.

Since the introduction of object-oriented concepts, a whole new breed of object technologies and paradigms has evolved for the development of software systems. One of these is the concept of components. Component Based Software Development is supported by, and has given rise to several new paradigms, programming environments, languages, tools and development architectures. These include Object Linking and Embedding (OLE), ActiveX, Common Object Model (COM), Distributed Common Object Model (DCOM), Open DOC, Java Applets and Java Beans, Common Object Request Broker Architecture (CORBA), Distributed Computing, Client Server Applications (CSA) and the new .Net framework, Sun Open Network Environment (Sun One) (http:// www.sun.com 1999) and the Java 2 Enterprise Edition (J2EE). A completely new way of building systems using software components has evolved (Crnkovic 2002). Most programming languages now provide support to the use of conventional function libraries, as well as these new component-based developments. The .Net framework introduced by Microsoft in July 2000, provides a productive multi-language environment for platform-independent development and deployment of applications using Extensible Markup Language (XML) as the standard for communicating and processing information between Web services (or components) for Visual Basic (VB) and C-based programming (Brill 2001). The J2EE is a conceptual definition for enterprise system architecture providing design philosophy for large, scalable (Web-enabled) systems. This definition comprises of several Application Programming Interfaces (API) for email, database, distributed computing and web-based applications. J2EE is a new deployment specification for packaging the Java components into a single enterprise system. It extends the basic Java Archive standard for a single deployable file to the Web Archive (WAR) and Enterprise Archive (EAR) formats, for deploying large component based systems (Brill 2001). A component coordination model for component-based systems has been presented (Mathee and Betanov 2000). It is believed that component-based approaches will be at the forefront of software development in the next few years (Capretz, *et al.* 2001). The challenges and difficulties faced during development, implementation and use of component-based systems have been addressed recently (Crnkovic and Larsson 2002). An application framework for the development of simulation tools for construction projects has been proposed (Hajjar and AbouRizk 2000), that is based on the object oriented library and design patterns. They have demonstrated that use of component based software development significantly reduces the development time for new program as well as adding functionality to existing tools. Krishnamoorty, Venkatesh and Sudashan (2002), have applied the concept of object-oriented framework to the development of program for space truss optimization. This framework, however, is only limited and specific to the truss analysis and optimization problem using genetic algorithm and not intended for overall integrated design of general structural system.

2. CBSD framework for structural design applications

The structural engineering discipline, and hence the scope of computer applications and software development, deals with a very broad range of activities and applications. The applications can be classified by the type of structure it handles and the extents of the overall design process. The applications can be as small and specific as a program for the design of reinforced concrete cross-sections, or as large and general as a fully-integrated analysis, design, detailing, and costing program for structures. Recently, applications for design process management, design documentation and web-based information collaboration have also become popular (Pena-Mora and Choudary 2001). Current structural engineering software not only addresses the computational aspects, but also focuses on interactive and graphical user interfaces, specialized pre-and post-processing, graphic visualization, database management, and integration and interaction with other software systems (Adeli and Yu 1995), (Stamatakis 2000) and (Sotelino, *et al.* 1998).

Formal development of software typically deals with two disciplines or domains: software engineering (along with related knowledge and expertise) and the application domain, such as structural engineering as in this case (Pressman 1992). Significant developments and diversifications have occurred in both the software development industry and the structural engineering discipline. At the same time the structural engineers' expectations of software range, application and level of sophistication are ever increasing. These expectations are driving attempts to develop more complex, comprehensive and integrated software systems. It is becoming increasingly difficult for the researchers and developers of software for structural engineering to keep abreast of all the latest developments, both within their own field and in the general structural engineering industry. It therefore makes sense for individual researchers or team of developers to focus on their own particular area of interest and expertise and develop programs and software within their specialty. These software or programs, when developed using well-defined frameworks and patterns, can become software components that can then be connected in a variety of ways to develop and deploy complete systems or solutions. In a typical scenario, researchers in the finite element technology could focus on developing a dedicated structural analysis component. Experts in computer graphics could develop components for graphic display and manipulation of structural models, as well as visualization of response. Similarly, specific components could be developed for design of concrete, steel or composite structural members. These components could be used time and again in a variety of ways and be physically integrated into applications or used over the Internet as web services or part of application servers. The component frameworks formalize and generalize the basic design issues in the structural design software. Component-Based Software Development ultimately hinges on the object-oriented techniques and concepts. Several successful applications of object-oriented software development in structural engineering have already been demonstrated, (Adeli and Yu 1995), (Anwar and Kanok-Nukulchai 1996), (Aster, Bergmeister, Rio, Schonach, and Sparowitz 1998), and (Tailibayew 1999).

For these components to work together, a framework defining the overall architecture must be developed. Such a framework has two parts: that dealing with the computer science aspects (commonly known as horizontal framework), such as programming environments, operating systems, communication protocols, hardware compatibility etc.; and that dealing with the structural engineering aspects (commonly known as vertical frameworks). Several horizontal frameworks already exist, such as COM/DCOM, CORBA (Pritchard 1999), Net, J2EE. As for the vertical frameworks in A/E/C discipline, an information technology-based framework for large-scale project planning with special emphasis on IT investment, and strategic planning using IT diffusion concepts has been proposed (Pena-Mora, *et al.* 1999). A detailed framework using components library to build software for construction management applications has also been proposed (Pena-Mora, Vadhavekar and Dirisala 2001). Their paper demonstrates the importance and usability of component-based software development in civil engineering applications. It presents not only the basic framework for developing components using patterns, but also incorporates a methodology for searching existing components and patterns for re-usability in new system. With regard to the development of component-based software in structural engineering, however, no well-defined domain-specific, vertical framework could be found in the literature.

3. Information oriented component based framework

By definition, frameworks "... provide a skeletal design that can be built upon to create an organized system where many packages or components work together" (Stevens and Pooley 2000). The basis of a framework in the development of structural design applications is laid out in Fig. 1, where the overall information space of a construction project and the role of structural design are shown. A conceptual information package architecture based on this information model is shown in Fig. 2, with the proposed information flow mechanism.

In general, the top level role of an application is to define the bounds of structural design information space; establish the extent of information processing to be performed by the application; establish ways and means of obtaining the information needed from other information sources, identify and integrate packages to complete the contents of the required information space; and finally, update and communicate with the outgoing information sources. The overall conceptual structural design information package architecture is shown in Fig. 3. This framework is derived from the information content and flow in a typical structural design process. The main packages have been identified along with their linkage to the information space as well as the user interface.

