EXPERIENTIAL LEARNING ENVIRONMENTS FOR STRUCTURAL BEHAVIOR

By

Victor Viteri

S.B. Civil Engineering
Massachusetts Institute of Technology, 1988
S.M. Civil Engineering
Massachusetts Institute of Technology, 1989

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Civil Engineering

at the

Massachusetts Institute of Technology

February 2003

Signature of Author .....
Department of Civil and Environmental Engineering
September 30, 2002

Certified by .....
Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .....
Oral Buyukozturk
Professor of Civil and Environmental Engineering
Chairman, Departmental Committee on Graduate Studies

EXPERIENTIAL LEARNING ENVIRONMENTS FOR STRUCTURAL BEHAVIOR

By
Victor Viteri

Submitted to the Department of Civil and Environmental Engineering on September 30, 2002 in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Civil Engineering

Abstract

Computers have dramatically changed the nature of structural engineering practice by shifting work from process-oriented tasks, such as performing hand calculations and drafting, to structural modeling and interpretation of computer results. To perform these new functions, engineers need to have a deep understanding of structural behavior. There exists, however, a strong sentiment that engineering education has not been able to effectively address these new needs.

Through the Experiential Learning Cycle, this research identifies structural behavior as knowledge associated with conceptual understanding that is most effectively attained through reflective action. Within this context, an epistemological analysis reveals that the use of computer applications that integrate simulation and assessment capabilities would enhance the traditional framework of structural engineering education. The Tutorial Cycle provides the pedagogical foundation for the development of a methodology that leads to the creation of such environments.

The principles proposed by this methodology were implemented in the building of an Experiential Learning Environment whose primary objective is to help develop a qualitative understanding for the structural behavior of beams. In order to address conceptual and developmental issues, a preliminary environment, PointLoad, with limited capabilities was first put together. Then, after conducting usability tests on this version, a more comprehensive learning environment was built. The end result, iBeam, combines a Finite Element based interactive simulator with an adaptive assessment and feedback component. A programming strategy that combines deterministic programming with rule-based reasoning agents was adopted in the development this environment. Feedback from student trials confirmed that the approach used in iBeam effectively fosters user engagement in reflective action.

Thesis Supervisor: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

ACKNOWLEDGMENTS

I would like to thank the following people and groups for their generous support which made this work possible

- Professor J. Connor for his wise guidance, encouraging advice, and patience; but most importantly for being there for me during my stay at MIT.

- The members of my Thesis Committee in particular to Professor H. Einstein for his excellent advice, direction and support.

- Dr. Santiago Gangotena and Dr. Carlos Montufar at Universidad San Francisco de Quito for their trust, encouragement and friendship.

- To my friends at MIT – The Swimming Washouts, The Intelligent Engineering Systems Laboratory

- The people at the Department of Civil and Environmental Engineering

- My Father and Mother for their love and encouragement

This thesis is dedicated to my wife,

Jane

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 MOTIVATION

Civil structures are often large and complex so their mechanical representations usually result in large mathematical models. Before the advent of the digital computer, these models had to be solved largely by hand requiring considerable effort. To minimize this labor, engineers developed specialized methods that facilitated the process of working out the numerical response of a loaded structure. Some of these methods make simplifying assumptions about the structural system, turning the model into a simpler mathematical representation. Other methods apply heuristics and make qualitative predictions as to how the structure would respond. This information is then used to simplify the solution process, thus arriving at an approximate but reliable answer. In addition to providing a means of getting the numerical solution of systems that would most likely be impossible to solve by hand, in an indirect way, some of these methods give an insight into structural behavior by forcing engineers to develop a sense for how the structure would respond under a particular load.

The introduction of the computer and the development of specialized structural engineering software have changed the way structural engineers do their work. The finite element method (FEM), introduced during the sixties, allows the systematic representation of just about any type of structural system as a set matrix equations. This revolutionary method combined with virtually unlimited computational capacity and the availability of user friendly software have made it possible to almost effortlessly obtain the numerical response of any structural model. As a result, the nature of structural engineering work has shifted from process-oriented tasks such as

performing hand calculations and drafting to structural modeling and interpretation of computer results.

To perform these new functions, engineers need to have a deep understanding of structural behavior. For instance, to correctly define the boundary conditions of a digital model, an engineer must possess a physical understanding of how the constituent parts of a system interact to resist a load. This knowledge is also needed to abstract the digital representation that best replicates the behavior of the physical structural system. Additionally, although computers can handle large and complex models, to get a clear picture of the structural response, it is often necessary to simplify the input. Engineers need to learn how to qualitatively identify the key elements and eliminate the non-essential features of a structure before modeling it. This kind of conceptual understanding is also required for result interpretation. Computer programs can provide an overwhelming amount of both graphical and numerical result information. Understanding the nature of this information is possible with a great degree of technical wisdom, but short of repeating the calculations, there is no easy way of validating the numerical output. As a result, to confirm computer generated results, a capable engineer must qualitatively predict the response of the structure and then compare this prediction to the response patterns provided by the computer.

Although the engineering work environment has changed, there exists a strong sentiment, shared by educators and professionals, that there is still too much emphasis being placed on teaching procedural analysis and not enough placed on attempting to teach behavior (Shepherdson, 2001). This thesis presented is based on the belief that the present framework of engineering education does not adequately support the teaching of structural behavior and that the use of specialized computer based learning tools having tutoring and simulation capabilities integrated in a virtual

environment would effectively assist in teaching structural behavior. A methodology for the development of such environments is also presented in this work.

1.2 COMPUTER BASED LEARNING TOOLS

Computer based learning tools can be broadly divided into intelligent tutoring systems (ITS) and Learning Environments. This categorization is based on the intended purpose of these tools. While ITS attempt to teach and motivate learning using computers (Beck et al, 2002), Learning Environments aim to support learning without direct didactic intervention.

1.2.1 INTELLIGENT TUTORING SYSTEMS

ITS have their origins in the computer assisted instructional (CAI) systems and the Computer based Training (CBT) systems first deployed during the 1960's. The first CAI and CAT systems consisted of automatic instructional flash card systems. Assessment techniques consisted of presenting a problem, receiving and recording the response, and tabulating student performance on the task. Because of computing power and programming limitations, these systems did not explicitly address pedagogical or domain dependent issues. During the late 1960's and early 1970's, CAI systems started to model the student responses and to alter the presentation of instruction accordingly. Although quite limited in scope, some of these systems proved effective at improving basic skills and factual recall (Beck et al., 1996). As computing power and programming capabilities improved, the goals of instructional systems became more ambitious. In 1982, Sleeman and Brown coined the term Intelligent Tutoring Systems (ITS) (Sleeman & Brown, 1982) to describe emerging "smarter" computer programs that incorporated pedagogical teaching strategies and could interactively generate teaching content. These programs would also assess student performance and infer about her/his mastery of specific topics in order to

adjust the style and/or the content of instruction (Murray, 1999). Incorporating this capacity to adapt and generate feedback was aimed at transforming computer educational programs from simple sources of information into true learning enabling tools.

1.2.2 LEARNING ENVIRONMENTS

Learning environments are computer based systems that aim to enhance but do not attempt to explicitly control the learning process. They tend to be domain specific and can be quite sophisticated. These systems can been categorized as Modeling Systems, Simulations, Object Worlds and Environments. A taxonomy based on the type of learning support they provide categorizes Learning Environments as Modeling Systems, Simulations and Object Worlds (Sellman, 1992).

- Modeling Systems consist of software platforms that allow the user to describe and manipulate a mathematical model of a system. MATLAB is an example.
- Simulations consist of platforms of a mathematical representation of a system that can be manipulated by the user through specific variables. Unlike Modeling Systems, Simulations do not allow the user to explicitly access the mathematical model.
- Object Worlds consist of transitional objects that have a direct representation of its state in some visual form and are manipulated by a specialized programming language. Microworlds is an instance of these objects
- Environments define groups of objects similar to those in Object Worlds, but were the rules of the system is accessible and can be changed by the user.

1.2.3 ITS VERSUS LEARNING ENVIRONMENTS

Although Learning Environments are widely used as teaching aids, the use of computers for tutoring purposes is still quite limited despite years of research in the development of ITSs. It appears that too much emphasis placed on research driven

by technical possibilities and on the elusive goal of accurately modeling the user has resulted in overly ambitious projects that fail to deliver substantial results. Another reason for the limited success of ITS is their bold approach. They are designed top-down and meant to be comprehensive. It is virtually impossible to conceive of a monolithic system that can withstand the diversity of users and multiplicity of knowledge representation levels ITS researchers have attempted to accomplish (Patel & Kinshuk, 1996 b).

Nevertheless, computers have shown to posses unique capabilities that could be effectively used to develop virtual tools that explicitly support conceptual learning. However, in order to appreciate how these tools would help in teaching structural behavior and to characterize their precise nature, it is necessary to first understand the cognitive framework of structural engineering and the fundamental nature of knowledge. In addition, in the absence of an ITS capable of fully replacing human interaction; it is essential to view any learning environment as a joint cognitive system that includes teachers, tutoring software, books and students (Teodoro, 1993; Dalal & Kasper, 1994). It has also become apparent that it is more effective to develop environments where the student is encouraged to explore the characteristics of a domain than it is to try to develop an ITS that attempts to comprehend and manage mental processes of students with diverse learning styles and backgrounds (Patel & Kinshuk , 1996 a).

1.3 SCOPE OF WORK IN THESIS

The following chapters first describe a cognitive framework for structural engineering knowledge. Within this framework, the understanding of structural behavior is characterized as conceptual knowledge that is best acquired through reflective action. This analysis also identifies the learning processes that are associated with conceptual learning. An epistemological study then reveals that a tutoring strategy based on

problem presentation and feedback can be implemented in combination with interactive simulations to foster these reflective learning processes. These ideas are then compiled into a methodology to guide the development of simulation based explorative tutoring environments or Experiential Learning Environments.

The second section of this thesis, describes how this methodology is implemented in the development of an Experiential Learning Environment that supports the understanding of the structural behavior of beams. This effort resulted in the realization of two applications, PointLoad and iBeam. Pointload was first developed to calibrate the principles proposed by the methodology. iBeam was then built as the final product. The characteristics and capabilities of these environments are also described in the second section of this thesis.

CHAPTER 2. STRUCTURAL ENGINEERING KNOWLEDGE

According to Cumming (1993), the fundamental issues that need to be considered when developing a virtual tutoring environment are: what constitutes knowledge; what are the methods for knowledge acquisition; how suitable are computers for knowledge transfer; and how to test, using computers, the level of acquired knowledge. The following sections will explore these issues and develop a version of the cognitive framework of structural engineering.

2.1 EXPERIENTIAL LEARNING

Structural engineering knowledge consists of concrete and formal information abstracted from experimental observations and analytical processes. This knowledge is represented by mathematical methods, formulae, and heuristic rules, with the primary objective of providing means of solving existing problems. Because of its empirical nature, the learning of Structural Engineering appears to best described by the processes of the Experiential Learning Cycle. These processes which stress the role of experience on the acquisition of knowledge were first proposed by David Kolb (1984) and are based on the intellectual work postulated by Lewin, Dewey and Piaget (Kolb, 1984, pp 1-19).

Experiential learning proposes that new wisdom and skills are achieved through the mutual resolution of dialectically opposed learning processes, which involve grasping and transforming experiences into knowledge (Figure 1). This theory also proposes

that learning is a continuous process that amalgamates four experiential learning modes: concrete experience (CE), reflective observation (RO), abstract conceptualization (AC), and active experimentation (AE). CE and AC represent two different and opposed approaches to grasping experiences. AC is associated with the conceptual interpretation and the symbolic representation of events in a process termed comprehension. CE, on the other hand, is associated with the assimilation of the tangible qualities of immediate experiences in a process termed apprehension. To create knowledge, grasped experiences must be transformed either through internal reflection, in a process called intention, or through active manipulation in a process called extension (Kolb, 1984, pp. 19-131).

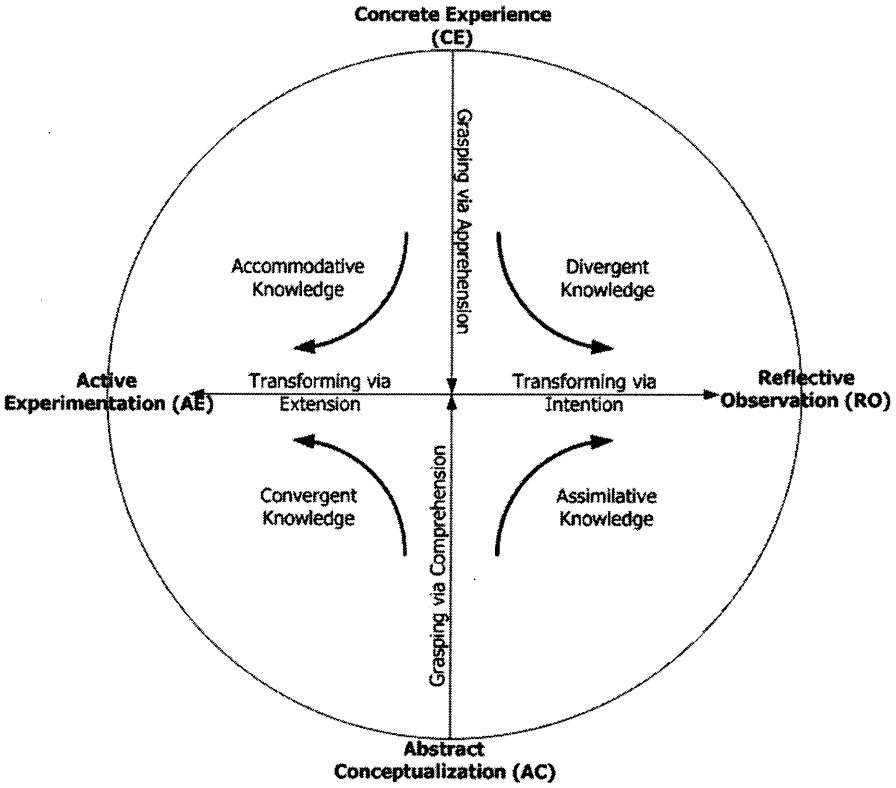


Figure 1 Dimensions of the process of Experiential Learning

Each of the four possible grasp/transformation actions results in different forms of knowledge. Divergent knowledge results form experience grasped through apprehension and transformed through intention (CE-RO). Assimilative knowledge results from grasping by way of comprehension and transformation via intention (AC-RO). Experience grasped through comprehension and transformed via extension results in convergent knowledge (AC-AE) while transformation via intention of apprehension experiences results in accommodative knowledge (CE-AE). Figure 1 illustrates these processes and their relationship with the experiential learning modes, while Table 1 describes the characteristics of each of these forms of knowledge.

TABLE 1 LEARNING STYLES OF THE EXPERIENTIAL LEARNING CYCLE (KOLB, 1984).

Divergent learning, which emphasizes CE and RO, views concrete experiences from many perspectives and organizes many relationships into meaningful, functional integrated units. The strength of this approach lies in imaginative (feeling) ability and awareness of meaning and values.
Assimilative learning, built from AC and RO modes, creates theoretical models from the integration of disparate observations and concepts into logical and precise generalized explanations.
Convergent learning captures the abilities of AC and AE and through hypothetical-deductive reasoning focuses generalized knowledge on specific tasks.
Accommodative learning emphasizes CE and AE and capitalizes on concrete information in order to solve problems in a trial and error fashion rather than on analytical impetus.

Kolb argues that although domain, purpose and learner preferences may define a tendency to a particular learning mode, the combination of all four elementary forms of knowledge results in the highest level of understanding. Therefore, an effective learner must be able to get involved fully, openly and without bias in new experiences (CE); reflect on and observe these experiences from different perspectives (RO); generate concepts that integrate these observations into logical theories (AC) and finally use these theories to make decisions and solve problems (AE) (Kolb 1984, pp. 19-39).

2.1.1 EXPERIENTIAL LEARNING CYCLE IN STRUCTURAL ENGINEERING

In the case of structural engineering, the need to solve a new problem or the desire to explain observed phenomena leads to a learning process that incorporates all stages of the Experiential Learning cycle. This process could start with a period of experimental observations. Analysis of the observed results would lead to the recognition of meaningful response patterns (CE – RO through divergent action). These steps are then complemented by compiling and assimilating these observations into laws and concepts through the application of fundamental scientific principles (AC-RO through assimilation). These laws, alone or integrated with previously known principles, would then form the basis of new analytical methods of solution for engineering problems (AC-AE through convergence). In cases where the system is too complex to be solely described by analytical laws, a process of accommodation supported by direct observations would lead to approximate solution methodologies (CE-AE through accommodation). See Figure 2.

The development of beam theory for straight prismatic members and its applications can be used to illustrate these processes. Direct experimental observations reveal that upon loading an initially straight beam member experiences transverse displacements and displays a deformed shape with a smoothly varying curvature. It can also be

assumed that these displacements are often small in relation to the geometrical dimensions of the beam so the curvature stays small as well. As a result, it is assumed that plane sections remain plane and that the sections remain normal to the deformed beam longitudinal axis (CE-RO). These observations and these assumptions are then integrated into a geometrical model that establishes a relationship between the rate of change of the curvature at a point along the beam and the longitudinal strains experienced by the corresponding cross section. This information combined with principles of equilibrium and elastic theory then leads to the development of beam constitutive equations (AC-RO). These equations can then be used to predict the structural response of a beam under any arbitrary loading configuration (AC-AE). Finally, in cases where an analytical solution is not possible or impractical, simplifying assumptions based on direct observation may lead to efficient and effective heuristic methods of solution.

Through this example, it is evident that the learning processes that result in Divergent and Assimilative Knowledge (transformations of apprehension and comprehension by intention) lead to a conceptual understanding of structural behavior. Convergent Knowledge and Accommodative Knowledge (transformation of apprehension and comprehension via extension), on the other hand, provide analytically and heuristically based problem solving procedures.