The specific design and functionality of the individual packages identified above vary significantly depending on the type of structure and level of integration of the packages. However, the common framework and pattern for these packages can be defined in terms of the basic role they play in the

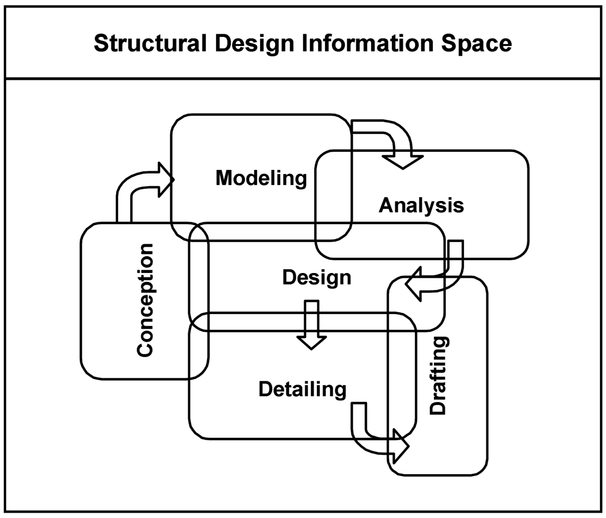


Fig. 1 Structual design information space

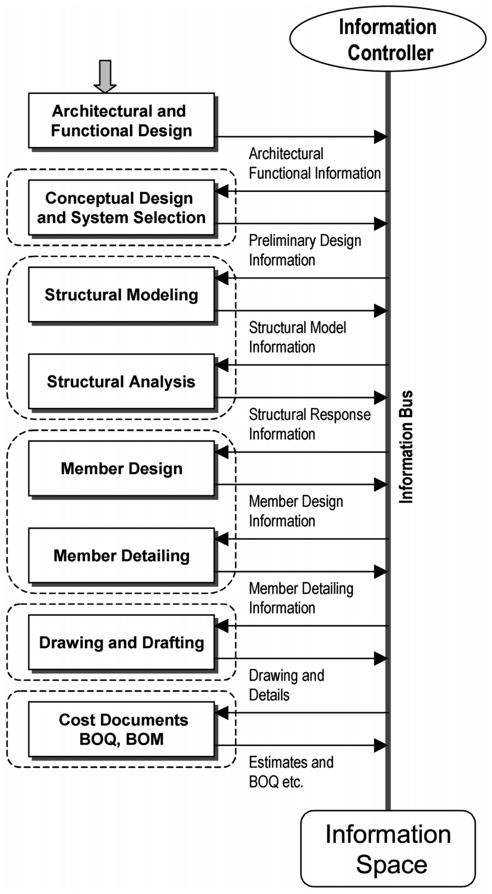

Information Controller
Architectural and Functional Design
Architectural Functional Information
Conceptual Design and System Selection
Preliminary Design Information
Structural Modeling
Structural Model Information
Structural Analysis
Structural Response Information
Member Design
Member Design Information
Member Detailing
Member Detailing Information
Drawing and Drafting
Drawing and Details
Cost Documents BOQ, BOM
Estimates and BOQ etc.
Information Bus
Information Space

Fig. 2 Information flow model for structural design process

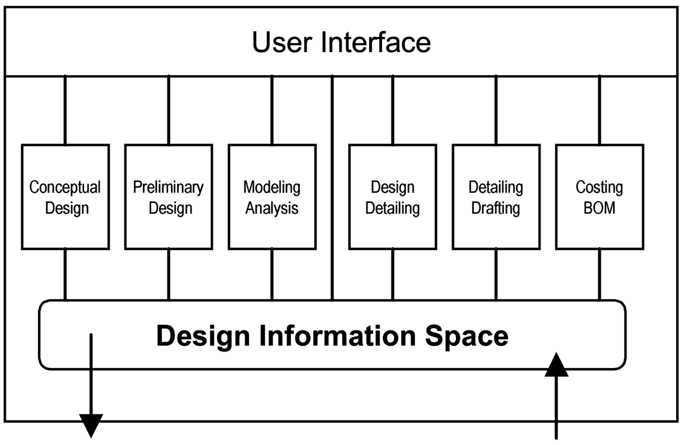


Fig. 3 Conceptual structural design information package architecture

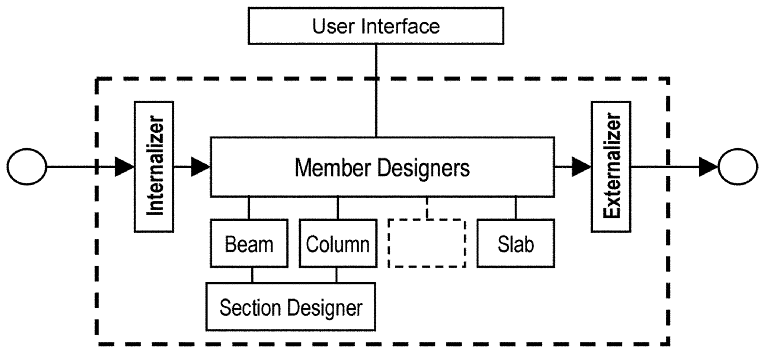


Fig. 4 The designing package

overall application. Each package is basically an information processor. Upon activation, it takes the required information from the information space using the information bus, converts it to appropriate context, processes this information using its components and produces new information or updates the initial information, converts it back to the global context, and returns it to the main information space. A sample framework pattern for the structural design package is shown in Fig. 4. The internal working of the package is not of concern at this stage of framework definition. In some ways, the general package framework is similar to the application framework, except that in this case the packages within the application are replaced by components within the package.

4. The section design (SD) component

4.1. Significance

For the Component Based Software to work, it is essential that ready-to-use plug-able components

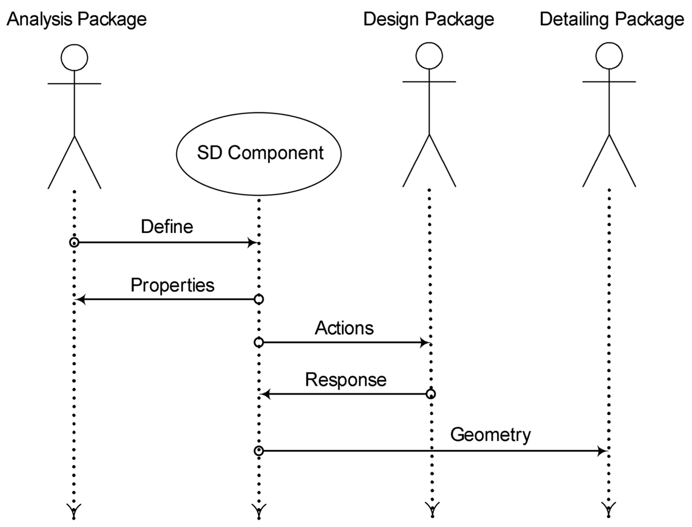


Fig. 5 Use case scenario for the SD component in overall software package framework

are available. Almost every structural analysis and design software handles the design of member cross-sections such as columns, beams, etc. The functionality to design a typical cross-section for frame type members can be encapsulated into a complete component that can be used both with the analysis package and with the design package. When linked to the analysis package it can provide the ability to create or define the cross sections and to compute the properties needed for analysis. When linked with the design package, it can provide the functionality to design the cross-sections, check the stresses, compute the capacity ratios, and determine the performance and ductility parameters. The section design component can also be used in the detailing package to provide the cross-section geometry, graphics and other detailing information obtained or updated from the design process. This use-case scenario of the SD Component is shown in Fig. 5. In addition to being used in main packages, the SD Component can be used as an independent stand-alone program as shown in Fig. 6.

4.2. Functionality

The SD Component, as proposed here, is intended to provide all necessary functionalities to determine the required response of structural member cross sections subjected to general actions and deformations from a typical finite element analysis, as well as to provide the properties needed for the analysis itself. The specific implementation of the component may be limited to the determination of a simple beam section properties, to a full performance-based analysis and design of general beam-column section used in non-linear dynamic or pushover analysis. The component design discussed and presented in this paper can handle all of these cases. The proposed component is able to handle cross-sections of different materials and shapes, including reinforced concrete, pre-stressed concrete hot rolled steel and composite sections and is able to determine response ranging from computing geometric properties, to elastic stresses, section capacity, ductility and performance indicators. The design and implementation of such a comprehensive component requires careful planning and consideration, as presented here.

4.3. Design patterns

The component needs to be developed using standard design patterns so that their re-use is guaranteed. Three design patterns for the SD Component are shown in Fig. 7, with progressive

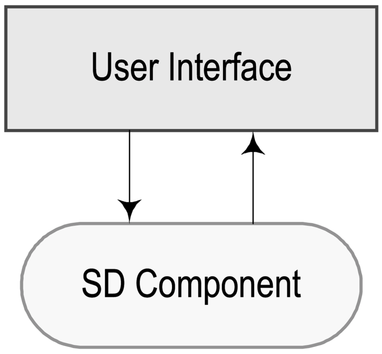
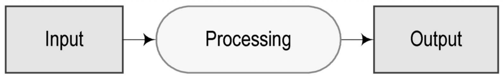
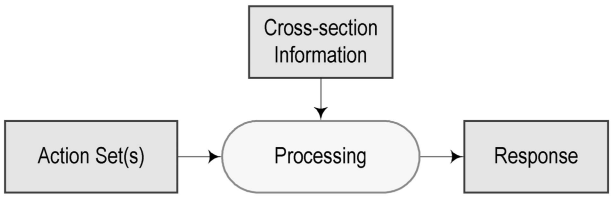


Fig. 6 Stand-alone use of SD component as complete section design program



a) High Level Patterns for SD Component



b) Second Level Pattern for SD Component

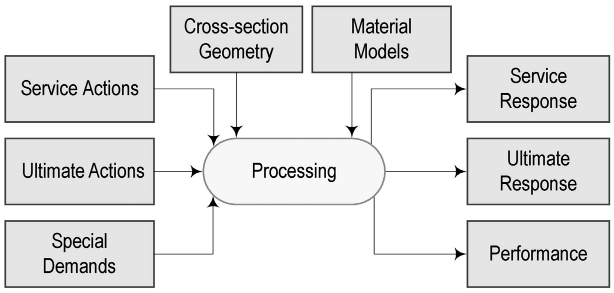


Fig. 7 Patterns for the section designer component

increase in level of details and corresponding reduction in generality. The highest-level pattern in Fig. 7 is in fact applicable to any information-processing component. The second level pattern considers the general demands of the SD Component, whereas, the third level pattern gives the functionality and interface of the SD Component. This pattern is further expanded into a full computational model presented in the next section.

4.4. The computation model for the cross-section behavior

The Section Designer component is designed to provide a significant amount of varied output. Some of the outputs are related to particular input parameters, whereas others are related to the intrinsic property of the cross-section and can be obtained once the geometry and materials are defined. Fig. 8 shows a computation model in which the sequence of computation and the relationship of various response parameters are linked to the input parameters. It can be seen from the figure that the cross-section response is progressively built-up based on the available information either from previous steps or from additional input from the component client. This computation model ensures that the component can be used at various levels and for various purposes. It is important to note that the axial-flexural interaction surface and curves are an intrinsic property of the section and can be generated without the need to define the loading conditions. This feature can be used for writing automated, iterative design procedures.

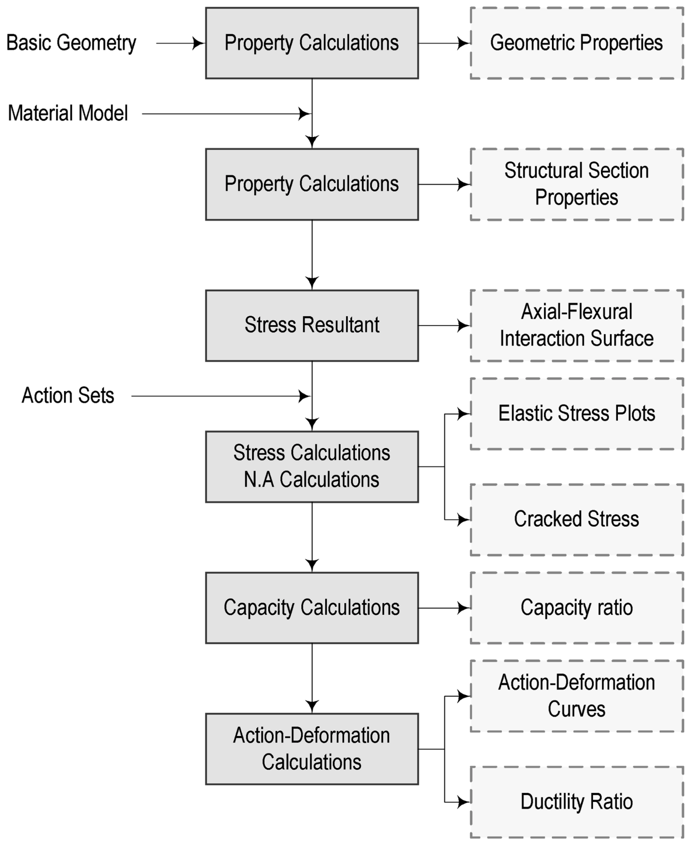


Fig. 8 Overall computation model for the determination of section response

4.5. The public interface

The entire functionality of the component needs to be exposed through a public interface. In other words, any item that is not defined in the public interface will not be available to the users or clients of the component. Therefore, the definition of the interface is critical to the use of the component. The minimum level of access to the component must allow for:

a) Defining section geometry and material models
b) Defining input actions
c) Obtaining required response.