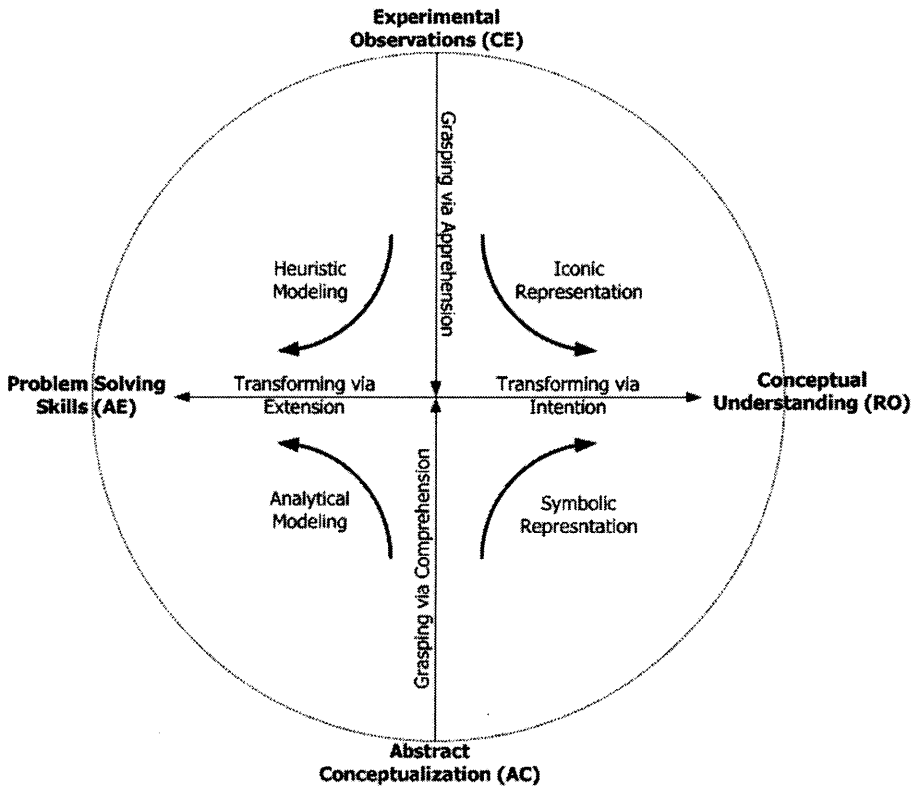


Figure 2 Dimensions of the Experiential Learning Process in Structural Engineering

ISSUES FOR UNDERGRADUATE STRUCTURAL ENGINEERING EDUCATION

Undergraduate structural engineering education aims to provide a framework where acquired knowledge is systematically transmitted in a process that recreates the experiential learning cycle. Classroom instruction provides fundamental theoretical principles and facts relative to the performance of physical systems. It then combines this information with results form experimental observation, and through inductive reasoning transforms all this input into concepts and behavior defining models. Through deductive reasoning, this knowledge is then used to derive analytical methods of solution. Problem sets and course projects are then assigned to motivate and provide the opportunity to put these skills into practice. Classroom instruction,

therefore, primarily facilitates comprehension and motivates assimilative and convergent knowledge acquisition.

Illustrations of the physical response of structural systems motivate and facilitate engagement in CE and divergent knowledge acquisition. Universities rely on guided laboratory experiments to carry out these demonstrations. However, civil structures tend to be large and complex, as a result, these experiments need to be scaled down. Additionally, performing physical experiments is often time consuming and processing experimental data for analysis is frequently cumbersome. These factors limit the effectiveness, frequency and the efficacy of laboratory sessions. As a result, the learning process reaches a bottleneck that seriously restricts the opportunities of successfully correlating structural input with meaningfully response patterns and consequently curtails the accumulation of Divergent Knowledge. Furthermore, since the analysis and interpretation of experimental data reinforces and validates theoretical concepts, experimental complexities compromise the capacity to abstract concepts from observation thus limiting the development of Assimilative Knowledge. Conceptual understanding is intimately related to Assimilative and Divergent Knowledge. As a result, these limitations make it difficult for student to develop a conceptual appreciation for structural behavior concepts.

A solution to the problems posed by laboratory based experiments has been sought in the use of commercial structural engineering software and of specialized simulators that illustrate structural response through virtual models. Computer simulations can create environments where students can easily explore the response of structural systems and discover the properties of a domain. Contemporary theories of learning such as *constructivism,* state that knowledge acquired in simulation based exploratory environments is deeply rooted in the learner's knowledge structure (Jonassen, 1991). As a result, it can be safely hypothesized that some of these efforts have had a positive impact on engineering education. However, it has also become clear that

offering simulations without offering additional support may result in the learner getting lost in the simulation environment and not learning very much (Joolingen & de Jong, 1993). Additionally, it has been shown that the mindset of most learners does not change as a result of engaging exclusively in simulations (Kashihara et al., 1994). Since knowledge creation requires that learners progress from concrete explorations in meaningful contexts to symbolic representations of these actions and then on to (the creation of) abstract models (Duffy & Jonassen, 1991), environments that support learning must not only be passive exploratory vehicles as it is the case of stand alone simulators. These environments must also provide an active framework that supports the transformation of external input into knowledge.

2.1.2 STRUCTURE OF KNOWLEDGE

In order to characterize the features that would result in an effective learning environment and to determine the supporting framework that must be provided, it is necessary to understand the fundamental structure of knowledge and how it applies to and is represented in structural engineering and the Experiential Learning Cycle. This understanding will be instrumental in identifying the specific processes that lead to the knowledge transformations associated with the conceptual understanding of structural behavior and in identifying how computers can be used to support these processes.

THE CONSTITUENTS OF KNOWLEDGE

At its most fundamental level, knowledge consists of factual free standing cognitive entities or variables. These facts interconnect through functional relationships producing larger and more complex cognitive clusters or concepts. These cognitive items are interwoven to form multi-layered networks of interconnected knowledge, which is characterized by its complexity or depth and by its extent or breath. The size of the network determines the breath while the nature or intensity of the interconnections characterizes the depth of knowledge (Kinshuk & Patel, 1996 b).

A hypothesis is the cognitive entity that describes a possible functional relationship between two or more conceptual variables. The collection of all the rules that can possibly describe the observable phenomena of a given domain is called the Hypothesis Space. Hypothesis spaces have a variable dimension and a relation dimension (Joolingen & de Jong, 1993). These dimensions are organized as hierarchical subspaces.

In the relational subspace the hierarchy is determined by the precision of the relationships. Three levels of precision have been proposed: quantitative numerical precision at the lowest hierarchical level, followed by quantitative relational and then qualitative relational precision at the highest hierarchical level (Opwis, 1993). For example, if A and B are two related variables, stating that if A grows, B grows as well, constitutes a hypothesis of qualitative relational precision. Stating that B is proportional to A constitutes a hypothesis of quantitative relational precision. Finally asserting that B is equal to 3 times A is a hypothesis of quantitative numerical precision.

The variable subspace is organized according to the generality of the variables. The most general variables are positioned at the highest hierarchical level while the lowest hierarchy corresponds to instantiated variables. Hypotheses that contain general variables have a wide range of applications while those containing instantiated variables that represent allow for more precise definitions (Joolingen & de Jong, 1993). For example, the geometrical profile of the moment diagram of a beam which describes the distribution bending forces along a beam is an example of a relationship based on a general variable. The magnitude of the moment at a point on a beam is, on the other hand, an precise definition based on a specific value that represent a relationship at a specific point, in other words, an instantiation. See Figure 3.

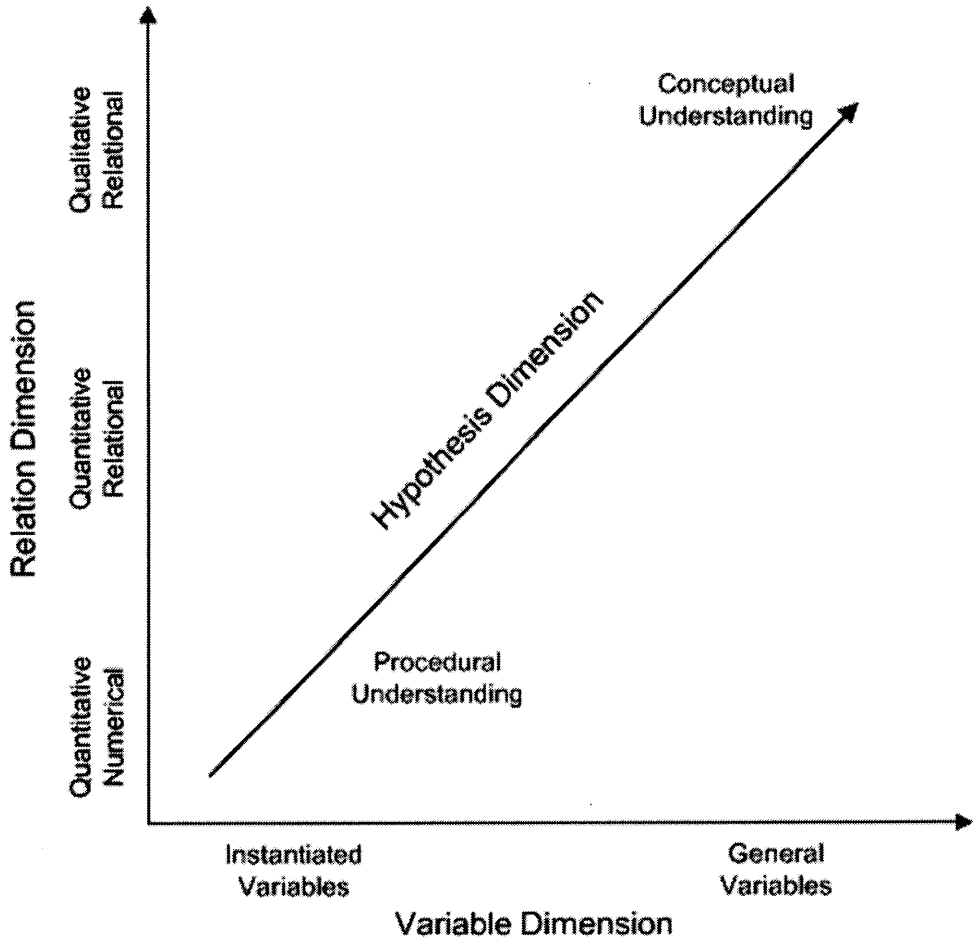


Figure 3 Dimensions of the Hypothesis Space

For deep understanding, a learner has to discover and integrate the rules that make up the Hypothesis Space (Opwis, 1993). However, in most cases, for conceptual understanding, the most significant relations to be discovered are not the exact quantitative, programmed variable interactions but rather the general associations that represent a more abstract, often qualitative appreciation of the domain (Billet & Rose, 1999). These associations are found in the upper hierarchical range of the hypothesis space.

Since it is based on hypotheses built on qualitative associations of generalized knowledge entities, conceptual understanding is flexible and as such its representations can be rearranged to predict behavior in new situations (Joolingen & de Jong, 1993). Additionally, conceptual knowledge provides the preconditions on which quantitative knowledge can legitimately be applied (Opwis, 1993).

KNOWLEDGE REPRESENTATION OF STRUCTURAL BEHAVIOR

Structural behavior is the physical response of a force resisting system to a set of prescribed loads. Response measures, such as deformations, reaction and member forces, describe the effect of these loads on the individual components of the structure and constitute the fundamental entities of engineering knowledge. Engineers have developed analytical methods that result in the numerical prediction of these measures. To facilitate interpretation, these results are often arranged into patterns, such as deflected shape, free body, moment, and shear diagrams.

A deflected shape diagram consists of an illustration of the deformations experienced by the structure in response to a loading. Although the magnitude of the deformations often needs to be super-scaled, these diagrams constitute direct pictorial representations of structural response. Free body, Moment and Shear diagrams respectively are visual representations of the reaction forces, shear forces and moments developed by the structure in response to the loading. Since abstract entities such as forces and moments do not have a concrete physical dimension, these later diagrams constitute symbolic representations of structural response.

In addition to synthesizing information and providing pictorial and symbolic representations of structural behavior, response patterns highlight the relationships that exist between each of the response measures and the make up of the structural system. Additionally, these diagrams portray the correlations that occur among these response measures. Consequently, patterns of response reveal the conceptual

relational framework that supports structural engineering knowledge. As a result the ability to interpret this information leads to a conceptual understanding of structural behavior and allows the qualitative prediction of the response of even complex systems.

CHAPTER 3. SIMULATION BASED EXPLORATORY TUTORING ENVIRONMENTS

Neither traditional ITS nor standard Learning Environments are sufficient to support learning. The Experiential Learning Cycle shows that effective knowledge acquisition requires a combination of the sorts of direct experience that can be gained from simulations with the more interventionist approaches that have traditionally been used in intelligent tutoring. This combination leads to a three-way interaction between a tutor (virtual), the learner and a simulation environment in what has been termed a Simulation Based Exploratory Tutoring Environment. The most important functional features of these environments are that both tutor and learner have access and are capable of manipulating the environment and to observe changes. Additionally there is a direct interaction between the tutor and learner about the events that are happening in that environment (Elsom-Cook, 1993).

Exploratory features in a Learning Environment, allow students to get a strong degree of familiarization with the basic ideas of the domain under study. According to Teodoro Duarte, through simulation software, students can see many situations, explore what happens under different conditions and reflect about what happens if they change conditions, thus becoming progressively more familiar with the ideas, the consequences of ideas and representations of the domain. When they become familiar with new ideas and new representations, learners can establish more meaningful relations with the ideas they already have (Teodoro, 1993).

In combining the exploratory environment with a tutor, the intention is to assist the learning experience without detracting from the discovery and experiential effects of the simulation alone. This requires from the tutor to not only have knowledge about

the nature of the simulations, but also of knowing ways of using this knowledge so as to capably decide which interventions are appropriate and which are unnecessary (Elsom-Cook, 1993). In order to achieve this balance, it becomes essential to define the makeup of the Exploratory Environment in accordance with the characteristics of the domain and of the tutoring support that is sought. The following sections explore these issues placing emphasis in the conceptual understanding of structural behavior.

3.1 INTERACTIVE SIMULATIONS

An exploratory environment that aims to foster qualitative learning must be capable of representing the interplay between the fundamental variables of the domain and the relationships that describe its hypothesis space or concepts. The unique capacity that computers have to portray otherwise abstract objects and illustrate and link multiple representations of an event, makes simulation based exploratory software ideally suited for this task. Seymour Papert (1980) was among the first to identify how these unique capacities for representation could become used to enhance the learning process. Papert argues that exploratory computer based tools can overcome the boundary between lower and higher cognitive stages by allowing learners to approach the formal (or abstract) in a concrete way: *Stated more simply, my conjecture is that the computer can concretize (and personalize) the formal. Seen in this light, it is not just another powerful educational tool. It is unique in providing us with the means of addressing what Piaget and many others see as the obstacle which is overcome in the passage from child to adult thinking. I believe that it can allow us to shift the boundary separating concrete and formal. Knowledge that is accessible only through formal processes can now be approached concretely. And the real magic comes from the fact that this knowledge includes those elements one needs to become a formal thinker* (Papert, 1980, p. 21).

In this context Teodoro Duarte proposed the existence of three types of objects unique in a computer exploratory environment. Type I objects refer to real entities such a structural support or a load while type II objects describe purely conceptual objects that have no perceptual fidelity, for instance, a reaction force or the stress distribution across a beam cross section. Type III objects represent the functional relations between type I and type II objects (Teodoro, 1993). In other words, type III objects represent the hypotheses space of a given domain.

In a computer environment type II and Type III objects acquire a real dimension in the sense that they can be represented and manipulated as real objects, on the screen, while still being abstract physical-mathematical constructs. The basic task for a learner in an exploratory environment is to determine the nature and form of type III objects. This type of qualitative learning is associated with meaning not on algorithms and requires establishing the relations between different representations of the simulated phenomena. Repeated exposure to different instances of an event facilitates this process of creating meaning from representation. Therefore, one of the most important features of a computer exploratory environment is that it should allow the user to explore the relations between the different kinds of objects, in real time or with different time scale, and under full user control. This forces the learner to establish strategies to explore and visualize the domain (Teodoro, 1993). Therefore, it becomes apparent that it is necessary to build a simulation model capable of rapidly predicting all the essential features of the domain. In addition, it is essential to base the simulation output on powerful visual representations capable of revealing the character of these features.

3.2 ASSESSMENT AND FEEDBACK

Although exploratory software can be very powerful, learners can only explore what they already know (Njoo & de Jong, 1993) and deep understanding, which comes

about through rule discovery (discovery learning), is unlikely to occur if engaging exclusively in simulations. In order to understand these limitations, it is necessary to investigate the properties of the hypothesis space.

The *Universal Hypothesis Space* consists of all the theoretically possible theories that can be constructed. Within this space lie the *Learner Hypothesis Space*, and the *Conceptual Model*. The Learner Hypothesis Space consists of all the hypotheses the learner knows about or has been told to exist while the Conceptual Model is the set of all the true hypothesis of the domain at hand. The *Learner Search Space* is a subset of the Learner Hypothesis Space and consists of the hypothesis the learner understands or has discovered and considers to describe the domain (Joolingen & de Jong, 1993). Figure 4 shows these relationships.

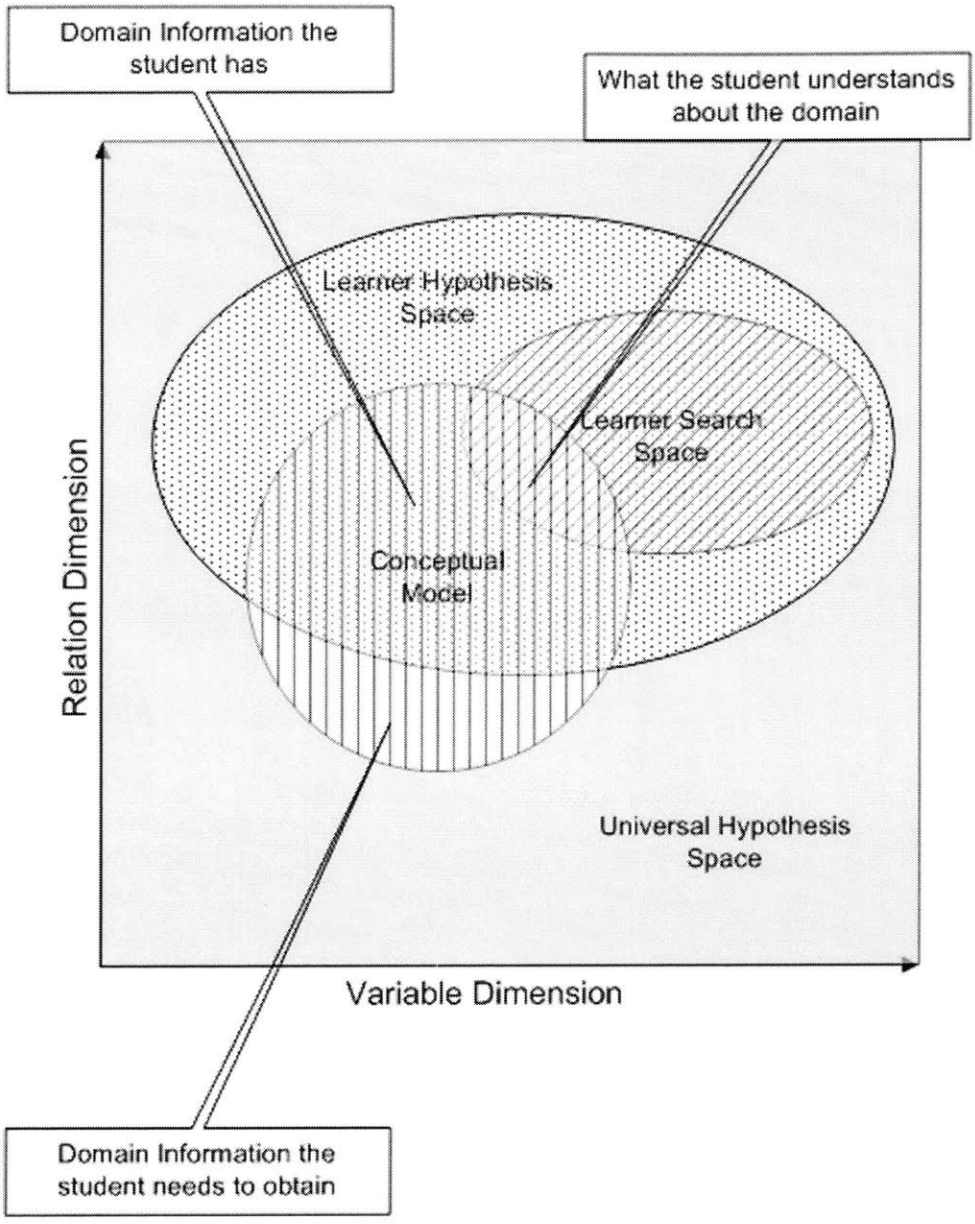


Figure 4 Hypothesis Space: Partial Understanding of the Domain

When engaging in a simulation, since the computer performs all the operations and establishes all the links between the variables of the problem being addressed, the learner is unlikely to explore beyond her search space to explain the observed results. Passive interaction with a simulator would at most reinforce or confirm the

hypotheses the learner has already discovered. Even if an effort is made to explicitly understand the solution, it does not mean that the learner has acquired deep knowledge. What is more important, even learners with incorrect representations of the domain, would justify the simulation output to reconfirm their beliefs.