The communication with the component can be done in several ways. The simplest interface would be to simply write three public functions:

1- ReadData (fileName)
2- Execute
3- WriteResults (fileName)

In this model, all data needed by the component will be written in a file and read through a call to

ReadData function. The Execute call will do all computation and WriteResults call will produce the output. However, this interface provides no flexibility in the use of component, and the file data format becomes a rather rigid link between the client and the component.

On the other extreme level of use, an object-oriented interface would expose the object model to the user, with the public interface of each object available for individual access in any order or for any purpose. This would provide complete freedom to the client of the component, but would require several calls and significant amount of programming.

A third alternative is to provide a logical set of higher level access functions in the component that will help in performing bigger and more abstract tasks. This third option is used in the present paper to define the public interface of the component. Table 1 lists the functions that are exposed for use by the component client. This is only a partial list and other functions for more refined access to the component functionality may be added in specific implementations.

4.6. Object model

The SD component does not have a very deep object model. The component itself is fairly specialized and has a lower level of functionality. It basically consists of a collection of shape objects and a collection of material objects. The cross-section geometry is handled by the shape objects that represent both polygon shapes as well as the rebars or prestressing strands. Each shape object is associated with one material object from a collection of various material objects encapsulating material properties and material behavior. The overall object model is depicted in Fig. 9. It can be seen from the figure that the shape objects have a one-to-many relationship with the section object whereas the material objects have one-to-one relationship with the shape objects. Therefore, the section is, in fact, composed of several shapes, each of a specific material.

Table 1 The public interface functions for the section designer component

S No	Public Interface Function
1	New Section() Section
2	Add Polygon Shape() Shape ID
3	Add Point Shape() Shape ID
4	Add Material Type() Material ID
5	Assign Material to Shape
6	Compute Geom Properties() Geom Prop
7	Generate Interaction Surface() IAS
8	Generate Elastic Stresses () Stresses
9	Generate Moment Curvature () M Fi Curves
10	Compute Capacity Ratios() Cap Ratio
11	Add Action Set () Set ID
12	Remove Shape()
13	Remove Material()
14	Draw Section()
15	Draw Section Stress()
16	Get PM Curve () Curve
17	Get MM Curve () Curve
18	Get Neutral Axis() NA Data

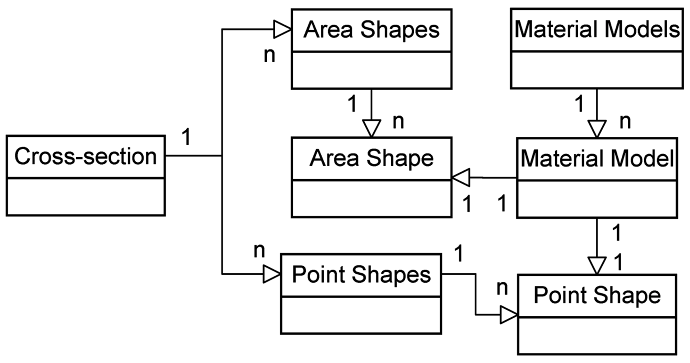


Fig. 9 The overall object model for SD Component

Defining the material model in separate objects rather than as properties of the shape objects allows for much greater flexibility in the handing of complex cross-sections made up of several shapes and materials.

4.7. Class hierarchy

The final coding of the component is done using object-oriented programming and is based on a hierarchy of classes. The SD Component is expected to be used in graphical user interface environment. It is, therefore, imperative that the component contains and exhibits graphics capability as well. The graphics functionality of the component is provided in either of the two ways.

1- The information generated or needed by the component is provided to a separate graphics component that handles the input as well as the graphics display of results.

2- The classes in the component are inherited or derived from the graphics classes so that all graphics interaction is handled by the classes within the component, in collaboration with other graphics related classes.

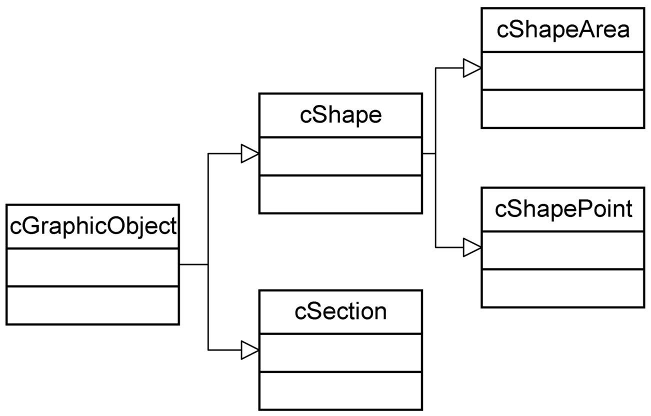


Fig. 10 Hierarchy of shapes and section classes

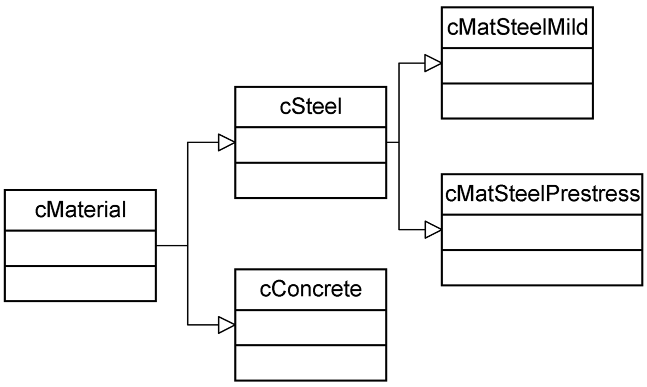


Fig. 11 The Column section classes

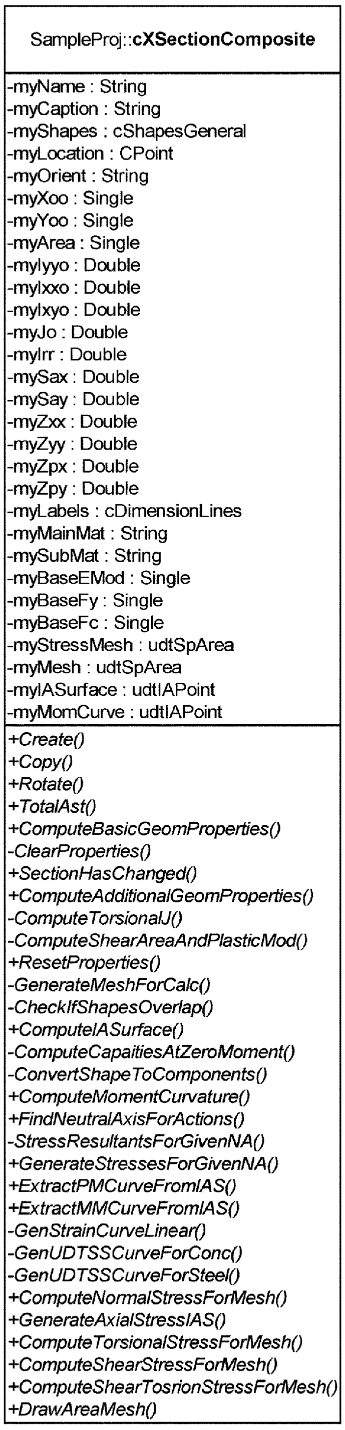
The second option is used in the present research as it has several clear advantages over the first option. The encapsulation of the graphics functionality and behaviors within the component makes the component complete and self reliant. It also provides for easier linkage to main programs as the graphics interaction does not have to be handled by the main program separately. For this reason both the section class itself and the shape classes are inherited or derived from the base graphics class. The shape class is further specialized into polygon shape class and point shape class to handle area and rebars etc. A base material class is used to define the basic material properties and behavior, which is then specialized into concrete material and steel material to handle specific demands of these materials. For example the concrete material class may be needed to handle creep and shrinkage in a specialized manner, not relevant to steel. Similarly the steel material is further specialized into mild steel to handle structural steel, and rebar and into pre-stressing steel to handle pre-stressing strands and wires. The need for this specialization is to tackle the special behavior such as relaxation, pre-stress losses etc., which is not relevant to the reinforcing or structural steel. The partial design of the section class, the shape class and the material class is shown in Fig. 10 and Fig. 11 respectively.

The main cross-section class contains the functionality to combine these shapes, mesh them into appropriate non-overleaping, exclusive and convex polygonal meshes. Each mesh element is derived from the corresponding shape and inherits the material properties. The final mesh elements are then used to compute various cross section responses including the axial-flexural capacity, the moment-curvature curve, the geometric properties, stress distribution and other required responses. Fig. 12 shows the overall design of the cross-section class showing the main properties and services.

5. Unified approach to cross-section response

5.1. The diversity of the problem and unification of the solution

Generally, design of structural members can be divided and categorized in many ways based on type of member, type of material, type of applied action, type of cross-section shape, design code and design method. Often each of these types and their combinations are treated separately and



```
SampleProj::cXSectionComposite
-myName : String
-myCaption : String
-myShapes : cShapesGeneral
-myLocation : CPoint
-myOrient : String
-myXoo : Single
-myYoo : Single
-myArea : Single
-myIyyo : Double
-myIxxo : Double
-myIxyo : Double
-myJo : Double
-myIrr : Double
-mySax : Double
-mySay : Double
-myZxx : Double
-myZyy : Double
-myZpx : Double
-myZpy : Double
-myLabels : cDimensionLines
-myMainMat : String
-mySubMat : String
-myBaseEMod : Single
-myBaseFy : Single
-myBaseFc : Single
-myStressMesh : udtSpArea
-myMesh : udtSpArea
-myIASurface : udtIAPoint
-myMomCurve : udtIAPoint
+Create()
+Copy()
+Rotate()
+TotalAst()
+ComputeBasicGeomProperties()
-ClearProperties()
+SectionHasChanged()
+ComputeAdditionalGeomProperties()
-ComputeTorsionalJ()
-ComputeShearAreaAndPlasticMod()
+ResetProperties()
-GenerateMeshForCalc()
-CheckIfShapesOverlap()
+ComputeIASurface()
-ComputeCapaitiesAtZeroMoment()
-ConvertShapeToComponents()
+ComputeMomentCurvature()
+FindNeutralAxisForActions()
-StressResultantsForGivenNA()
+GenerateStressesForGivenNA()
+ExtractPMCurveFromIAS()
+ExtractMMCurveFromIAS()
-GenStrainCurveLinear()
-GenUDTSSCurveForConc()
-GenUDTSSCurveForSteel()
+ComputeNormalStressForMesh()
+GenerateAxialStressIAS()
+ComputeTorsionalStressForMesh()
+ComputeShearStressForMesh()
+ComputeShearTosrionStressForMesh()
+DrawAreaMesh()
```

Fig. 12 The cross-section class

Table 2 Diversity of the concrete section design problem

Types of Material Combinations	Types of Actions	Location of Reinforcement	Stress-Strain Curve	Cross-section Shape	Design Approach and Method
- Un-reinforced Concrete - Reinforced Concrete - Partially Prestressed Concrete - Fully Prstressed Concrete - Fiber Reinforced Concrete - Steel-Concrete Composite	- Uniaxial Bending, *Mx* or *My* only - Uniaxial Bending and Axial Force, *Mx* or *My* and *P* - Biaxial Bending, *Mx* and *My* - Biaxial Bending and Axial Force, *Mx* and *My* and *P*	- Singly Reinforced - Doubly Reinforced - Arbitrarily Reinforced	- Simplified Rectangular Block - Semi-Parabolic and Full Parabolic Curves - Curves for Confined and Unconfined Concrete - Linear Elastic - Bilinear Elasto-Plastic - Elastic, Post Elastic, Strain Hardening	- Rectangular - Circular - Flanged - General Polygonal	- Allowable Stress Design - Ultimate Strength Design - Load Factor Design

differently for the purpose of determining the design strength. The design strength of a member is largely derived from the design strength or capacity of the cross-section. The diversity of the concrete cross-section design problem is summarized in Table 2. The proposed component is designed to handle most if not all of these specializations. The proposed component is designed to handle several types of cross-section shapes and materials. This requires the development of unified approach for representing cross-section shapes and materials and for determining the cross-section response. Based on unification and integration of the following specializations:

- Beams and Columns
- Reinforced Concrete and Pre-Stressed concrete
- Reinforced Concrete and Composite concrete
- Design Methods, Design Approach and Design Codes
- Cross-Section Shapes and Reinforcement
- Cross-Section Materials

All of these unifications are based on the development of a generalized cross-section representation and determination of its response.

5.2. Generalized cross-section representation and response

A general cross-section consisting of a combination of shapes of different materials is considered including concrete and steel. The cross-section may also contain holes, and reinforcements at any arbitrary locations. These material shapes may lie partially or completely within another shape. It is also considered that all parts of the section have not been built at the same time and that the cross-section has been loaded incrementally. This means that at any given time and loading state the strains in adjacent materials at the same location may not be the same. A generalized representation of such a cross-section is shown in Fig. 13.