If the Conceptual Model lies partially or wholly outside the learner search space, in order to attain full deep knowledge of the domain, the student will have to actively engage in a process of constructing knowledge (constructivism). This process demands finding directions for assembling known bits of knowledge in new and significant ways.

For this to occur, two conditions must be met: the conceptual model must be part of the learner hypothesis space and the learner must be confronted with the need to stretch her search space. Only then, can computers become instrumental in acquiring new knowledge by allowing the user to integrate known information (learner hypothesis space) with the results of the simulations into behavior defining conceptual representations.

3.2.1 ASSESSMENT

According to Forman and Pufall (1988), constructing knowledge is a process that embodies three stages: inducement of epistemic conflict, self-reflection and self-regulation

- Epistemic conflict involves the awareness and the internal desire to resolve cognitive uncertainty
- Self-reflection is construed as a response to conflict.
- Self-regulation is the developmental restructuring of thought.

This principle was also stated by John Dewey: *Conflict is the gadfly of thought. It stirs us to observation and memory. It instigates invention. It shocks us out of sheep-*

like passivity, and sets us at noting and contriving...conflict is a sine qua non of reflection and ingenuity.

Within the context of a simulation based exploratory environment, domain knowledge is made explicit by illustrating solution sets of simulated problem setups. In addition, Structural Engineering Knowledge becomes meaningful when providing solutions to existing problems. Therefore, the motivation for deeper understanding and the source of epistemic conflict could also be sought out in problem solving. . Self-reflection would occur during and as a result of the problem solving process while self-regulation would come about when as the learner integrates newly discovered hypotheses within her cognitive structure. This is supported by Boder and Cavallo's quotation of G. Cellerier that *it is only when previously acquired knowledge effectively interacts with a material or symbolic problem universe during its application to a problem that schemes are accommodated, and that new constructs result from empirical or reflexive abstraction, which in turn permit new potential applications and problem formulations. Thus acquisition serves application and conversely*. (Boder & Cavallo, 1991).

When confronted with a problem outside the Search Space, solutions are unlikely to come by following algorithmic processes. Experienced learners would first try to establish a course of action based on understood representations of the domain. If this fails, other representations may be attempted. Sophisticated learners would, in addition, engage in a heuristic process that involves recombining their knowledge and integrating new facts (from her Hypothesis Space) into new representations. Consequently, while it's elementary pieces may be universal, when the solution is finally found, its representation is unique and might only be useful to the person who *discovered* it. Simulation based learning environments can stimulate this discovery process by allowing the student to interact freely with a representation of the problem

while attempting to generate a solution. Feedback can further facilitate the knowledge building process

3.2.2 FEEDBACK

Two basic types of feedback strategies can be given to a learner. The first type of feedback focuses on the correct way of attaining an immediate goal. This feedback mode requires constant monitoring and immediate feedback, which prevents the student from proceeding along a wrong path. This type of feedback is suitable for novices as it allows learning in small doses. This approach, however, does not make a distinction between minor and major errors.(Boder & Cavallo, 1991)

The second type is delayed feedback based on diagnostics. Since feedback is deferred, it has to deal with a range of mistakes so it provides advanced learners with an opportunity receive and or to explore the why of a knowledge entity. For a novice, this type of feedback could prove to be burdensome as terms that are not fully understood are used to explain other terms and as the focus of feedback shifts from correct relations to mistakes. The extent of the delay is also critical. A student, who repeatedly commits a mistake before being advised, will have to unlearn what has been reinforced by repetition. (Patel & Kinshuk, 1996 b).

FEEDBACK STRATEGIES

Since, computers lack the linguistic flexibility needed for analogy and explanation, a computer based tutoring environment should minimize narrative feedback. Instead, feedback should capitalize on the capabilities that computers have to illustrate and simulate structural response. In addition, it does not make sense to offer full solution schemes since in most cases they will not fit the learner's representation of the problem. Mistakes are part of a structure the learner brings to the problem solving process. If the student representation of the problem domain is incorrect, then explanations are unlikely to be properly parsed and would not be incorporated in the

student's cognitive structure (Boder & Cavallo, 1991). Effective tutoring should identify the universal basic inadequacies in the response and allow the learner assimilate corrective suggestions into her rule discovery process. As a result, in a computer environment, immediate feedback is often more appropriate and more practical to implement. To allow the necessary interventions, however, immediate feedback requires engaging the student in an interactive problem solving process. After the student has finished this process, delayed feedback could be offered if necessary.

3.2.3 SUPPLEMENTING INFORMATION

The two types of knowledge represented in memory, conceptual knowledge and procedural knowledge are associated with different kinds of information. Conceptual knowledge is dependent on know-why information while Procedural knowledge is supported by know-how information. Know-how information is associated with methods that provide the means to secure goals. Because of its operational orientation, this information can be transmitted without much reliance on linguistic expressions. On the other hand, know-why information has a causal orientation. It is mainly reflection driven and based on abstraction. Passing on this type of information requires the reconstruction of reasoning processes which requires good communication channels and often needs employing specialized linguistic forms, metaphors and symbolic representations (Patel & Kinshuk, 1996 b).

As it was argued before, in order to benefit from the interaction within the exploratory environment, the Conceptual Model must be a subset of the Learner Hypothesis Space. If this is not the case, the student must seek to complement her information base (Figure 5). However, Computers lack the capacity for language and linguistic representation (Patel & Kinshuk, 1996 b). As a result the primary source of conceptual information should still be the classroom, text based explanations or a

human tutor. Interaction with tutoring software should then be aimed to draw out misconceptions the student may have and correct them through appropriate feedback.

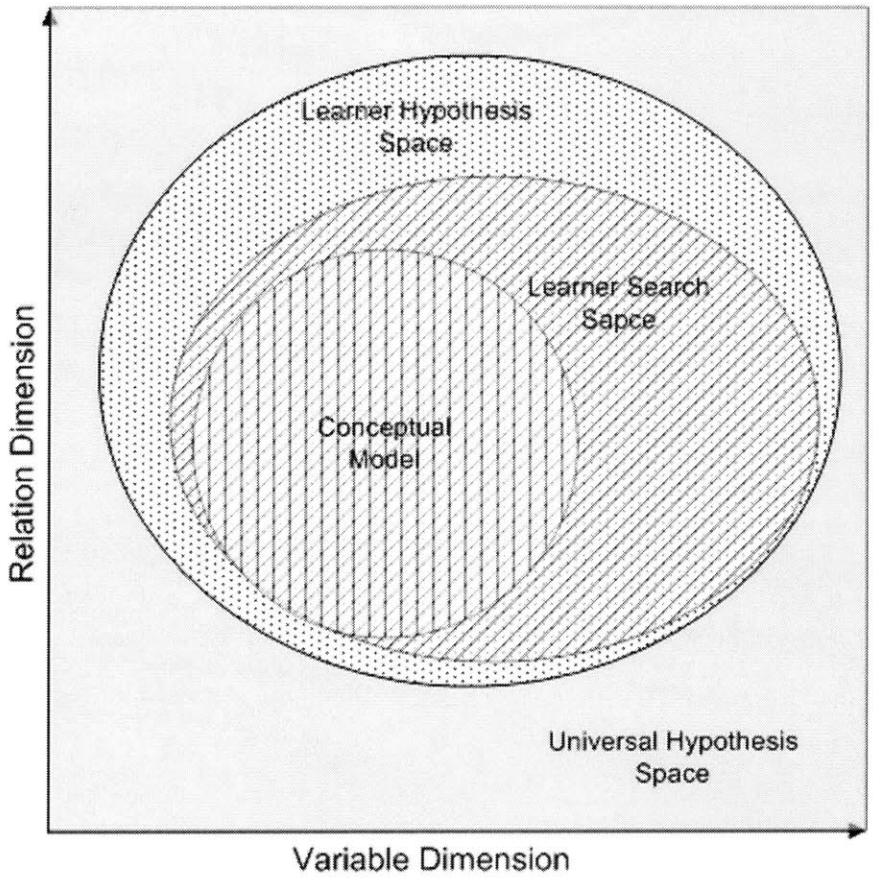


Figure 5 Hypothesis Space: Total Understanding of the Domain

3.3 THE NEED FOR MULTIPLE ENVIRONMENTS

According to its context, knowledge can be clustered and organized into domains or topics. Also, it has been observed that, initially, learning takes place on a number of topics over different domains until the basics of a subject are mastered. Then, over time, through vertical and horizontal integration, both knowledge breadth and knowledge depth increase progressively. Vertical integration involves the repeated

application of knowledge under different circumstances followed by reflecting over the results. Horizontal integration involves linking knowledge from different domains to accomplish a goal (Patel & Kinshuk, 1996 a). Therefore, rather than attempting to develop learning environments that incorporate entire disciplines, more congruous with learning practice and far more practical is to provide multiple specialized settings covering a single or a group of related domain topics. Since these environments are exploratory in nature, vertical integration would come as result of more interaction with the simulation and of progressive assessment; while exposure to different Learning Environments would permit the horizontal integration of knowledge.

CHAPTER 4. METHODOLOGY

4.1 THE TUTORIAL CYCLE

The Tutorial Cycle, developed by George Brackett (Shepherdson, 1998), can provide the pedagogical framework needed to integrate the theoretical principles presented in previous chapters into a methodology that guides the development of simulation-based exploratory learning environments. The Tutorial Cycle defines the process that must be followed to facilitate learning. The underlying idea is to recreate a setting that emulates the interaction between a learner and a highly skilled tutor. According to Brackett an effective tutoring environment must:

- Present information related to the goals.
- Elicit student action towards these goals.
- Assess the student's action.
- Provide feedback.
- Offer strategic guidance.
- Manage and motivate the process.

The description of the Experiential Learning Cycle reveals that in order to grasp an understanding of structural behavior, the educational framework needs to better support the types of learning transformations that lead to conceptual insight and that are associated with Divergent and Assimilative Knowledge. To facilitate divergent learning action, a learning environment must resolve and incorporate all the aspects of the Tutorial Cycle with respect to a process that provides tangible information (Apprehension) and helps organize it into meaningful relationships (Intention). Assimilative learning is supported by theoretical principles and the symbolic

interpretation of concrete experiences. Classroom based instruction imparts most of these principles. Therefore, the learning environment would need to complement the assimilative learning process by providing symbolic representations of structural response and in helping correlate this information with theoretical principles.

The following sections define a methodology that satisfies these requirements.

4.1.1 PRESENT INFORMATION RELATED TO THE GOALS

An Exploratory Tutoring Learning Environment should administer a well-defined topic or a small cluster of related topics that bring together a cohesive set of theoretical ideas or Lesson Objective. In addition to clarifying the purpose of the learning environment, having a well-defined objective is essential when determining the features of the simulation and its interactions, and facilitates the selection of the tutoring strategies. Limiting the scope of a lesson is also important to focus the student's attention and in preventing the user from getting lost or frustrated while navigating the learning environment.

After selecting the topic, it is necessary to define the Conceptual Model. The first step in this endeavor consists in identifying the response measures that are inextricably linked to the Lesson Objective. These measures lack concrete perceptual fidelity so within the simulation environment, they are represented as Type II objects. The next task consists of determining the governing parameters such as boundary conditions and loading configurations that have an influence on these response measures. These governing parameters are concrete entities so they become the type I objects that allow the learner to interact with the simulation. The simulation environment must then capture and make explicit the functional relationships between input and response parameters as these constitute the key concepts and as such are the basis of the Conceptual Model. This step entails determining or developing display patterns and graphical arrangements capable of clearly portraying these associations.

In other words, capture type III objects by cleverly displaying the response of type II objects to changes in type I objects. For instance, if the purpose is to illustrate bending action, the response measures that would be related to this topic are the shear forces, bending moments, reactions, stresses and deformations. These values are in turn dependent on support boundary conditions, load position and magnitude, etc. The functional relationships between these dependent measures and the input parameters are then revealed by the free body, shear, moment and deformation diagrams.

The definition of the key concepts and governing parameters would lead to the development of a simple structural arrangement whose response typifies the Conceptual Model. The purpose of this action is to ensure that all information essential to the lesson is presented and that superfluous details are avoided. In cases where it is not possible or desirable to explore all the relevant issues of a topic within a single environment, the hypotheses that this structural setup is able to bring forth would be a subset of the Conceptual Model and is defined as the *Experiment Hypothesis Space.*

Finally develop a simulation environment that illustrates the response of the selected structure. The primary function of the simulator is to provide illustrations of structural response (concrete experiences) so as to facilitate the discovery of typical response patterns that support the development of meaningful internal representations of structural behavior. Discovery is triggered by repeated exposure from different perspectives to the properties of the domain. As a result, powerful interactive capabilities that allow plentiful opportunity for exploration would greatly enhance this process. Furthermore, to help the student establish typical links, the simulator should interactively respond to changes of the selected governing input parameters and should provide the response in terms of the response measures.

In essence identifying the goals defines the content and objective of the Learning environment. Table 2 summarizes these principles.

TABLE 2 TUTORING ENVIRONMENT CONTENT AND OBJECTIVES

Action	Purpose
Identify the Lesson Objective	Narrow the scope to focus student's attention
Identify the concept(s) associated with Lesson Objective	Determine the content of Lesson Objective.
Identify the input parameters that affect the response	Determine the parameters that control the explorative environment
Identify a simple setup that characterizes the Lesson Objective	Avoid providing excess or irrelevant information.
Develop a layout that clearly portrays the relationship between the response measures and the input parameters, and the relationship that exists among these parameters	Provide clear pictorial illustrations and symbolic representations of the structural response.
Provide strong simulation capabilities	Facilitate exploration

4.1.2 ELICIT STUDENT ACTION TOWARDS THESE GOALS

The information provided by traditional laboratory sessions consists of visual demonstrations as well as numerical data measured in physical experiments. These data often need to be post processed to yield the symbolic representations that support initial physical experimental observations. Like physical experimentation, post processing experimental data is often awkward and time consuming. In an Exploratory Tutoring Environment it is possible to seamlessly integrate data gathering and post processing into powerful graphical representations, thereby bridging the gap between the intuitive and qualitative, and the conceptual and precise. Furthermore, interactive simulation environments afford the capability of repeated

experimentation. These features would be impossible to realize in the traditional laboratory since it would require instantaneous modifications in the setup conditions of often large and cumbersome physical arrangements.

In order to profit fully from these advantages, the features and capabilities of the learning environment should be immediately apparent to the learner. In addition, interaction within the environment should be effortless and appealing. Hidden controls and sophisticated interfaces should be avoided. In addition, although learning environments should be powerful, only features that support knowledge should be enabled. It should be remembered that learning is a process of transforming external information into knowledge. Before understanding a domain, it is difficult for a student to differentiate between relevant and irrelevant information. Consequently only those features and capabilities essential to the learning process should be offered.

The primary objective of the Exploratory Tutoring Learning Environment is to develop a conceptual understanding of structural behavior. According to Opwis, *students should be allowed to start reasoning on that level which is in accordance with their intuitive reasoning of the physical phenomena. Very often, such intuitive reasoning can be described as a kind of qualitative reasoning. Hence, students should first be provided with an accurate qualitative presentation of the domain under study before more sophisticated quantitative reasoning is taught. Quantitative information should then extend the qualitative knowledge in order to enable more precise problem solving* (Opwis, 1993). As a result, the simulation input and output should encourage qualitative thinking over numerical inquiry.

Fundamentally, *eliciting action toward the goals* consists of facilitating the grasping of experiences. Table 3 summarizes the most relevant points associated with this endeavor.

TABLE 3 STEPS THAT FACILITATE THE GRASPING OF EXPERIENCES

Action	Purpose
Carefully select control features and only provide indispensable capabilities	Prevent learner form being distracted by non essential
Provide intuitive and easy to use graphical user interfaces	Encourage interaction
Provide non-dimensional simulation output	Encourage qualitative thinking over numerical inquiry

4.1.3 ASSESS THE STUDENT'S ACTION

The interaction within the tutoring environment should not be guided. Students should be free to interact with the simulator and the explanatory interface at will. The idea is to involve the student fully, openly and without bias in new experiences (Concrete Experience). However, interaction alone is unlikely to result in deep understanding and is prone to lead to unstructured knowledge. An assessment module should provide a framework that encourages and guides reflective action, identifies the learning goals and what is required to fulfill them.

The strategy is to pose problems whose solutions lie outside the student's Learner Search Space. This action generates a conflict that is resolved as the student recombines information from her Learner Hypothesis Space into a new cognitive network that provides or justifies new solutions. This network is entirely made up of facts the student already knows about and probably has used in other already discovered hypotheses. The knowledge gained by the learner is in the novel way these facts are arranged to generate the solution for the problem at hand.

The assessment utility should implement exercises that:

- Support the development of an intuitive sense of structural response
- Lead to an understanding for the functional relations that exist between the input parameters and the structural response measures

- Help in establishing the interconnected hypothesis network that describes the interaction among the various response patterns
- Integrate observed simulation results with the theoretical knowledge in the Learner hypothesis space

To develop an intuitive sense of structural response, assessment problems should promote the integration of facts and ideas relative to the tangible measures of structural response such as displacements and rotations into behavior defining archetypes. This approach would elicit divergent reflective action as concrete experiences are transformed to deliver meaningful patterns in the observable response of a structural system, thus developing the student's capacity of reasoning instinctively about the domain.

Symbolic response patterns such as moment diagrams are abstract mathematical constructs derived from observation and deductive reasoning. However, in the computer environment, these objects acquire a physical dimension. As a result, by engaging in heuristic problem solving actions, the learner can develop an intuitive feeling for the parameters that these patterns represent.

The theory that supports the nature of these symbolic response patterns requires a high level of conceptualization. Therefore, in addition to developing an intuitive feeling, the student needs to reflect upon the meaning of her observations, reason out an explanation and assimilate her conclusions and domain knowledge into general conceptual models. Consequently, assessment exercises should encourage the student to propose and test the validity of the rules that underpin the solution of a given problem.