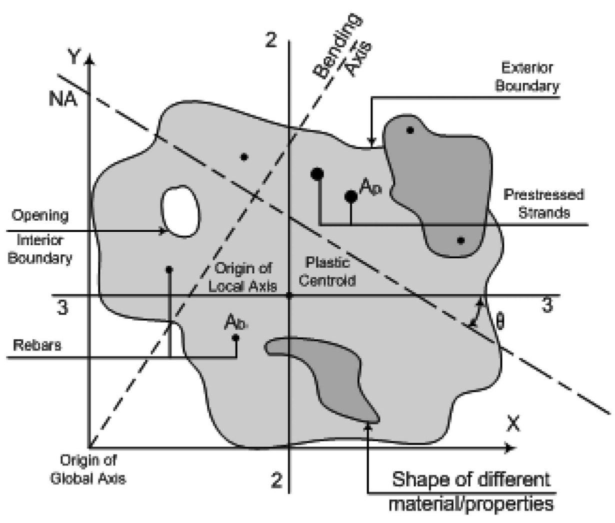


Fig. 13 General cross-section with arbitrary arrangements of shapes, holes and reinforcement

In the proposed model, such complex cross-sections are represented by a combination of two basic entities, polygons and points. The polygons may be used to represent solid shapes and holes, and the points may represent the conventional reinforcing bars and prestressed strands. Curved shapes and boundaries are modeled by several straight edges. Each different material or same material with different properties or loading history is represented by a separate entity. For the convenience of computations, the global X-Y coordinate system is defined such that the entire section (all shapes, holes, reinforcement and prestressed strands) lies in the first quadrant. The plastic centroid of the section is chosen as the origin of the local *xy* axes, and located at X_p and Y_p.

Each material used in a cross-section is defined as a separate entity. For each material the following relationships are defined in the overall material model.

- The basic stress-strain relationship: used to obtain basic stress for a given strain;
- The time dependent stress modification relationship: used to model change in concrete strength and modulus of elasticity with time;
- The basic stress modification relationship: used to model pre-stress conditions, such as residual stresses or other non-strain dependent stresses, and;
- The time dependent strain relationship: used to model creep, shrinkage and relaxation.

Once the materials are defined, they are assigned to corresponding entities on the cross-section. It is important to note that each of these relationships can be assigned to different entities in the section independently. In theory, therefore, any combination of the basic stress-strain relationships, time dependent stress variations, time dependent strain variations and pre-strain conditions can be used for the same entity within the cross-section.

5.3. Determination of section capacity

The capacity of a general cross-section made up of several shapes and materials as described above is determined using the basic concepts of the fiber model. In this approach, the stress at each point (or fiber) in the cross-section is determined and then integrated to find the internal stress-resultants. Although the concept is simple its application requires handling of several aspects that are not as simple. For example, how to discretize the cross-section into appropriate fibers/mesh? How to determine stress at each point? And how to integrate the stresses accurately across the areas? Several approaches and methods have been developed and proposed by researches to handle these issues. Some are applicable to simple or specific sections and materials, whereas, others are quite general in application. The authors have developed a fairly general procedure for automatic discretization of a general cross-section made up of several materials, including the effect of non-linear strain distribution, and non-linear stress-strain relationship.

Discretization of the cross-section converts a planer continuum to fibers or a mesh element of finite size and position. For a general cross-section, represented by a number of complex polygons and points, the points need not be discretized, and each is taken as an individual fiber. Each of the complex polygons is discretized in following levels:

- Primary meshing based on cross-section geometry
- Secondary meshing based on the stress variation along the bending direction
- Tertiary meshing based on the stress variation normal to the direction of bending

After the final mesh has been generated, it is converted into triangles and the stress variation on each triangular element is considered linear in both directions and can be integrated exactly.

Once the cross-section has been discretized, the procedure followed for the determination of capacity interaction surface of any given cross-section is summarized as follows and diagrammatically represented by Fig. 14. Various steps can be omitted from this procedure if certain phenomena are not considered. However, in general:

- Assume a neutral axis depth and orientation for the specified failure criteria. For this assumed depth and orientation of the neutral axis, determine the strain profile.

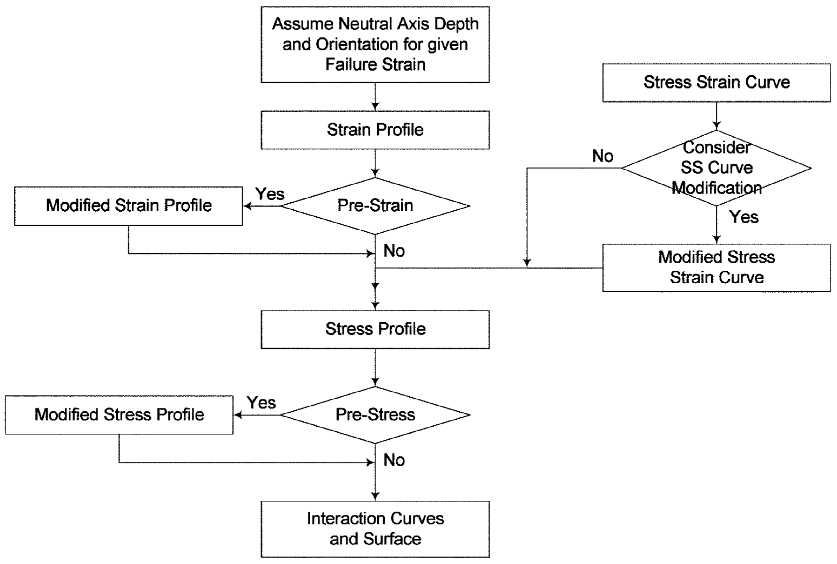

Assume Neutral Axis Depth and Orientation for given Failure Strain
Strain Profile
Modified Strain Profile
Yes
Pre-Strain
No
Stress Strain Curve
No
Consider SS Curve Modification
Yes
Modified Stress Strain Curve
Stress Profile
Modified Stress Profile
Yes
Pre-Stress
No
Interaction Curves and Surface

Fig. 14 Cross-section capacity determination procedure

- Modify this strain profile for strains obtained as an effect of loading history, residual strains or other pre-strains.
- Using the age of concrete determine the modified stress-strain curve for each material in a cross-section.
- Using the modified stress-strain relationships of the various materials over which the strain field is obtained, determine the stress distribution.
- Modify the stress distribution obtained for the cross-section for other time dependent stresses (such as relaxation).
- Calculate the stress resultants using the modified stress distribution.
- Repeat the process for various neutral axis orientations varying the depth throughout the depth of the cross-section.

Plot the stress resultants in 2D and 3D space.

Similar procedures are used to determine other responses such as, elastic and cracked stresses, and moment curvature curves.

5.4. Implementation and deployment issues

The conceptual design of the SD component, as presented here, is fairly general and comprehensive and can be implement in various systems. The implementation, however, requires handling of several specific issues related to three main aspects:

A. The scope and functionality actually implemented in a particular instance of the Section Designer component using this overall framework is presented here.
B. The use-case scenarios for the SD Component and the main analysis and design system in which SD Component will be used.
C. The programming environment and the object technologies used for the development of the SD Component, including the underlying operating system and hardware platforms.

As for the programming environment there are basically two choices:

1- Use the Microsoft-based technologies: In this case, the choice can further be made between the COM/DCOM and the .Net framework. The choice of language can be between VC++ and VB6 for COM/DCOM and between C# and VB .Net for the .Net framework.
2- Use the non-Microsoft technologies: In this case the main object specification is the CORBA, which can be implemented using either C++ or Java.

The above choices are in fact quite general as far as implementation of component-based software systems is concerned, especially on the PC platform. The other systems suitable for Mac/OS, Unix or Linux-based development are not considered here, as their use in the current structural engineering applications is not as widespread as windows-based PC's. Fig. 15 shows the summary of choices for implementation of the SD Component on PC platform.

The deployment, integration and use of standard component is just as important issue as the development and implementation. The SD Component is expected to be used by other software systems on varied environments. Its deployment can be done in several ways:

a) The developers of main analysis program may use the patterns and framework proposed in the present paper and write their own source code and completely integrate it within their software. This is the minimum level of reuse of this component and does not exploit the full benefits offered by the development of this component.

b) The Section Designer component can be implemented as an ActiveX control, or an ActiveX DLL using the COM/DCOM technology. In this case, the interface exposed by the component can be used by the host programs to access its functionality.

c) The component can be implemented as a .Net assembly in which case it can be used in .Net based development environment, again using the public interface of the component.

d) The component can be programmed as a Java Bean and to be used with Java-based technologies.

e) The component is deployed as an ASP (Application Server Provider) or a Web Service implementation that can provide access to the components functionality in Web-enabled system.

The actual implementation and deployment choice will depend to a great extent on the personal and organizational preferences as well as on industry demands. Probably more than one implementation and deployment choice may be considered for wider usage.

In this research, the implementation and deployment is focused on the .Net framework as that is seen by many to be the development choice for a significant amount of PC-based software in the future, replacing the COM/DCOM technologies.

5.5. Sample implementation

A generalized component, within the framework described thus far in the paper, has been developed. The component is designed to be used in component-based structural engineering software as well as a stand-alone application. The component is used in analysis, design and detailing packages, such as BATS, GEAR, CSISectionBuilder and CSICOL to handle plain concrete, reinforced concrete, prestressed concrete, composite section, and hot-rolled sections. The entire response parameters of cross-sections can be determined using this component, including geometric properties, elastic stresses, cracked stresses, moment curvature curves, flexural capacity curves, and ductility ratios. The SD component receives the basic input parameters such as section dimensions, amount of reinforcement, material properties, etc., from the client application. These properties are then used to determine the requested response of the section and the output is

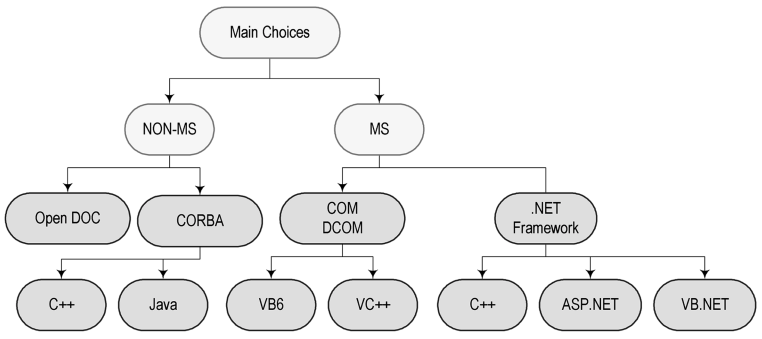


Fig. 15 Some of the choices for the implementation of SD component in PC platform

provided to the client.

As a stand-alone component, the SD interacts with the user through a graphical, highly interactive and user-friendly interface. Material properties for the section may be defined using the various material types available in the material library.

In this implementation the section geometry may also be created easily using various available section shapes in the built-in shape library. The component is also capable to import section coordinates from other file formats such as .txt and .DXF. In addition to this, various tools are provided to facilitate the accurate graphical drawing and editing of section shapes such as rotate, flip, merge and scale. Similarly, tools for addition and distribution of rebars and prestressing strands over the section are provided. The materials defined earlier may be assigned to any part of the section. Thus, the component allows for building of sections having non-uniform material properties for a single section. Once the section and material properties have been defined various outputs may be generated. A few of these are described next in this paper.

5.6. Geometric properties

The geometric properties are computed and reported in terms of the base material property defined for the entire section. If the material specified for the section is concrete of a particular strength, then the properties of all shapes in that section are weighted using the appropriate factors and the concept of modular ratio. The individual properties of the shape are therefore multiplied by this factor and added to the properties of other shapes. This is true even if there is only a single shape in the section.

5.7. Flexural capacity curves

The detailed procedure for the determination of the flexural capacity of cross-sections has been described earlier in this paper. Flexural capacities of sections are reported in various different formats according to user preferences. These include capacity surface generation, P-M curves at various neutral axis angles, M-M curves at various axial load values and capacity ratios for defined loading conditions. These outputs are generated and reported in both graphical and text formats and may be exported to other applications for post-processing. Although the program is able to generate capacity curves for any section and combination of different shapes and materials, it is important that this information is used with proper understanding and checks on the validity and applicability of such calculations. In general, the capacity calculations are intended for reinforced concrete sections, but may also be used for fully composite and plastic sections.

5.8. Moment-curvature curves

The SD component generates the moment curvature curves for a given value of axial load, direction and magnitude of moment, and a specified strain criterion. Various strain criteria such as maximum strain, failure of first rebar, failure of any or all parts of the section may be specified for the computation of the moment curvature curves. The component reports the output in both text and graphical form.

5.9. Stress distribution plots

This component generates the following stress plots on the section:

- The combined normal stress for axial load, moment M_x and moment M_y. This stress calculation is based on elastic properties and the linear strain distribution assuming fully composite and connected behavior of various shapes and components in the section.
- The shear stress distribution along the y and x axes. This shear stress is computed using the general shear stress equation. The same equation is also used to calculate the shear area. All shapes in the section are assumed to be fully connected and fully effective in resisting shear force. The shear stress distribution is computed along two orthogonal axes independently, assuming no interaction.
- Cracked Section Stresses showing stresses variation according to the stress-strain curve assigned to the section are displayed in 2D and 3D space along with the location of the neutral axis.

The section or parts of the section can be rotated at any angle for computation of geometric properties and capacity calculations as well as for determining the stress distributions. This allows capacity calculations to be performed in any direction.