Like the simulator, to boost understanding of structural behavior, assessment should elicit action over general qualitative hypotheses rather than on specific-algorithmic procedures. Consequently, exercises should stay within what can be qualitatively

explained. For example, questions for which it is essential to have precise numerical intermediate results should not be presented. Rather, the student should be challenged to build solutions based on hypothesis generation and to demonstrate her reflective and heuristic abilities. As a result, in order to support appropriate assessment and feedback actions, the learning environment should be capable of identifying the limits of qualitative and heuristic reasoning. Providing this capability makes it necessary for the environment to have and manipulate a fine grained symbolic representation of the hypothesis of the domain.

Although the primary purpose of the assessment module is not to characterize performance, the assessment module should keep a record of the student interaction with the exercise problems. This record would provide a means to gauge progress against the objectives of the environment, but most importantly, it would be the reference basis to determine the sequence of assessment and the type of feedback the tutor should provide to the learner.

In synthesis, assessing the student serves the purpose of facilitating the transformation of experience into knowledge. Table 4 summarizes the most important points of this actions.

TABLE 4 PROCESSES THAT FACILITATE LEARNING TRANSFORMATIONS

Action	Purpose
Develop and pose assessment exercises that test the learner's ability to identify the relationship between input parameters and the nature of the structural response.	Induce Epistemic Conflict to trigger reflective action.
Develop a hierarchical representation of the knowledge of the domain	Provides a reference framework for assessment and feedback
Provide feedback that guides but does not totally reveal the solutions	Draws out misconceptions while allowing the learner to discover the rules of the domain and built her own solution schemes.

4.1.4 PROVIDE FEEDBACK TO THE STUDENT

Feedback action in response to assessment can be categorized as active or passive. Active feedback action recreates the transformations that are required to arrive at a solution and transmits evaluative or corrective information to the student. Passive action does not provide explicit feedback, but rather identifies the knowledge state of the student and guides the interaction process so as to support learning. In general, it is neither possible nor desirable to prescribe feedback actions. These actions depend on the context, the pedagogical approach and nature of the exercise and their effectiveness varied from learner to learner. It is possible, however, to characterize the competences required for effective feedback within a learning process that encourages understanding the association between representations of behavior and conceptual principles.

Learning is viewed as successive transitions between knowledge states. Therefore effective tutoring must assess current understanding and encourage the transition to a

higher knowledge state. In order to accomplish this task, it is essential to have a hierarchical representation of the knowledge of the domain at hand. This representation becomes the reference framework against which knowledge is categorized and measured. Since hypothesis spaces are hierarchically organized, domain knowledge can be ranked and represented by the rules that make up the Conceptual Model. For conceptual understanding, learning should be guided according to the generality of the rules from the qualitative to the precise, and form the general to the particular. Consequently, assessment exercises and feedback strategies should be organized according to this hierarchical rank.

The Learner's knowledge state is assessed by correlating the assessment responses with the representation of domain knowledge. Feedback should then attempt to help build the hypotheses that lie just outside the learner's knowledge state. Consequently, modeling the student's knowledge is essential for diagnosis (assessment) while modeling and manipulation of the domain knowledge is essential for didactic support (feedback). Since student knowledge is a subset of the knowledge domain, the student's record must also be a subset of the domain knowledge representation.

The premise that all relevant domain information be known to the student before engaging in a knowledge transformation process makes it futile to give long theoretical corrective explanations if the student's response to a problem is incorrect. Instead, feedback should consist of hints that help the student progressively build solution schemes (Zhou et al., 1999). In addition, whenever possible, feedback should not expose the solution but rather advice the learner. Boder and Cavallo (1991) analysis of Piaget's statement that every time that you teach a child something, you prevent them from learning themselves supports this argument. This analysis states that *...it is better to acquire knowledge by oneself rather than being taught by others; rather, when learned by oneself, one needs to construct heuristics which determine the direction to take. On the other hand when taught by someone else, the*

sequence of steps is already provided and the problem of finding guidance is eluded. Thus, the learner does not have a true sense about why a path is right or wrong (p. 209).

4.1.5 OFFER STRATEGIC GUIDANCE

Although hints should also be available to the student while engaged in assessment exercises, strategic guidance would be largely provided through post assessment active feedback actions. In addition, an explanatory interface should be available during simulations. This utility would characterize the response of the sample structure in terms of governing theoretical principles. This information would be valuable to the student who needs to reinforce his background. Modeling and manipulation of the domain knowledge would be essential to the implementation appropriate guidance schemes.

4.1.6 MANAGE AND MOTIVATE THE PROCESS

The student record should supply the user with information about his/her progress in achieving the goals of the lesson. The student record would also be used to control the difficulty and access to the assessment exercises. This would appear as restrictive, but it is important to insure that the student is not discouraged by questions that are too easy or frustrated by questions that are too difficult. Finally, this record should gather information that can be used for statistic processes when analyzing a group of students. The outcome of these processes could be used to identify weaknesses in other areas of the experiential learning cycle such as in classroom instruction.

In summary, choosing the appropriate feedback action and calibrating the type and level of assessment provide a guiding framework and manages the learning process. Table 5 summarizes the most relevant aspects of these actions.

TABLE 5 OFFERING STRTEGIC GUIDANCE AND MANAGING AND MOTIVATING INTERACTION

Action	Purpose
Implement a student record	Provide a basis on which to determine assessment and feedback actions
Track the interaction in assessment exercises.	Identify areas where the student(s) need more or less information.
Adjusting the exercise complexity level in and control the access to exercises.	Motivate student and avoid frustration.

CHAPTER 5. APPLICATION OF THE METHODOLOGY

5.1 DESCRIPTION

The principles proposed by the methodology of the previous section were implemented in the development of an Experiential Learning Environment whose primary objective is to help develop a qualitative understanding for the structural behavior of beams. This environment is targeted to students who are familiar with the fundamental concepts of beam theory and have already learned analytical methods of solution for simple beam setups.

5.1.1 PROCEDURE

In order to address conceptual and developmental issues on a smaller and more manageable scale, a preliminary environment, PointLoad, with limited capabilities was first built. Then, after conducting usability tests on this application, a more comprehensive learning environment was constituted. The end result, iBeam, consists of an interactive simulator with an adaptive assessment and feedback component. This environment interactively models and illustrates the structural response of multi-span continuous beams and implements an assessment exercise that provides interactive tutoring advice and post evaluation feedback based on diagnostics.

The following sections will first develop concepts derived from beam theory that are relevant to the development of these environments. Then, Chapter 6 describes the

programming strategy adopted in the development of these environments. Next, in Chapter 7, PointLoad will be briefly described followed by an assessment of its capabilities. Finally, Chapter 8 will provide a detailed description of iBeam.

5.2 BEAM BENDING-ACTION

A beam is a structural member designed to resist transverse forces and bending moments applied along its longitudinal axis. Most beams are supported at discrete points along their length. These supports produce reactions that also consist of transverse forces and moments. In response to these loadings, beams develop internal stresses and experience deformations.

Bending action characterizes the behavior of beams, whose primary mechanical response consists of developing a moment resisting distribution of tensile and compressive stresses on the member's cross-sectional area. This stress distribution is often defined as the bending stress. In addition to bending stresses, bending action may be also associated with shear stresses. These stresses develop in response to transverse loads applied along the length of the beam and "carried" by the member to its supports. The structural behavior of long straight beams is typical of Bending Action (Figure 6).

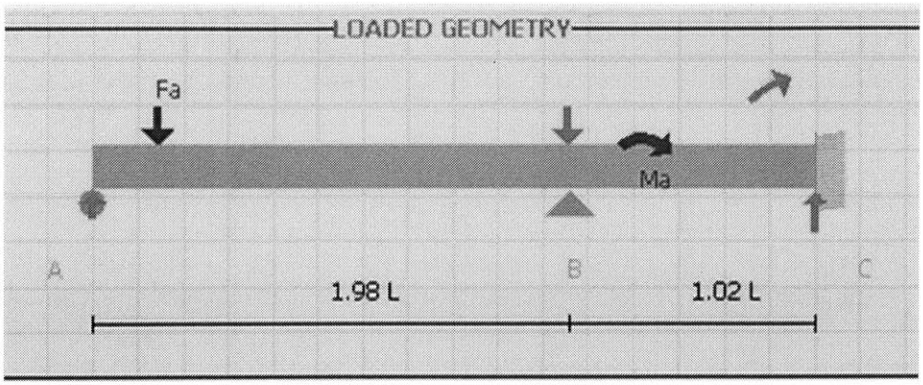


Figure 6 Beam loaded with a Concentrated Load and a Concentrated Moment

A beam member that is required to resist axial loads is often referred to as a beam-column. Although axially loaded beams are important, iBeam and PointLoad only consider loading configurations that include moments and transversely applied forces. Consequently, beam-column theory is not included in this analysis.

5.2.1 RESPONSE MEASURES

The response measures associated with Bending Action are the internal shear forces, the internal bending moments and the deformations (cross sectional rotation and deflection) experienced by the beam. Internal shear is the transverse force that the beam needs to develop in order to equilibrate the externally applied forces and force reactions. Likewise, the internal moment is the resistance to bending the beam has to provide in response to all externally applied loads. These measures are defined for all points along the member longitudinal axis. At any point, the shear force distribution and information about the member configuration allow the calculation of the shear stresses acting on the member's cross section. Similarly, the bending moment distribution combined with information about the beam cross section, leads to the calculation of the bending stress distribution.

In response to bending stresses, a structural member experiences transverse translations (deflections) and rotations of the cross section. Shear stresses also

produce deformations. On long beams subjected to substantial bending, however, these later deformations are often small enough to be neglected.

The response patterns associated with Bending Action are shear diagrams, moment diagrams, the deformed shape profiles and free body diagrams. A shear diagram is a scaled chart of the shear forces acting along a structural member. This diagram is often drawn on the profile of the member's longitudinal axes. Likewise, the moment diagram is a chart of the bending moment distribution. Deformation diagrams depict the geometrical changes experienced by the beam in response to the application of the loads. Deformation measures tend to be relatively small in comparison with the physical dimensions of the beam. As a result, in order to adequately illustrate these changes, deformation diagrams often need to be super scaled. A free body diagram is the graphic representation of the externally applied loads and support reaction forces acting on a structural system. These forces are normally sketched superimposed on an abstract geometric representation of the structure (Figure 7).

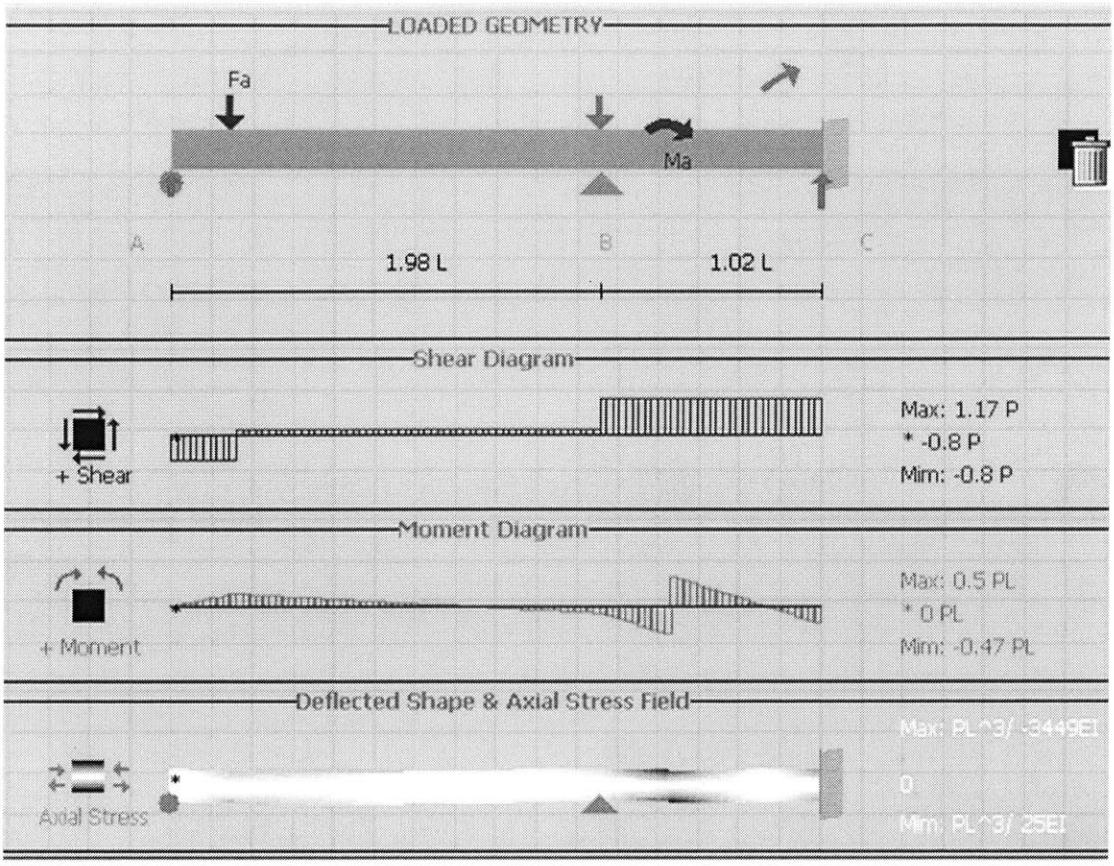


Figure 7 Loaded Beam and Corresponding Structural Response Patterns

5.2.2 GOVERNING PARAMETERS

The nature and configuration of the loading has a significant impact on the response of a structural system. Several types of loads can act on a beam, but they can be divided into two categories: concentrated loads and distributed loads. Concentrated loads pertain to either single forces or couples (moments) applied at discrete points along the beam. These loads are characterized by their magnitude and point of application. Distributed loads refer to forces or couples spread along the longitudinal axis of a beam. These loads are measured by their intensity and are expressed in load units per unit distance; and are characterized by their distribution profile along the beam.

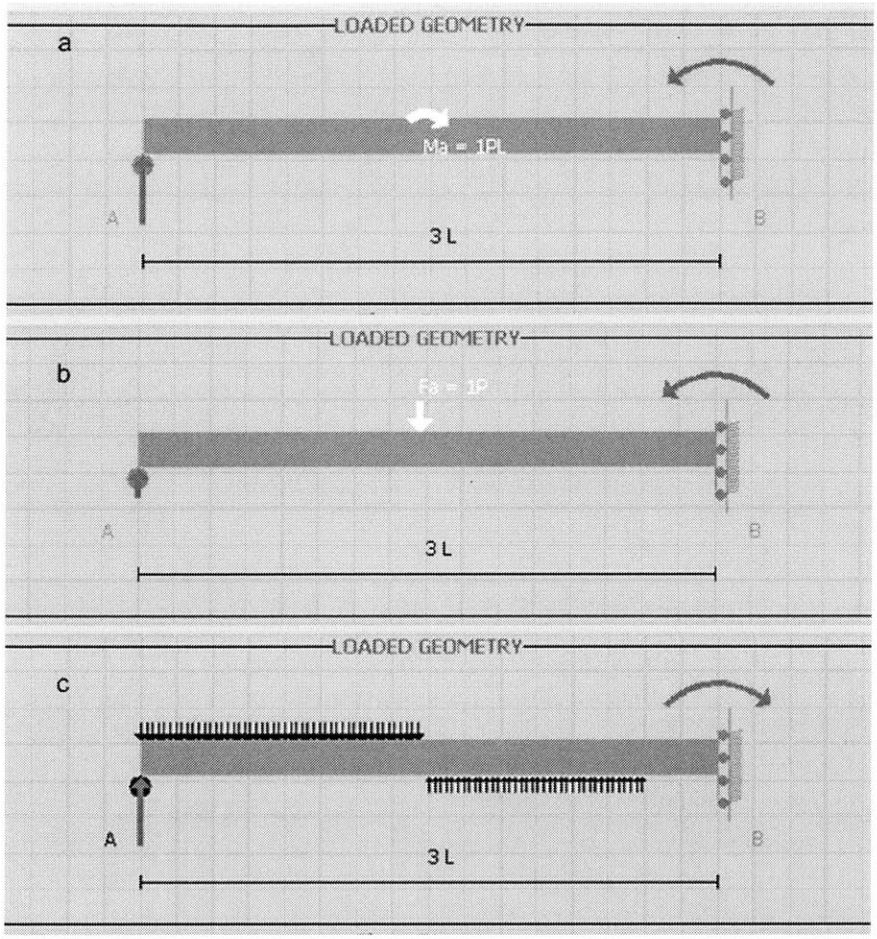


Figure 8 Beam Loaded with: a. A Concentrated Moment, b. A Concentrated Load, c. Distributed Loads

In addition to the loading configuration, the response of a beam depends on its geometrical configuration and boundary conditions. The geometrical configuration of a beam is defined by its span length(s) and the cross sectional properties such as bending and shear stiffness. The span length is the unsupported distance between beam supports. In cases where a member is continuous across supports, the beam has more than one span. The bending stiffness is a measure of the beam's resistance to rotation when acted on by a bending moment. Similarly, the shear stiffness is the resistance of the beam to transverse deformations due to shear forces. These two measures are properties of the beam's material and cross section. In a homogenous prismatic beam, the bending and shear stiffness are constant throughout its entire

length. iBeam allows the user to change the cross sectional properties from one span to another. However, within a span, the beam member remains homogenous and prismatic (Figure 9).

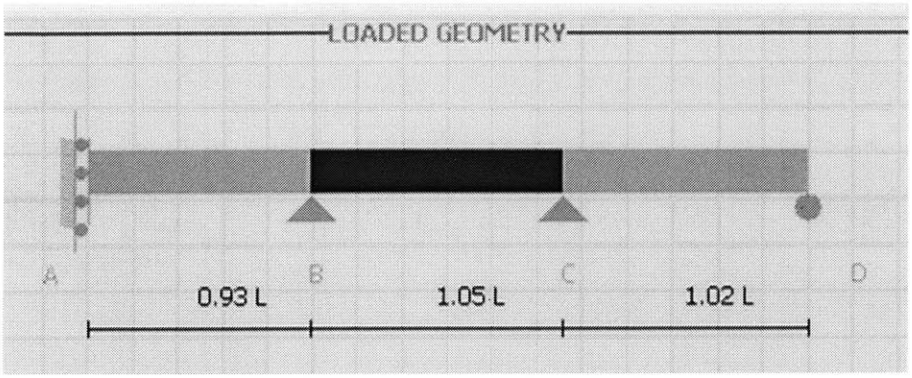


Figure 9 Three-Span Continuous Beam where the Center Span (darker) has Different Cross Sectional Properties

The boundary conditions have a significant effect on the response of a beam. Boundary conditions pertain to the deflections and slopes at the supports of a structural member (Gere, 2001). A support is an idealization of the physical means used to shore up a discrete point of a structure. In two dimensions, beam support types are idealized as being fixed, hinge, roller, moment, and free. These definitions describe the kind of reaction forces and displacement restrictions a support is capable of providing. For instance, a roller provides a supporting force and restricts travel in the direction perpendicular to the surface on which it rests. A roller allows the support point to rotate and does not provide a moment reaction. No reaction force and no displacement restrictions are provided by a roller support in the direction parallel to the resting surface. Table 6 depicts the graphical symbols used to describe two dimensional beam supports while Table 7 describes the idealized boundary conditions for these supports.

TABLE 6 GRAPHICAL REPRESENTATIONS OF TWO DIMENSIONAL BEAM SUPPORTS

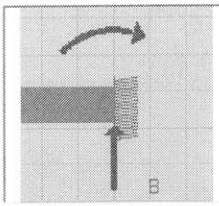
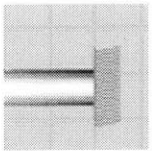
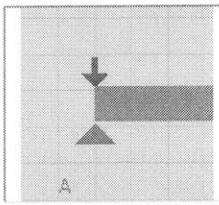
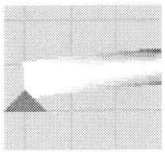
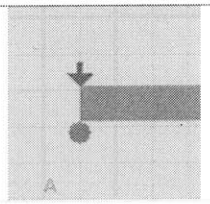
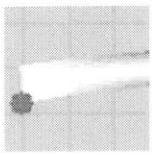
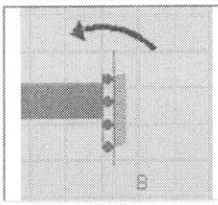
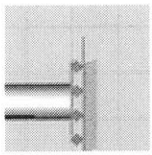
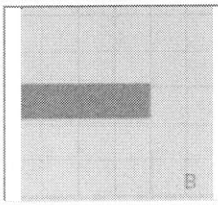
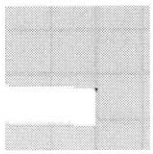
Support Type	Symbol	Structural Response
Fixed		
Hinge		
Roller		
Moment		
Free		

TABLE 7 BOUNDARY CONDITIONS FOR TWO DIMENSIONAL BEAM SUPPORTS.

	DIRECTION					
	Perpendicular to Supporting Surface		Parallel to Supporting Surface			
Support type	Force Reaction	Displacement	Force Reaction	Displacement	Moment Reaction	Rotation
Fixed	yes	no	yes	no	yes	no
Hinge	yes	no	yes	no	no	yes
Roller	yes	no	no	yes	no	yes
Moment	no	yes	yes	no	yes	no
Free	no	yes	no	yes	no	yes

5.2.3 CONSTITUTIVE EQUATIONS

The structural response of a prismatic straight beam is defined by the following set of uncoupled differential equations

$$EI\frac{d^4v}{dx^4} = -q \quad (1) \qquad EI\frac{d^3v}{dx^3} = V \quad (2) \qquad EI\frac{d^2v}{dx^2} = M \quad (3) \qquad \frac{d\ v}{dx} = \theta \quad (4)$$

Where:

EI = bending rigidity

q = load intensity equation

V = shear-force

M = bending-moment

θ = angle of rotation

v = deflection

x = the coordinate along the beam longitudinal axis

These equations assume that the material of the beam is linear elastic and that the deformations, angle of rotation and deflection, are small. In addition, they assume that deformations due to shear are negligible, so only deformations due to bending are considered; assumptions that apply correctly to most beams (Gere, 2001, pp 614).

To more clearly reveal the relations between the structural response parameters, these equations can be rewritten as follows

$$\frac{d\ v}{dx} = \theta \qquad (5)$$

$$EI\frac{d\theta}{dx} = M \qquad (6)$$

$$\frac{d\ M}{dx} = V \qquad (7)$$

$$\frac{d\ V}{dx} = -q \qquad (8)$$

It then becomes evident that the beam formulae constitute an uncoupled system of ordinary differential equations where the displacement is the final solution and is obtained by successive integrations of the load intensity equation *q(x)*.

The first integral delivers the shear-force equation, *V*. However, in most cases the definition of *q(x)* varies along the beam. Therefore different expressions need to be integrated between the points where these changes occur. In addition, concentrated forces and support force reactions acting at discrete points along the beam need to be integrated into the shear equation as Dirac-Delta terms. In a determinate system the value of these reactions can be calculated using equilibrium principles. In

indeterminate systems, the reactions can be incorporated in the integration with unknown values to be later determined using equilibrium and the known boundary conditions.

When the resulting equations are plotted, the result is a piece-wise continuous pattern that describes the shear force distribution acting along the entire beam. The discontinuities occur wherever a concentrated force or force reaction is applied. This pattern corresponds to the beam's shear diagram (Figure 10).

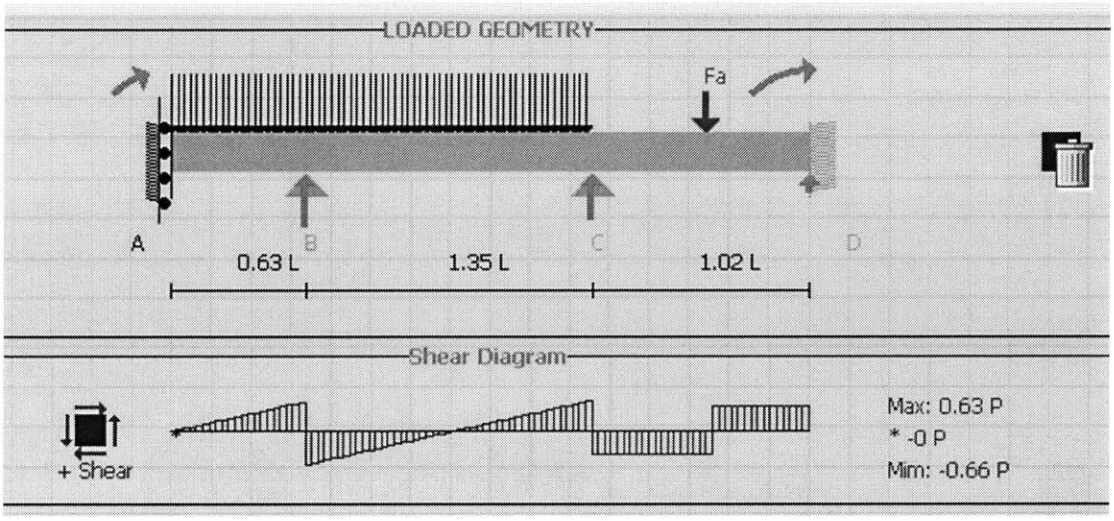


Figure 10 Loaded Beam and Corresponding Shear Diagram

The second integral delivers the bending-moment equation. Similarly to the previous step, obtaining this equation requires that different integral expressions be written for each of the intervals that define the shear force. In addition, it is also necessary to account for distributed moments and concentrated moments acting along the beam. These conditions introduce additional changes and discontinuities to the resulting bending moment distribution. The results of this second integration plotted along the beam profile, correspond to the moment diagram. Discontinuities in this diagram appear wherever a concentrated moment is applied.

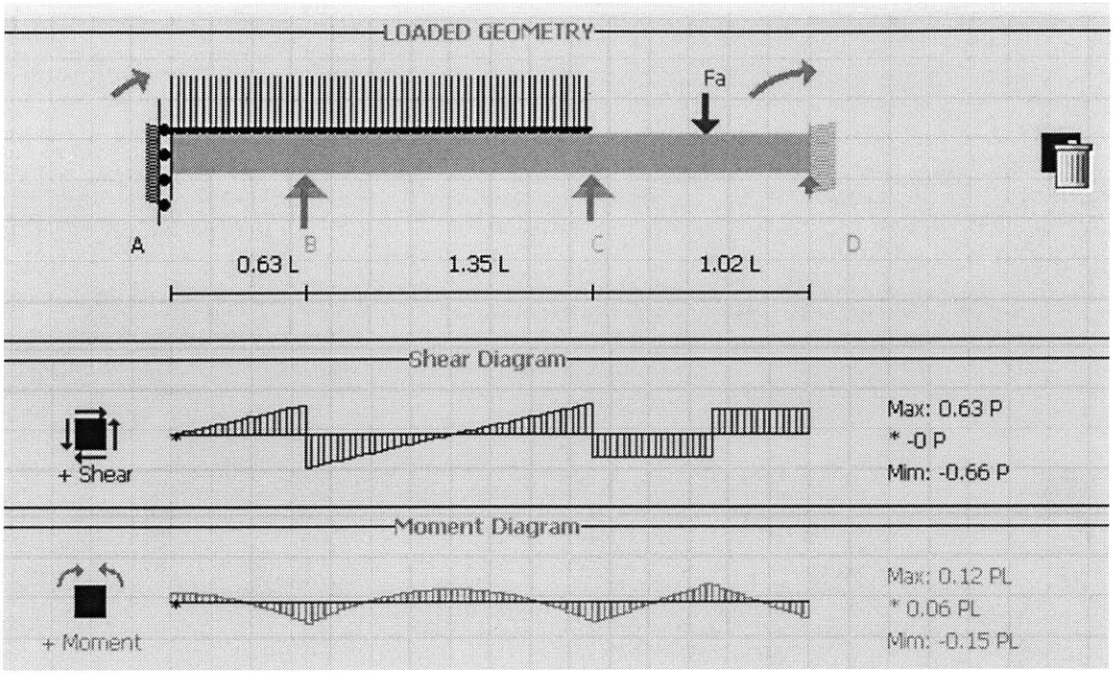


Figure 11 Loaded Beam and Corresponding Shear and Moment Diagrams

The third and fourth integrals yield equations that respectively define the slope times the bending rigidity and the deflection times the bending rigidity. The process needed to get these equations is as follows. Each interval of the bending moment equation is integrated once to obtain a set of slope equations v'. At each interval, the integration produces a new constant. Next, each of the slope equations is integrated once more to obtain a set of deflection equations. Once again, at each interval, a new constant of integration is introduced. Consequently there are two constants of integration for each beam interval. These constants are evaluated from known conditions pertaining to the slopes and deflections of the interval boundary points. These conditions fall into three categories: boundary conditions, continuity conditions and symmetry/anti-symmetry conditions. Boundary conditions relate to the deflections and slopes at the supports of a beam. Continuity conditions are connectivity constraints imposed on the deformation measures where the intervals of integration meet. For instance the slope and deformation of the point where two adjoining intervals connect needs to be the same if the beam is to remain intact. Two symmetry conditions can be specified

for beams. A symmetric beam under a symmetric load will experience a no rotations along its symmetrical axis. A symmetric structure under an anti-symmetric load will experience a no translations along its symmetrical axis (Figure 12) (Gere, 2001).

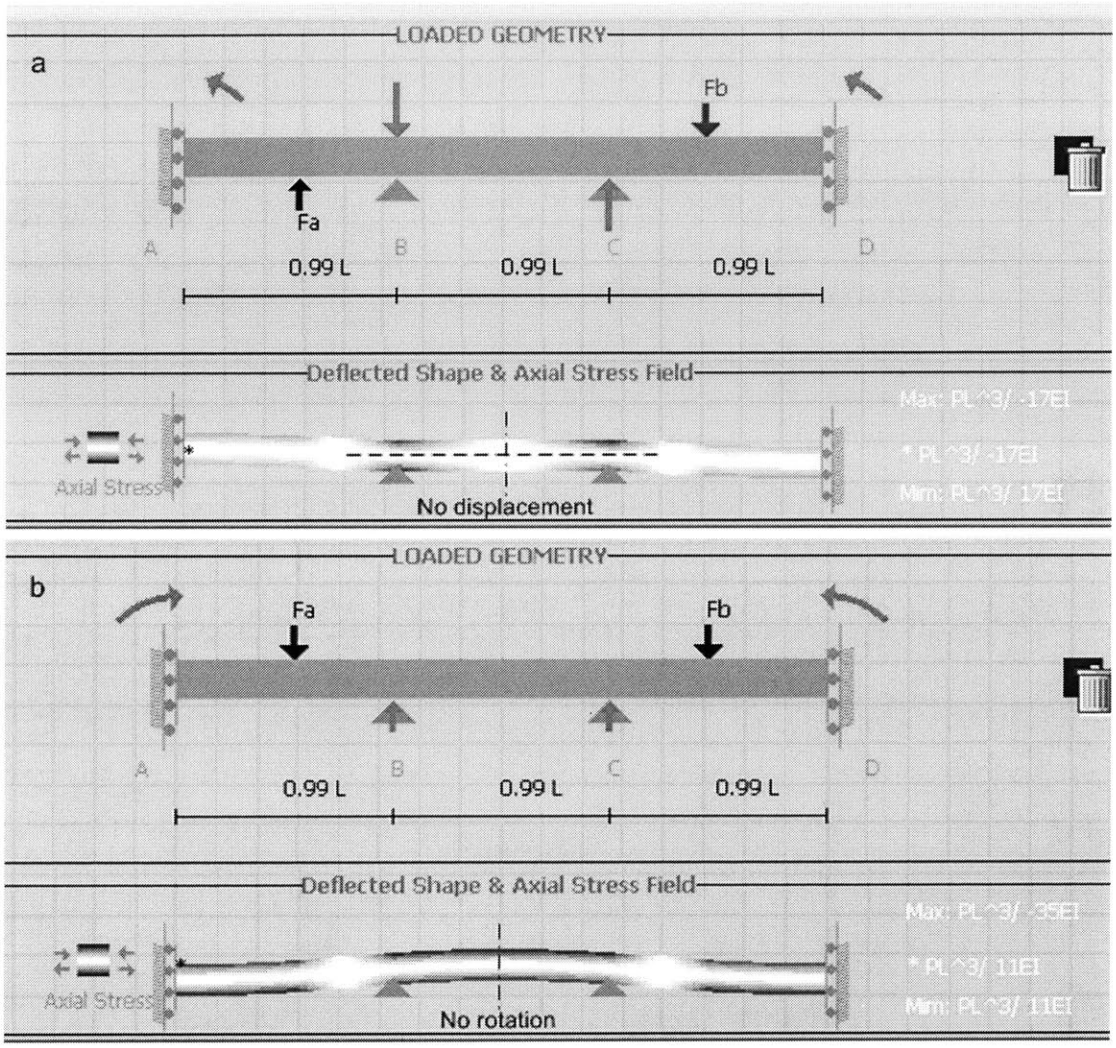


Figure 12 Top: Symmetric Beam Under Anti-Symmetric Loading configuration. Bottom: Symmetric Beam Under Symmetric Loading configuration

Once evaluated, the constants of integration are replaced back into the slope and deflection equations. The slope and deflection values plotted along the beam profile are known as the slope and deflection diagrams, respectively. Because of the continuity constraints, unless there is a moment release along the beam (internal

hinge), these diagrams can not experience abrupt changes, but rather have to be continuous along the entire beam.

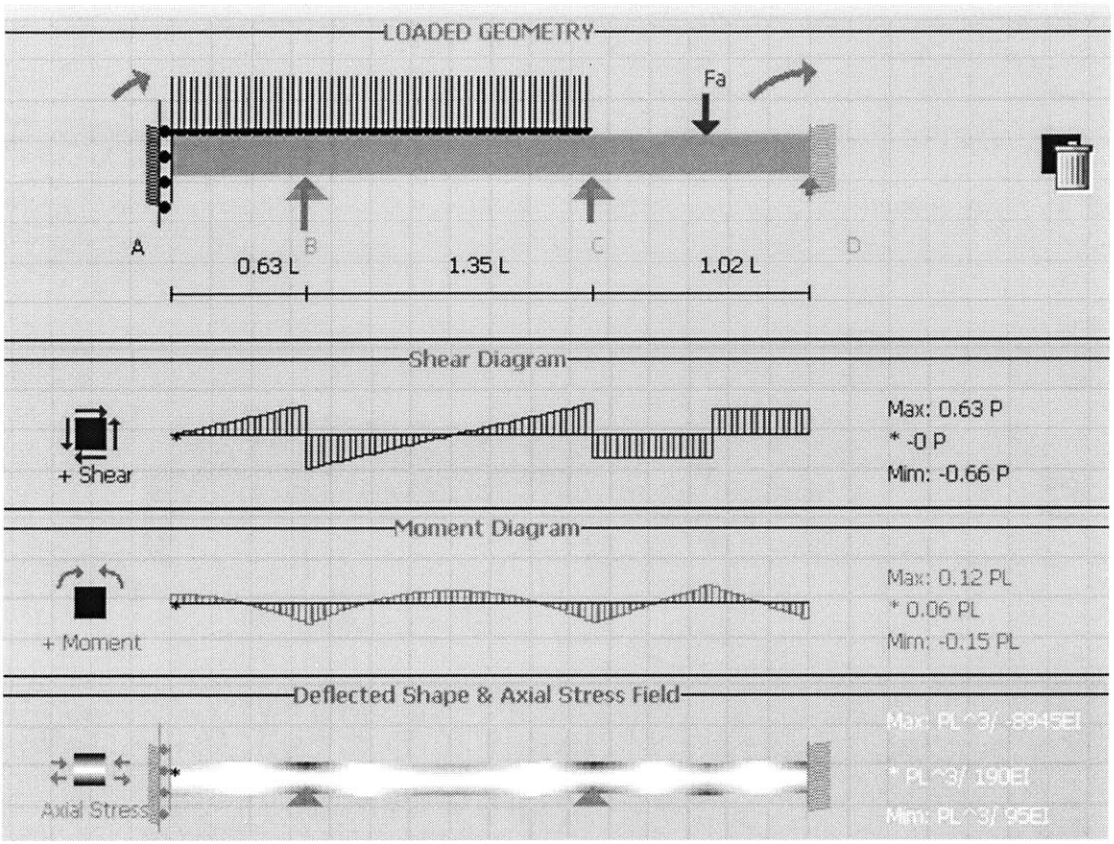


Figure 13 Loaded Beam and Corresponding Shear, Moment and Deformation Diagrams

When integrating, it is conventional practice to systematically incorporate the influence of the boundary conditions and loadings by moving along the beam's longitudinal axis from left to right. Therefore; the beam equations can be written as sums, where the terms correspond to the contribution that each these constraints make to the each of the structural response measures. This breakdown further clarifies nature of the solutions and can be used to facilitate the definition of qualitative principles relative to beam behavior.

Defining $x_{o,i}$ and $x_{f,i}$ the beginning and end points of a beam segment and defining q_i as a continuous load intensity function in $[x_{o,i}\ ,\ x_{f,i}]$, then:

$$\Delta V_i(x) = \int_{x_{o,i}}^{x'} q_i(x)dx \qquad (9)$$

$x \geq x_{o,i}$

$$x' = \begin{cases} x \rightarrow x \leq x_{f,i} \\ x_{f,i} \rightarrow x > x_{f,i} \end{cases}$$

see Figure 14.