Fig. 16 shows the various outputs that have been generated using CSISectionBuilder®, a software developed on the proposed component based framework.

6. Conclusions

The overall development of a software component for the design of member cross-sections is presented. The component is developed with a view to be used in component-based structural engineering software as a stand-alone program developed around the component. The paper describes the basic use-case scenarios for the component, its basic design patterns, the integrated

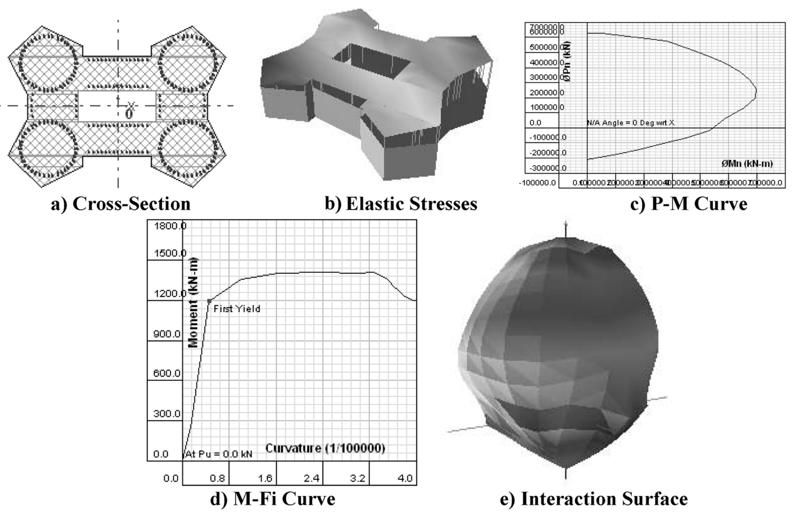


Fig. 16 Various outputs developed using the component-based section designer program

and unified handling of cross-section behavior and implementation issue. It is expected that a component developed using the proposed patterns and model can be used in analysis, design and detailing packages to handle reinforced concrete, partially prestressed concrete, steel-concrete composite and steel sections. The component can provide the entire response parameters of the cross section including determination of geometric properties, elastic stresses, flexural capacity, moment curvature and ductility ratios. The component can also be used as the main computational engine for stand-alone section design software. The component can be further extended to handle the retrofitting and strengthening of cross-sections, determination of fire-damage parameters, etc. Other extensions, specialization and implementations of the proposed component are possible within the overall component-based framework.

References

Adeli, H. and Yu, G. (1995), "An integrated computing environment for solution of complex engineering problems using the object-oriented programming paradigm and a blackboard architecture", *Comput. Struct.*, **54**(2) 255-265.

Anwar, N. and Kanok-Nukulchai, W. (1996), "Application of object oriented techniques in structural engineering software", *International Conference on Urban Engineering in Asian Cities in 21st Century, Proceedings,* Volume 1, School of Civil Engineering, Asian Institute of Technology, Bangkok, Thailand.

Aster, M., Bergmeister, K., Rio, O., Schonach, A. and Sparowitz, L. (1998), "Iterative object-oriented modeling for structural engineering", *Comput. Mech.* New Trends and Applications CIMNE, Barcelona, Spain.

BATS, is a product and trademark of Asian Center for Engineering Computation and Software (ACECOMS), Asian Institute of Technology, (AIT) Bangkok, Thailand.

Brill, G. (2001), "CodeNotes for VB.NET", Random House Inc., New York, USA.

Brill, G. (2001), "CodeNotes for J2EE", Random House Inc., New York, USA.

Capretz, *et al.* (2001), "Component-based software development", *27th Annual Conference of the IEEE Industrial Electronics Society*, IECON2001, Denver, United States.

Crnkovic, I. and Larsson, M. (2002), "Challenges of component based development", *J. Systems and Software* **16**(3) 2002.

Crnkovic, I. (2002), "Component-based software engineering: building systems for software components", *Proceedings-IEEE Computer Societys International Computer Software and Applications Conference, 2002.*

CSICOL, is a product and trademark of Computers and Structures, Inc., Berkeley, CA, USA.

CSISectionBuilder, is a product and trademark of Computers and Structures, Inc., Berkeley, CA, USA.

GEAR, is a product and trademark of Asian Center for Engineering Computation and Software (ACECOMS), Asian Institute of Technology, (AIT) Bangkok, Thailand.

Hajjar, D. and AbouRizk, S. M. (2000), "Application framework for development of simulation tools", *J. Comput. Civ. Eng., ASCE*, **14**(3), July 2000.

Krisnamoorty, C. S., Venkatesh, P. P. and Sudarshan, R. (2002), "Object-oriented framework for genetic algorithms with application to space truss optimization", *J. Comput. Civ. Eng., ASCE*, **16**(1), January 2002.

Mathee, O. and Betanov, D. N. (2000), "A component coordination model for customization and composition of component-based system design", *7th IEEE International Conference and Workshop on Engineering of Component Based Systems*. 3-7 April 2000, Edinburgh, Scotland.

Pena-Mora, F. and Choudary, K. K. (2001), "Web-centric framework for secure and legally binding electronic transactions in large-scale", *A/E/C Projects*, *J. Comput. Civ. Eng., ASCE*, October 2001, **15**(4) 248-258.

Pena-Mora, F., Vadhavakar, S. and Dirisala, S. K. (2001), "Component-based software development for integrated construction management software applications", *AIEDAM*, **15**(2) April 2001.

Pena-Mora, F., *et al.* (1999), "Information technology planning framework for large scale projects", *J. Comput. Civ. Eng., ASCE*, **13**(4), December, 1999.

Pressman, R. S. (1992), (1987), (1982), *Software Engineering A Practitioners Approach Third Edition*, Mcgraw-

Hill International Inc.

Pritchard, J. (1999), *COM and CORBA Side by Side Architectures, Strategies, and Implementations*, Addison Wesley Longman, Inc. USA.

SAP2000, is a product and trademark of Computers and Structures, Inc., Berkeley, CA, USA.

Sotelino, E. D., Chen, W. F. and White, D. W. (1998), "Future challenges for simulation in structural engineering", *Computational Mechanics New Trends and Applications CIMNE*, Barcelona, Spain.

Stevens, P. and Pooley, R. (2000), *Using UML Software Engineering with Objects and Components*, Pearson Education Ltd.

Stamatakis, W. (2000), *Microsoft Visual Basic Design Patterns*, Microsoft Press Washington, USA.

Tailibayew, S. I. (1999), "Development strategies leading to a general purpose finite element analysis program using object-oriented paradigm", M. Eng. Thesis, No. ST-99-5, Asian Institute of Technology, Bangkok, Thailand.

www.sun.com, 1999 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054, USA.

CC