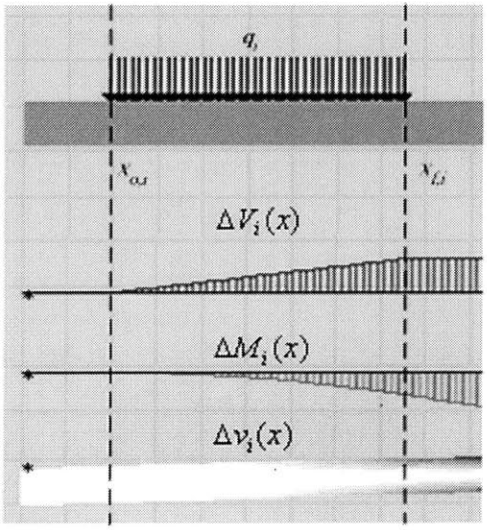


Figure 14 Beam Segment with Uniform Load

In the particular case of a concentrated force of magnitude F applied at $x_{o,i}$, this can be written as follows:

$$\Delta V_i(x) = F \qquad (10)$$

$x \geq x_{o,i}$

*zero otherwise,(*Figure 15*)*

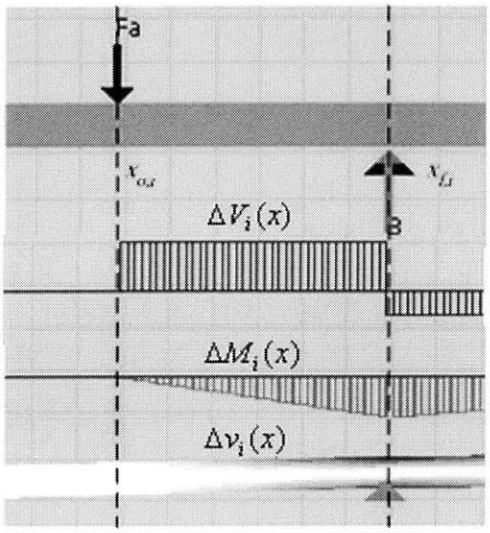


Figure 15 Beam Segment with Concentrated Load

Then,

$$\Delta M_i(x) = \int_{xo,i}^{x} \Delta V_i(x)dx \qquad (11)$$

$$EI\Delta\theta_i(x) = \int_{xo,i}^{x} \Delta M_i(x)dx \qquad (12)$$

$$EI\Delta v_i(x) = \int_{xo,i}^{x} \Delta\theta_i(x)dx \qquad (13)$$

$x \geq x_{o,i}$

$$x' = \begin{cases} x \rightarrow x \leq x_{f,i} \\ x_{f,i} \rightarrow x > x_{f,i} \end{cases}$$

Where, $\Delta V_i(x), \Delta M_i(x)$, $\Delta\theta_i(x)$, $\Delta v_i(x)$ respectively correspond to the effect in the shear force, bending moment, rotation and displacement due to q_i.

Similarly for a moment intensity equation m_i defined along a beam interval between in $[x_{o,i}\ ,\ x_{f,i}]$

$$\Delta V_i(x) = 0$$

$$\Delta M_i(x) = \int_{x_{o,i}}^{x'} m_i(x)dx \qquad (14)$$

$x \geq x_{o,i}$

$$x' = \begin{cases} x \rightarrow x \leq x_{f,i} \\ x_{f,i} \rightarrow x > x_{f,i} \end{cases}$$

In the particular case of a concentrated moment of magnitude Ma applied at $x_{o,i}$, this can be written as follows:

$$\Delta M_i(x) = Ma \qquad (15)$$

$x \geq x_{o,i}$

*zero otherwise, (*Figure 16*)*

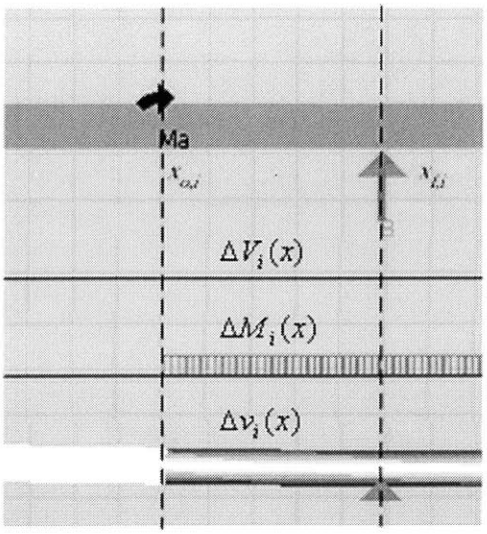


Figure 16 Beam Segment with Concentrated Moment

Then,

$$EI\Delta\theta_i(x) = \int_{xo,i}^{x} \Delta M_i(x)dx \qquad (16)$$

$$EI\Delta v_i(x) = \int_{xo,i}^{x} \Delta\theta_i(x)dx \qquad (17)$$

$x \geq x_{o,i}$

Where $\Delta V_i(x), \Delta M_i(x), \Delta\theta_i(x), \Delta v_i(x)$ respectively correspond to the effect in the shear force, bending moment, rotation and displacement due to m_i.

In addition, because of linearity:

$$V(x) = \sum_{k=1}^{k=i} \Delta V_k(x) \qquad (18)$$

$$M(x) = \sum_{k=1}^{k=i} \Delta M_k(x) \qquad (19)$$

$$\theta(x) = \sum_{k=1}^{k=i} \Delta \theta_k(x) \qquad (20)$$

$$v(x) = \sum_{k=1}^{k=i} \Delta v_k(x) \qquad (21)$$

Where $V(x), M(x), \theta(x), v(x)$ correspond, respectively, to the shear force, bending moment, rotation and displacement at $x \geq x_{o,i}$

5.3 CONCEPTUAL MODEL

Although the integration process described above would systematically deliver a precise account of the structural response of any loaded beam, it can be quite laborious to perform; hence it is practical only in small problems. However, it is possible to develop an understanding for the nature of these mathematical interactions and with it, deduce the qualitative functional relations and the heuristics that are implicit in the different structural response patterns. These functional relationships constitute the fundamental qualitative hypotheses of the Conceptual Model. Consequently, this is the knowledge that a needs to be symbolically represented and manipulated in order to develop effective assessment and feedback strategies within a virtual environment that tutors the conceptual understanding of structural behavior.

5.3.1 INTERVAL AS THE UNIT OF KNOWLEDGE

The mathematical model indicates that for analysis, a prismatic beam member needs to be subdivided into intervals of integration according to the loading configuration and to the geometry of the supports. This mathematical model also indicates that as

long as the continuity conditions between these segments are satisfied, the structural response of a beam can be described by the collective account of the behavior of each of its segments. Consequently, a beam can be viewed as a collection of interconnected beam intervals each responding to a different set of boundary and loading conditions. As a result, a set of definitions capable of fully describing the behavior of any such interval (generalized interval) would capture the qualitative principles that govern the structural behavior beams.

This section describes the functional relations relevant to the structural response of a generalized beam interval and organizes them into a hierarchy according to their generality and precision. This description is consistent with the hierarchy in a Conceptual Model. The highest hierarchical level corresponds to the general qualitative principles that define an interval. The description of the interactions that characterize the nature of the response patterns along an interval is next in the hierarchy. These interactions first define the relation between the input and the response patterns, and then between the response patterns themselves. Finally at the lowest hierarchical level, the continuity relations are implemented as heuristic principles. . These principles will then become the basis for the development of the symbolic knowledge representation strategies that support PontLoad and iBeam.

INTERVAL

An interval consists of a beam segment limited at each extreme by one or more ***Boundary Constraints***. These constraints are characteristics such as structural supports and loadings that introduce changes to any of the beam equations. These changes can be abrupt or gradual. Consequently, Boundary Constraints may be categorized as ***Abrupt Boundary Constraints*** or ***Gradual Boundary Constraints***.

ABRUPT BOUNDARY CONSTRAINTS

Abrupt Boundary Constraints pertain to conditions that introduce an impulse in either the load intensity or moment intensity equations. Mathematically these constraints are handled as Dirac-Delta loading functions and as such they introduce a sudden change in the value of the shear diagram or the moment diagram at the point where the load or the moment impulse is applied, thus defining the beginning of a new interval.

In addition to externally applied concentrated forces and moments, support reactions are sources of abrupt boundary constraints. These reactions are normally associated with restrictions to rotation and/or deflection at the support point, so in addition, they provide information relevant to the deformation of the beam. In the case of rigid supports, the values of these deformations are prescribed but the associated reaction is not. In the case of spring supports, both the value of the displacement and its associated reaction are unknown. There exists, however, a relation of dependence between these two measures. With a linear spring, for instance, the force is equal to the displacement experienced by the spring times the spring rigidity.

Physical boundaries also define beam intervals. These boundaries include all the geometric changes that perturb the continuity of the structure. In most cases they are associated with support boundary conditions. As a result, they introduce abrupt changes in the nature of the response measures. In the case of a continuous prismatic straight beam, the beginning and end points of the structure and internal shear releases constitute the physical boundary constraints

GRADUAL BOUNDARY CONSTRAINTS

Gradual Boundary Constraints occur at points along the beam where smooth changes to the definition of the force intensity or moment intensity equations occur. In general, if the function that defines the change to the force or moment intensity

equations is continuous within an interval, constraints do not introduce discontinuities in any of the structural response patterns. The beginning and end of uniformly distributed loads are instances of smooth boundary constraints. Table 8 illustrates the load related boundary constraints considered in iBeam, while Table 9 illustrates the support related boundary constraints.

TABLE 8 LOAD RELATED INTERVAL BOUNDARY CONSTRAINTS

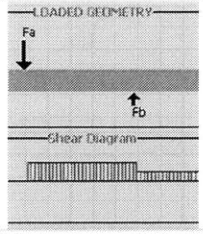
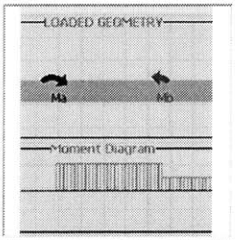
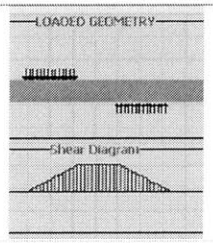
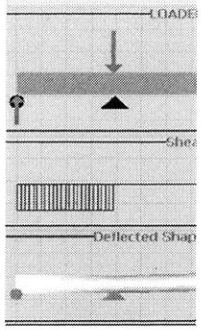
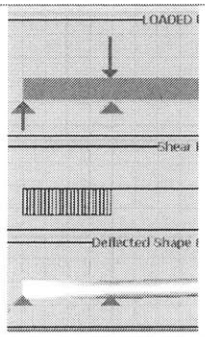
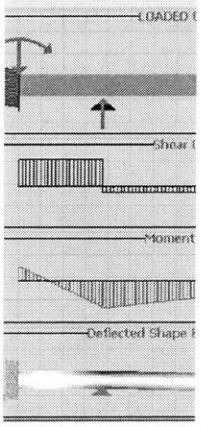
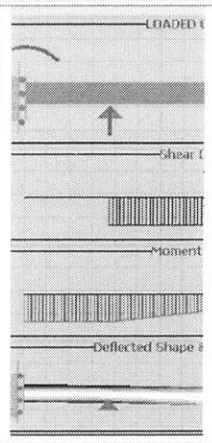
Source	Constraint Type	Constraint
	Abrupt change in shear	Externally applied concentrated force
	Abrupt change in moment	Externally applied concentrated moment
	Gradual change the shear	Beginning or end of externally applied distributed force.

TABLE 9 SUPPORT RELATED INTERVAL BOUNDARY CONSTRAINTS

Source	Constraint Type	Constraint
	Abrupt change in shear. No transverse displacement	Vertical reaction force from "Roller" type support.
	Abrupt change in shear. No transverse displacement	Vertical reaction force from "Hinge" type support
	Abrupt change in shear. No transverse displacement. Abrupt change in moment. No rotations	Vertical reaction force from "Fixed" type support. Resisting moment reaction from "Fixed" type support

	Abrupt change in moment. No rotations	Resisting moment reaction from "Moment" type support

5.3.2 QUALITATIVE RELATIONSHIPS OF A BEAM INTERVAL

According to the analytical study, loading and boundary conditions introduce changes in the nature of the beam equations. These changes emerge as distinct features that define the nature of the various response diagrams thus revealing the correlation between input parameters and structural reaction. In addition, the mathematical relations show that these changes ensue between the intervals of integration where these boundary conditions occur. Since these patterns embody the functional relations that define the structural behavior of a beam, understanding the nature of these changes is central to defining the character of beam intervals.

The process of sequential integrations organizes the response patterns into a hierarchy that starts with the shear diagram at the highest level, followed by the moment diagram, then the rotations and finally the vertical deflections. Therefore, the effect a Boundary Constraint has on a given response pattern propagates to the patterns up the hierarchy. For instance, a concentrated force acting on the left end boundary of an interval is mathematically defined as a Dirac Delta term, then according to Equation 10, at the beginning of the interval, the shear force should experience an abrupt change in its value equal to the magnitude of the force. If no other constraints affect the left boundary, the value of the shear force remains constant until the end of the

interval at the right end boundary. This effect would become evident as a jump in the shear diagram at the point where the load is applied followed by a constant value along the interval. Equation 11, then, indicates that the bending moment should experience a change in slope equal to the magnitude of the concentrated force. Two more steps of integration according to equations 12 and 13 reveal the rotation and the deflection will be third and forth order polynomials respectively. Following a similar reasoning process, Table 10 through Table 14 develops a set of relationships that describes the changes introduced by the different boundary constraints on the response patterns of a beam interval. Understanding these relations is necessary to qualitatively correlate loading conditions and structural make up with their corresponding structural response profiles.

TABLE 10 EFFECT OF LEFT BOUNDARY CONSTRAINT ON INTERVAL RESPONSE PATTERNS: CONCENTRATED FORCE

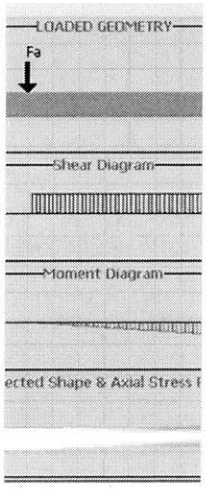
	Description: Force *Fa* applied at a point along the beam
	Shear Diagram: Sudden increment in the value equal to *Fa*
	Moment Diagram: Value increases linearly with a Slope equal to *Fa*
	Deflected Shape: Longitudinal stress intensity is proportional to the value of the moment.

TABLE 11 EFFECT OF LEFT BOUNDARY CONSTRAINT ON INTERVAL RESPONSE PATTERNS: UNIFORMLY DISTRIBUTED FORCE

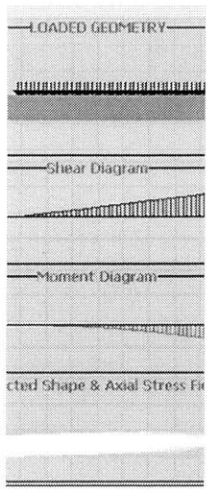
	Description: force of magnitude *w* applied along a beam interval at unit length intervals
	Shear Diagram: Value increases linearly with a Slope of *w*
	Moment Diagram: Value increases proportionally to *w* squared
	Deflected Shape: Longitudinal stress intensity is proportional to the value of the moment

TABLE 12 EFFECT OF LEFT BOUNDARY CONSTRAINT ON INTERVAL RESPONSE PATTERNS: CONCENTRATED MOMENT

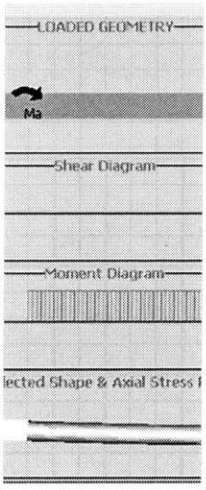
	Description: Moment of magnitude *Ma* applied at a point along the beam
	Shear Diagram: no effect
	Moment Diagram: Sudden increment in the value equal to *Ma*
	Deflected Shape: Constant curvature, constant longitudinal stress intensity.

TABLE 13 EFFECT OF LEFT BOUNDARY CONSTRAINT ON INTERVAL RESPONSE PATTERNS: RIGID SUPPORT FORCE REACTION

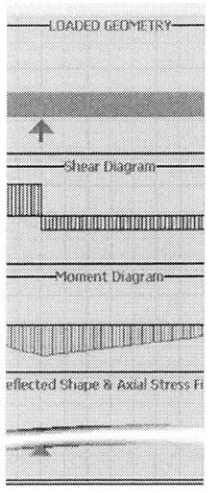
	Description: Rigid support capable of developing a transverse force reaction occurring at a point along the beam.
	Shear Diagram: Sudden increment in the value equal to the force reaction magnitude
	Moment Diagram: Value increases linearly with a Slope equal to reaction magnitude
	Deflected Shape: No vertical displacement at support point

TABLE 14 EFFECT OF LEFT BOUNDARY CONSTRAINT ON INTERVAL RESPONSE PATTERNS: RIGID SUPPORT MOMENT REACTION

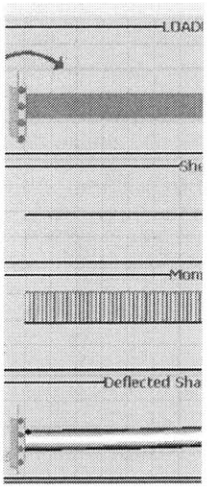
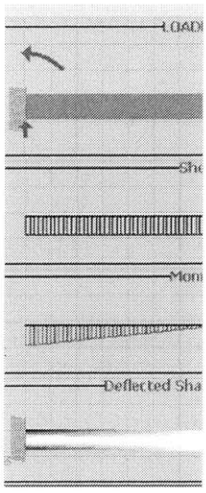
	Description: Rigid end support capable of developing a moment reaction.
	Shear Diagram: no effect
	Moment Diagram: Sudden increment the value equal to moment reaction
	Deflected Shape: No rotation at support point. Constant curvature

TABLE 15 EFFECT OF LEFT BOUNDARY CONSTRAINT ON INTERVAL RESPONSE PATTERNS: FIXED SUPPORT (COMBINES RIGID SUPPORT FORCE AND MOMENT REACTIONS)

	Description: Rigid end support capable of developing a force and a moment reaction.
	Shear Diagram: Sudden increment in the value equal to the force reaction magnitude
	Moment Diagram: Sudden increment the value equal to moment reaction
	Deflected Shape: No rotation at support point. No displacement at support point.

In cases where the left end of an interval is defined by a combination of one or more Boundary Constraints, equations 18 thru 21 indicate that the character of the response is equal to the linear combination of the responses generated by each of these constraints. In order to facilitate the representation of an interval, whenever one or more Boundary Constraints act at a given point along the beam, they are combined to form an Interval Boundary Condition. These Boundary Conditions incorporate the individual effect of each if its constraints into an entity that completely defines the behavior of its corresponding interval limiting point. A fixed support, for instance, is represented by an Interval Boundary Constraint that combines a reaction-force and a moment reaction. In addition, a fixed support provides a restriction to displacements and rotations (Table 15). Another example occurs when a concentrated force and a concentrated moment coincide at the same point along the beam, these boundary constraints would form a single Interval Boundary Condition characterized by these loadings but without restriction to either displacements and/or rotations.

5.3.3 RESPONSE PATTERN CORRELATIONS

The structural response measure equations are related to each other by one or more differentiation steps. This mathematical interdependence implies that specific correlations exist among the response patterns and that the characteristics of one may be inferred from the characteristics of another. For instance, the moment equation is proportional to the second derivative of the deflection. Consequently, if the moment diagram is positive along an interval, the curvature of the deflected shape would have to be positive. Table 16 illustrates the most informative pattern correlations that apply to the domain of beam bending action.

5.3.4 HEURISTIC SET

In addition to relations that can be inferred with a high degree of certainty, such as those shown on Table 10 through Table 16, there exist heuristic observations that while not always applicable, when combined with other information may help establish the nature of the structural response of a beam. For instance, while not always true, the curvature of the deformation in the vicinity of a concentrated load is likely to be of the opposite sign as the load. This hypothesis can be verified by looking at the moment diagram. According Table 16, a positive moment value is associated with a positive curvature and vice versa. Table 17 lists the heuristic relationships considered in PointLoad and iBeam.

TABLE 16 RESPONSE PATTERN CORRELATIONS

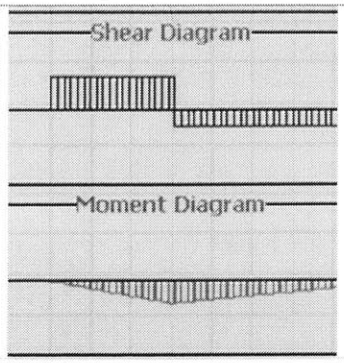
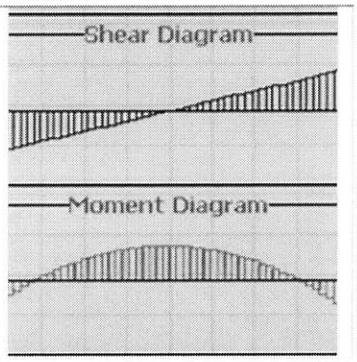
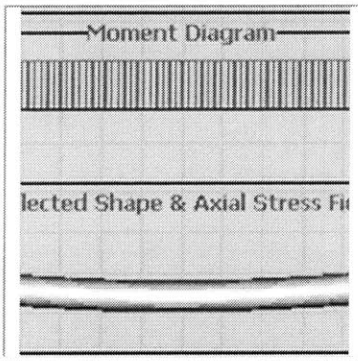
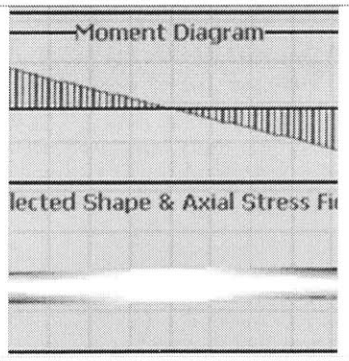
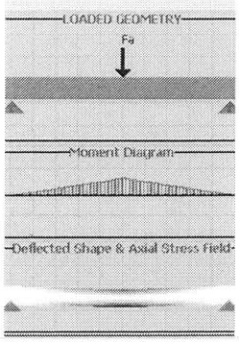
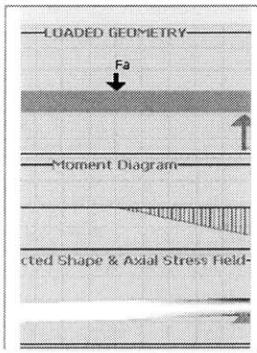
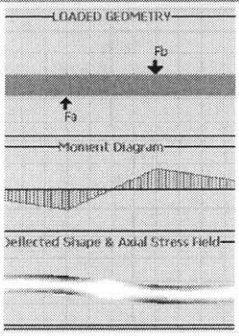
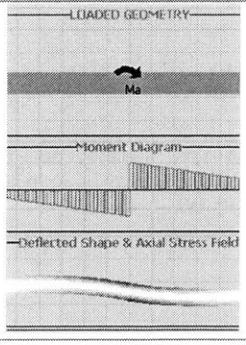
	Observation	Inference	Cause
	Shear diagram is Positive/Negative	Moment decreases/increases along Interval	Shear is the derivative of the Moment.
	Shear diagram changes signs	Moment reaches a local maximum or minima. Axial stresses reach maximum absolute values	Shear is the derivative of the Moment. Bending stress values are proportional to the Moment
	Moment diagram is Positive/Negative	Deflected shape curvature is Positive/Negative	Moment is the second derivative of the deflection.
	Moment diagram changes sign	Deflected shape has an inflection at point of sign change	Moment is the second derivative of the deflection.

TABLE 17 HEURISTIC SET

	Observation	Possible Inference	Confirmation
	Positive/Negative concentrated force on interior span	Negative/Positive curvature around point where load is applied	Check against sign of Moment Diagram
	Positive/Negative concentrated force on free ended span	Positive/ Negative curvature on free end	Check against sign of Moment Diagram
	Positive/Negative concentrated force	Upward/Downward deflection at and around point where load is applied	None
	Positive/Negative concentrated moment	Sign of curvature Changes where moment is applied	None

5.4 HIERARCHY OF THE CONCEPTUAL MODEL

The qualitative relationships described in Table 8 trough Table 17 form a set of relations that can be organized into a hierarchy that goes form the principles that define the boundaries of an interval to heuristic rules that help define the structural response of the beam according to the geometric and loading configurations. Table 18 shows this hierarchy and relates it to the content of this chapter.

TABLE 18 HIEARCHICAL ORDERING OF THE QUALITATIVE PRINCIPLES THAT DEFINE THE BEHAVIOR OF BEAM INTERVAL

Level	Principles	Tables
I	Relationships that facilitate the identification of Interval boundaries	Table 6, Table 7, Table 8, Table 9
II	Relation that define the nature of the response within an Interval and at its boundaries	Table 10, Table 11, Table 12, Table 13, Table 14, Table 15
III	Relations that help define the response of the beam between interval boundaries	Table 16, Table 17

CHAPTER 6. THE TECHNICAL APPROACH

The following sections describe the schematic architecture of the principal components that constitute iBeam. Although PointLoad is a less sophisticated environment, it uses the same basic architecture and has similar components to those used in iBeam. As a result, most of the descriptions in this chapter are also applicable to PointLoad. Significant differences between these two environments are addressed in Chapters 7 and 8 where the functionality of PointLoad and iBeam are respectively described.

As simulation based Experiential Learning Environments, PointLoad and iBeam have four primary functions: a. Model and simulate the structural response of the loaded beam. b. Facilitate the modification of these models and automatically illustrate the response of the modified beam. c. Introduce Epistemic Conflict through problem generation and presentation. d. Evaluate the user's response, assess the level of knowledge and provide post assessment feedback.

In order to fulfill these functions, PointLoad and iBeam integrate an interactive simulator and an intelligent virtual tutor into a cohesive software application. The programming strategy adopted in the development of these elements consisted of building specialized modules and reasoning agents that are accessed by a main control unit according to the needs of the four primary functions performed by these environments. This control unit handles the interaction with the user through a graphical user interface.

The interactive simulator employed by iBeam relies on five primary components: the Control Module, Problem Generation Module, Preprocessing Module, Finite Element Module and the Display Module. These components provide the Learning Environment with most of the capabilities associated with modeling, simulation and

illustration. These modules are also responsible for problem generation and problem presentation. In order to achieve effective assessment and feedback capabilities, iBeam employs an intelligent unit that consists of three elements, a Data Transfer Module, a Student Record and a Reasoning Module. Interaction between the environment and the user is facilitated by a Graphical User Interface (GUI). See Figure 17..

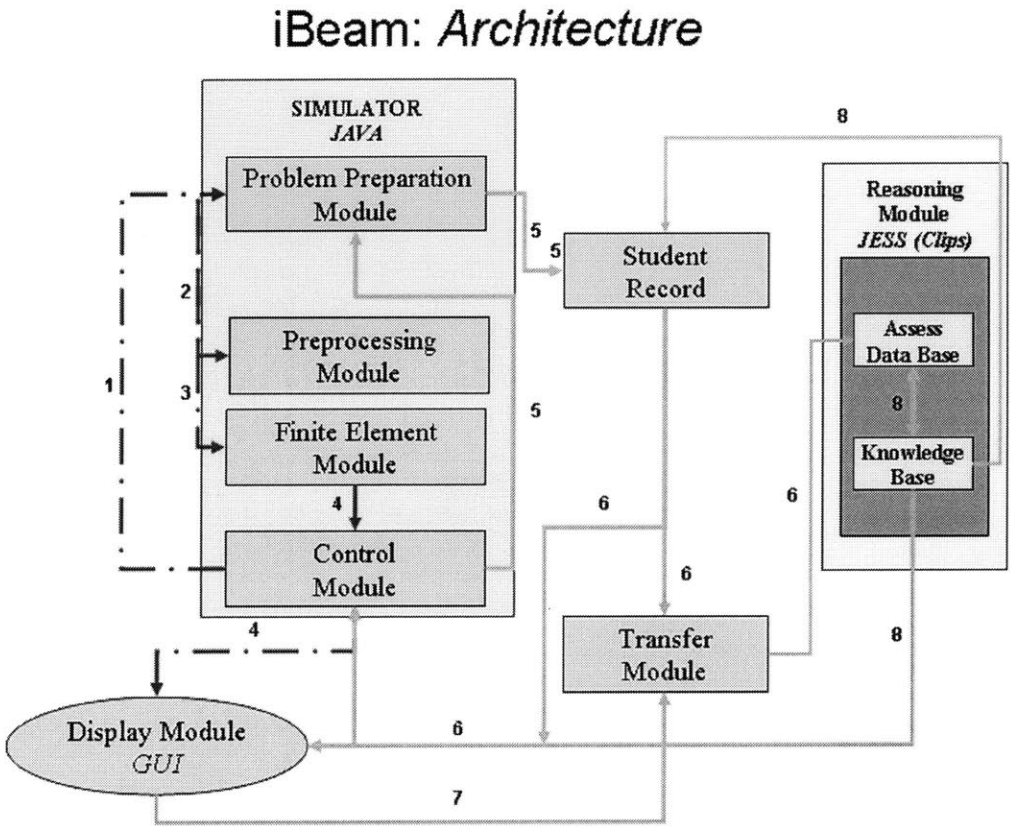


Figure 17 iBeam Basic Architecture and Functionality

6.1 COMPONENT FUNTIONALITY

In very general terms, these components work as follows:

1. When the Learning Environment is set off the Control Module calls the Problem Generation Module to generate a model of a beam with random structural and loading configurations.

2. This model is then passed to the Preprocessing Module whose main function is to insure that the model is properly configured and supported.

3. The Control Module then calls the Finite Element Module to get the structural response of the beam. This response is in numerical form. In the meantime, the Control Module prepares the elements of the GUI such as buttons and other control features.

4. Once the structural response has been calculated, the Control Module calls the Display Module. This module creates four graphic's panels that are embedded as part of the GUI. The top panel depicts the structural and loading configuration of the beam and allows changes to be made to the model through drag-and-drop control features. The other three panels present the structural response of the beam in the form of shear, moment and stress-deformation diagrams. In addition to these panels, the GUI has an explanatory text area where textual information relevant to the simulation and/or feedback is presented to the user.

At any time, the user can change the structural and/or loading configuration. When this occurs, the modified model is sent to the Preprocessing Module and steps 2 through 4 are repeated.

In addition to interacting with the simulator, the user can go into assessment mode by selecting the problem based exercise(s) implemented by both PointLoad and iBeam.

5. When the application goes into assessment mode, the Control Module generates an exercise problem whose level of complexity is determined by the information contained in the Student Record.

6. Then graphical and narrative information relevant to the problem is presented to the user through the GUI Display Module panels and the explanatory text area. This information is also sent to the Reasoning Agent through the Data Transfer Module.

7. When the student responds, the Control Module relays the answer to the Reasoning Engine which in turn evaluates the answer and provides feedback information that is processed and displayed either as text or the graphical output through the GUI. Also at this point, the reasoning agent modifies the Student Record accordingly. After solving a problem, the user can opt to solve another problem or to go back to simulation.

Both environments were developed as Applets. Most of the components were written using object oriented programming in Java. The reasoning modules were implemented in JESS using the JESS programming language. The following sections describe the make up and functionality of these components which for the most part consist of Java Classes or Java Class objects.

6.1.1 THE PROBLEM PREPARATION MODULE

In iBeam, the creation of the structural models for simulation or for problem presentation occurs through the instantiation of objects of the class `Input` and `Output`. `Input` objects contain attributes that define the structural and loading configurations of a continuous beam (Figure 18). `Output` objects initially contain empty categories where data relevant to structural response of its corresponding input structure can be stored Figure 24.

When an `Input` object is instantiated, the class constructors call native methods that automatically generate the necessary input parameters that define the structural and loading characteristics of a beam. The methods called and the generated parameters depend on the constructor used during instantiation. Two different constructors are available in the `Input` class. The first constructor receives only minimum information such as the name of the model, and creates a beam with a default

structural and loading configuration. This constructor is called whenever the environment is set off or when the user resets the structure during simulation or reenters the simulator after assessment. A second type of constructor is used when the environment goes into assessment mode. This constructor receives more specific information as to the make up of the structure so the attributes of the instantiated `Input` and its corresponding `Output` objects are established accordingly. This constructor is normally used during assessment when the nature of the problem is determined by information resident in the Student Record.

In general, the structural configuration of the beam is determined by the boundary conditions at both ends and the layout of intermediate hinge supports. To define these parameters, `Input` objects contain a `Vector` of objects of the class `support`, which represent structural supports. The structural configuration is also dependent on the number of spans, and the length and bending stiffness of each span member. The number of spans on a system is equal to the total number of supports minus one, while the distance between consecutive supports determines the length of the spans. All this information is contained in a `Vector` of objects of the class `member`. Objects of this class represent beam members (Figure 18).

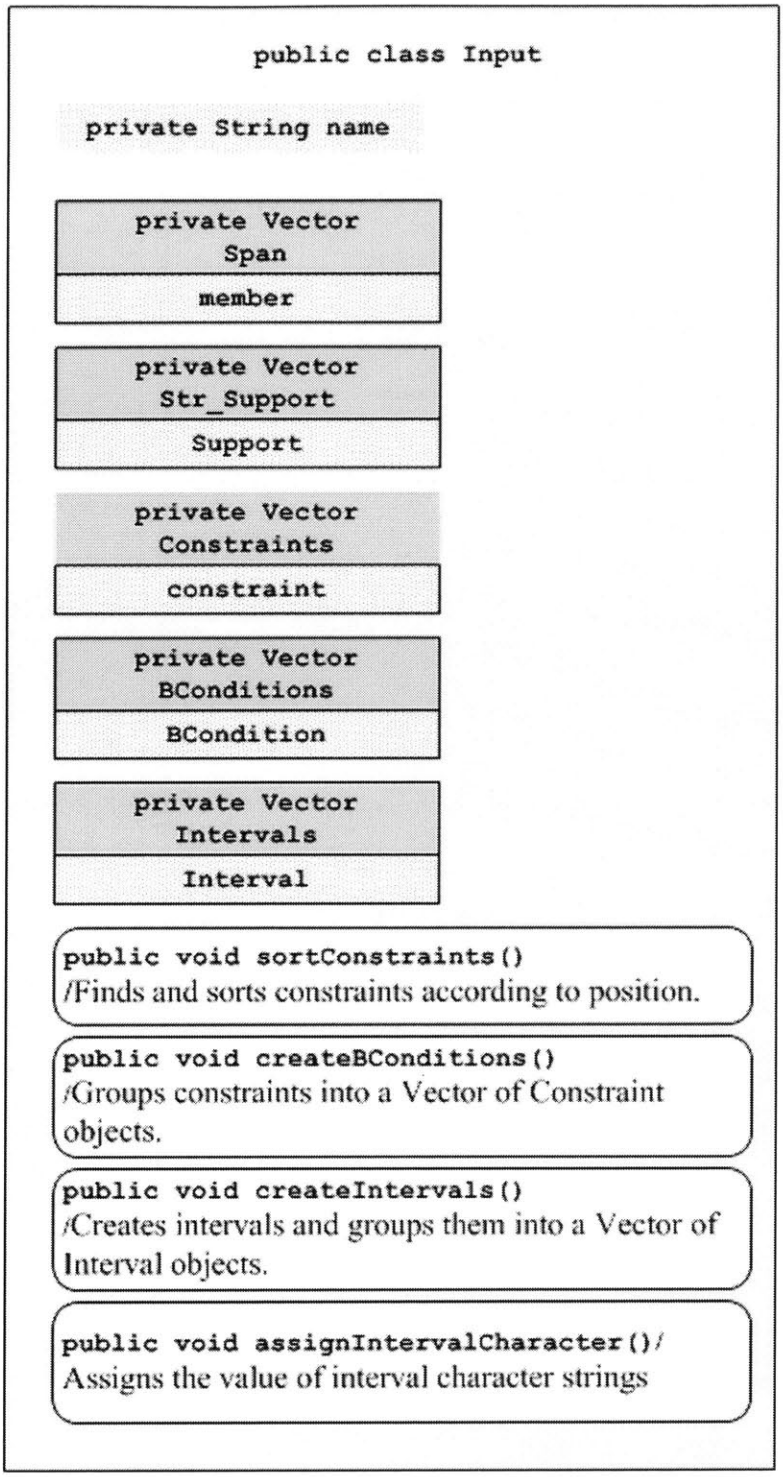


Figure 18 Main Properties of the `Input` Class

The loading configuration of the beam is determined by the type and the number of the applied loads. Multiple instances of concentrated forces, concentrated moments and distributed loads can be applied to the beam. This information is contained by each of the span `member` objects in three vectors: `PointLoad`, `DistLoad` and `ConcMoment`. `Pointload` and `DistLoad` contain objects of the class `force` and represent the concentrated loads and the distributed loads applied to their corresponding span member. Similarly, `ConcMoment` contains objects of the class `Moment` which represent the concentrated moments applied on its corresponding span member (Figure 19). Figure Figure 20 illustrates the contents of `Input` objects.

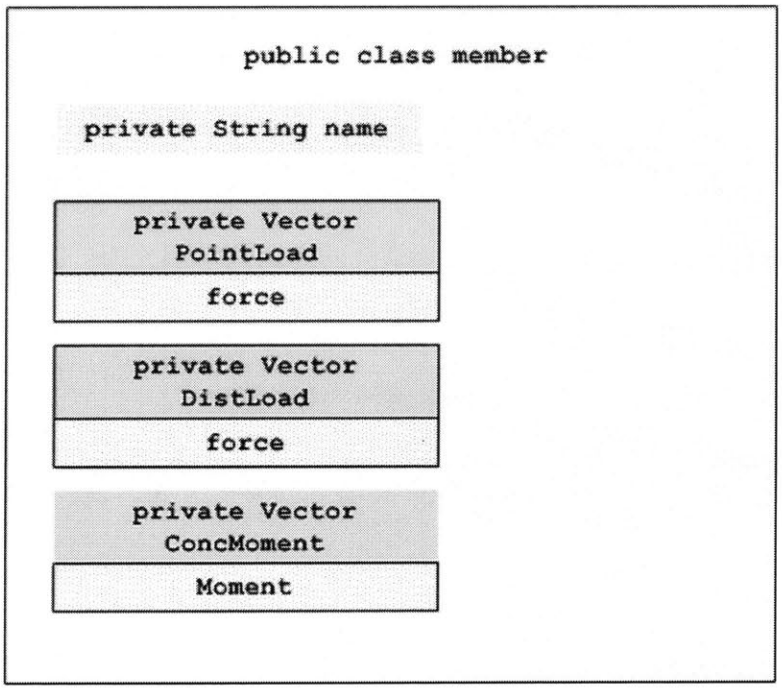


Figure 19 Main Properties of the `Member` Class

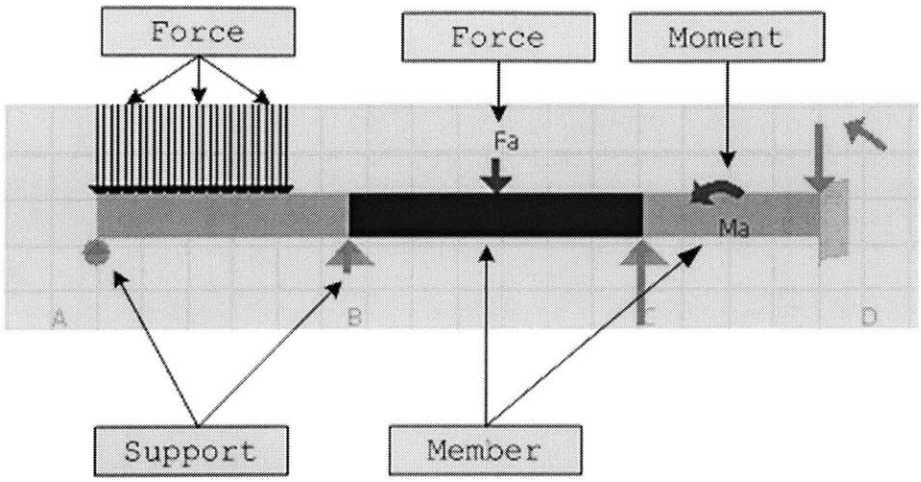


Figure 20 Loaded beam showing the Java Classes corresponding to the Objects that represent its structural and loading Parameters. These Objects are contained by Input.

For analysis, the beam model is subdivided into a discrete number of even spaces along its longitudinal axis. Each of these spaces is defined by two consecutive structural nodes. These nodes are represented by objects of the class `Node`. Each `Node` object can activate up to six `dof` objects. `dof` objects store information relevant to displacements, rotations, forces and moments along the global coordinates. In iBeam, each node activates only two `dof` objects; `dof_v` for vertical translations and forces and `dof_b` for moments and rotations about the out of plane axis (Figure 21).

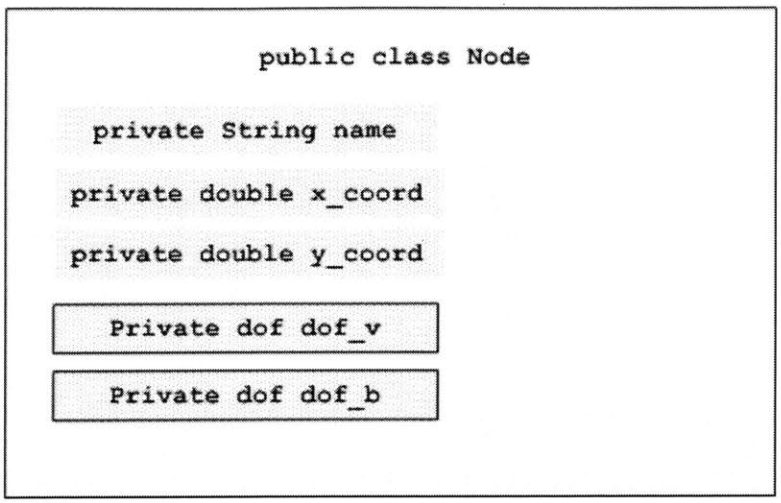


Figure 21 Main Properties of the `Node` Class

A flexural element is placed between each pair of consecutive nodes. The position and the length of each element are defined by the coordinates of the interval nodal points. These elements are represented by objects of the class `Element` and are characterized by their material and geometric properties such as length, moment of inertia and the material elastic modulus (Figure 22). These properties are used to generate the element stiffness matrix that is used by finite element formulation in order to generate the system stiffness matrix. Although each `Element` object can be assigned different structural characteristics, in iBeam, beam members are prismatic and homogeneous, so all the elements that correspond to a given span are assigned similar properties Figure 23.

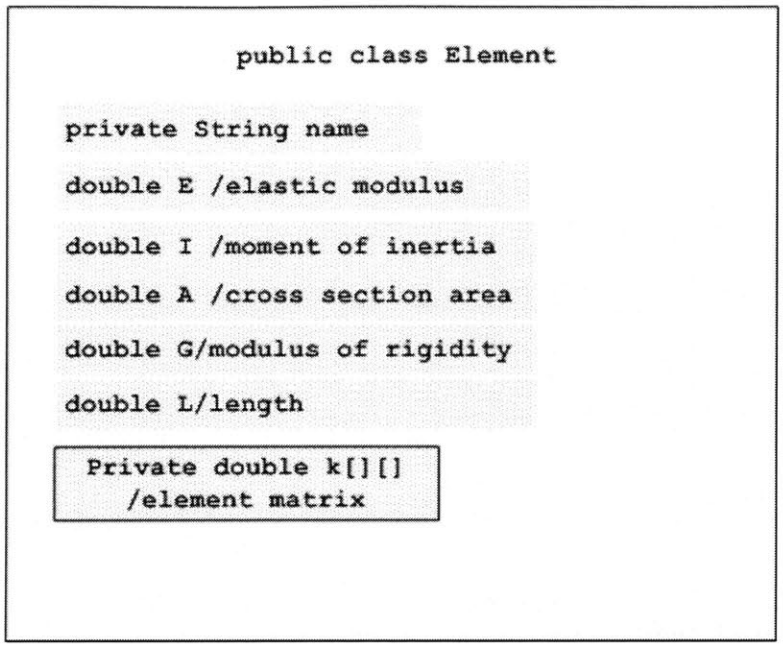


Figure 22 Main Properties of the `Element` Class

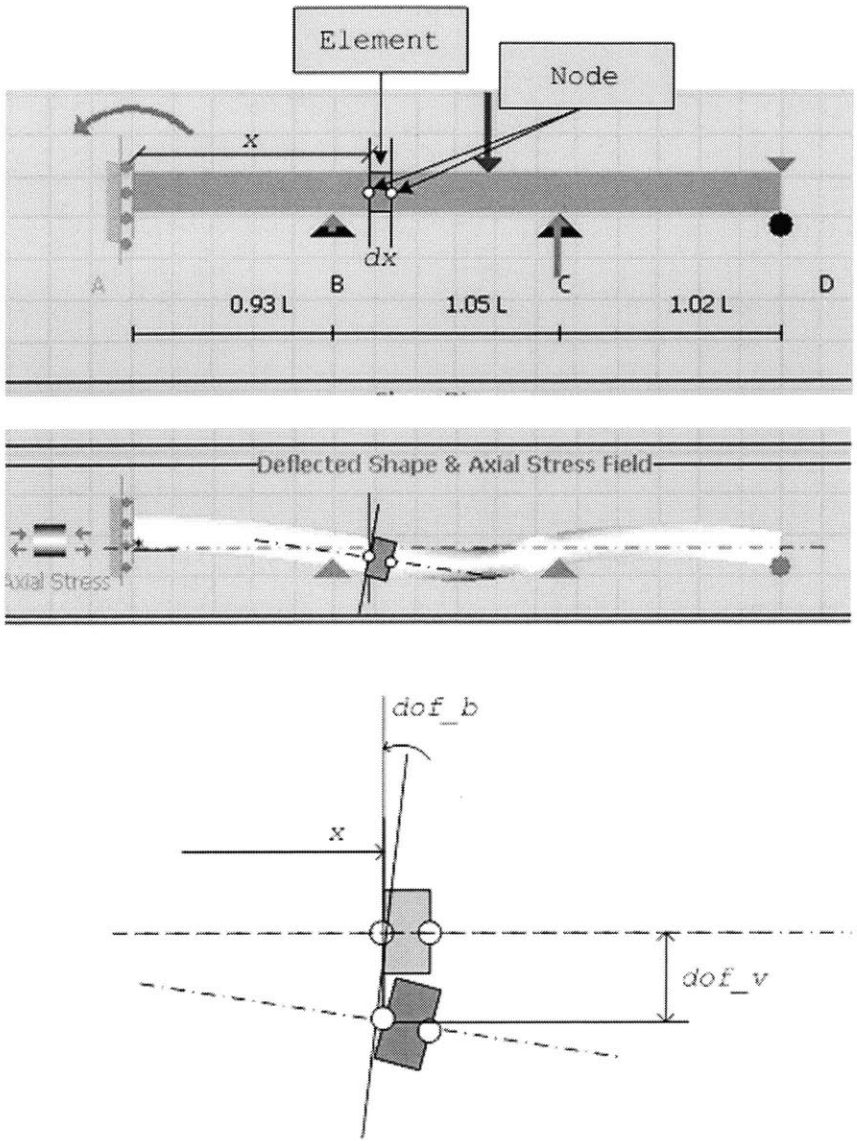


Figure 23 Top: Beam showing a Flexural Element and bounding structural Nodes. Middle: Deformed Beam showing rotated and displaced Flexural Element. Bottom: Detailed view of Flexural Element and Nodes and activated degrees of freedom.

The information relevant to the structural response of the beam is stored in objects of the class `Output` in two Java `Arrays`; `Str_Node` and `Str_Elem`. `Str_Node` is a sorted set of all the Node objects that represent the beam. The sorting is based on

the position of each node along the longitudinal axis. Each of these `Node` objects contains nodal response measures such as the nodal displacements, rotations, forces and moments. `Str_Elem` is a sorted set of all the `Element` objects that represent the beam. The sorting is also based on the position of each element along the beam longitudinal axis. These `Element` objects contain the response measures relevant to axial stresses along the beam Figure 24.

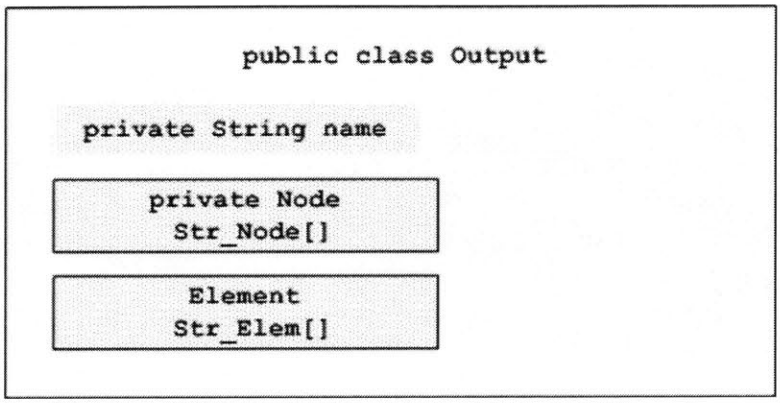


Figure 24 Main Properties of the `Output` Class

6.1.2 THE PREPROCESOR

During interaction with the Learning Environment, the user can modify the structural and loading configuration of the beam. To sort out these changes, the Preprocessing Module is called to adjust the input parameters of the modified virtual model and to check whether the beam has enough support points before the Control Module calls the Finite Element engine to recalculate the structure. In addition, the preprocessor re-labels items such as the forces and span lengths before displaying the modified structural and loading configurations, and the recalculated response diagrams on the screen.

The Preprocessing module consists of the class `Preprocess`, which contains the static method `adjustInput`. This method receives an `Input` object that has been

modified and rearranges the supports, modifies the span layout, redistributes the loads as it sees it necessary.

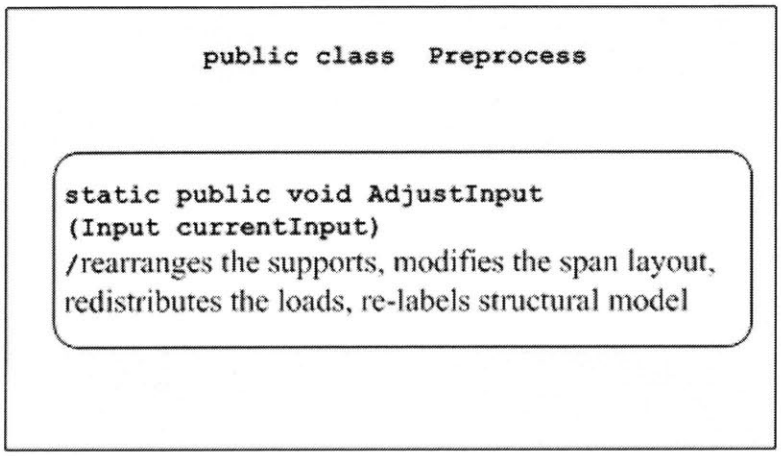


Figure 25 Main Properties of the `Preprocess` Class

Rearranging the supports consists of adjusting their ordering sequence according to their new position along the beam and if the distance between two supports is smaller than a minimum tolerance, one of the supports is eliminated. This action is aimed to prevent having a cluttered beam model which results in crowded display diagrams Although in most instances the right support gets eliminated, this changes if the right support is and end support.

Once the supports have been rearranged, the spans need to be redefined. This consists of assigning newly instantiated `member` objects to each of the beam intervals lying between the updated supports. Finally, `adjustInput` collects all the loads and allocates them to the appropriate span in function of their position along the beam longitudinal axis.

6.1.3 THE FINITE ELEMENT ENGINE

The Finite Element Engine performs all the calculations that determine the structural response of the beam. This task is performed by a finite element algorithm based on

flexural elements. The programming of this algorithm and supporting capabilities such as processing the input and output information is performed through deterministic processes in the Java programming language version 1.3.1. Matrix solving routines use the Jama Java class package developed by The MathWorks and the National Institute of Standards and Technology (NIST) and released to the public domain.

This algorithm is contained in the static methods of the class `Calculate`: `genNodes`, `genInitDisp`, `genElements`, `assembleMatrices`, `solve` and `getNodalForces`. All of these methods, except for solve and `getNodalForces`, receive an `Input` and an `Output` objects as arguments. `solve` and `getNodalForces` get the `Output` object only (Figure 26).

`genNodes` defines the finite element mesh, instantiates the corresponding `Node objects` and stores them in the `Str_Node` array held by Output. `genElements` generates all the necessary flexural elements according to this mesh and organizes them into the `Str_Elem` array of the `Output` object received as reference. `genInitDisp` assigns initial end actions, and end displacements according to the loading distribution, and according to the material and the mechanical properties of the elements in the `Str_Elem` array. `assembleMatrices` reorganizes the system matrix, performs static condensation and sets up the vector infrastructure before `solve` performs the matrix operations that result in the numerical prediction of the beam structural response. Finally, `getNodalForces` updates the attributes of the `dof` objects of each node of the `Str_Node` array.

In addition to performing the finite element calculations, `Calculate` contains the method `setPictLimits` which takes an `Input`, an `Output` and an object of the class `GraphicLimits`. This method uses information from the `Input` and `Output` objects to determine a set of sizing parameters that allow the Display

Module adjust the magnitude of the illustrations so that they are not under scaled and fit within the display panels. These parameters are stored in the `GraphicLimits` object.

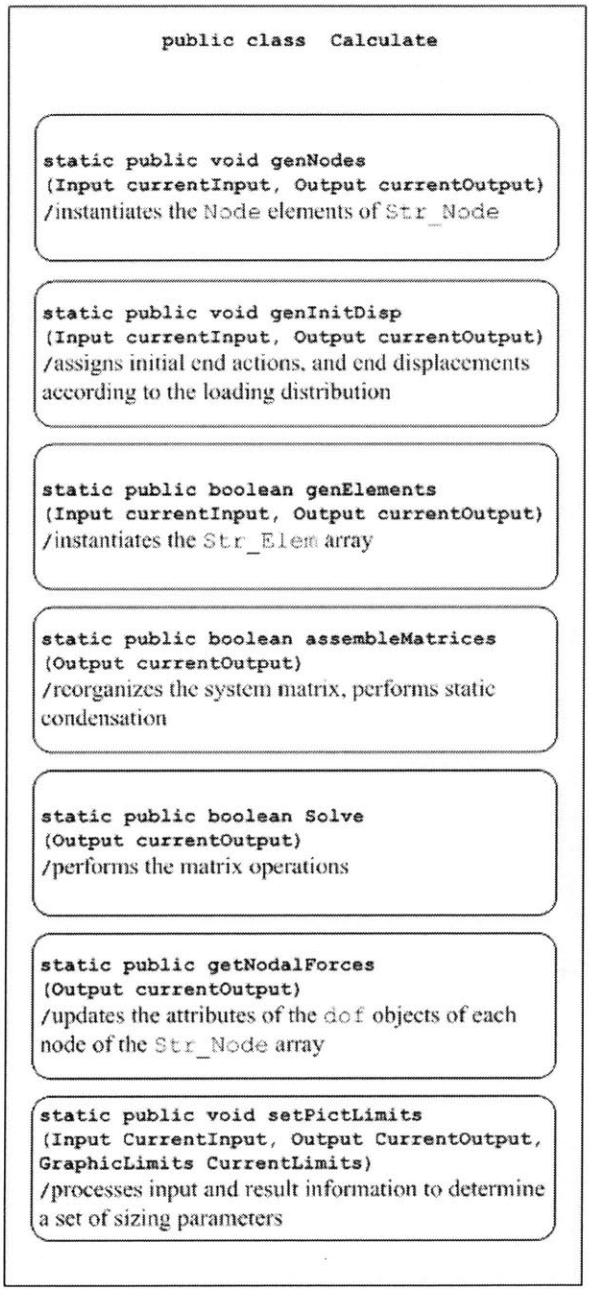


Figure 26 Main Properties of the Calculate Class

6.1.4 THE DISPLAY MODULE

Through the GUI, the Display Module provides the user with images of the structural and loading configuration of the beam model and with graphical representations of the beam's structural response. This response is illustrated through the recreation of the free body, shear, moment and stress-deformation diagrams. The Display Module, in addition, facilitates the modification of the beam configuration through interactive drag-and-drop control features.

The Display Module consists of objects of the classes `ControlPanel`, `BeamPanel` and `GraphicLimits`. iBeam depends on objects of these the first two classes to illustrate the beam configuration and its structural response, and to allow the user make changes to the beam's input parameters. Objects of the class `GraphicLimits` contain information so that `ControlPanel` and `BeamPanel` scale the display drawings so that reaction forces and the response diagrams fit within the display panels.

`ControlPanel` objects take information from `Input` and `Output` objects and process it into graphical illustrations of the structural and loading configuration of the beam, and of the support reactions. In addition, `ControlPanel` displays quantitative relational information of both the loading and the reaction forces and moments. `ControlPanel` extends the Swing Java class `JPanel`, which allows it to outline drawings on the computer screen (Figure 27). Two main methods are involved in providing the functionally of `ControlPanel`; `drawBeam`, and `paintComponent`.

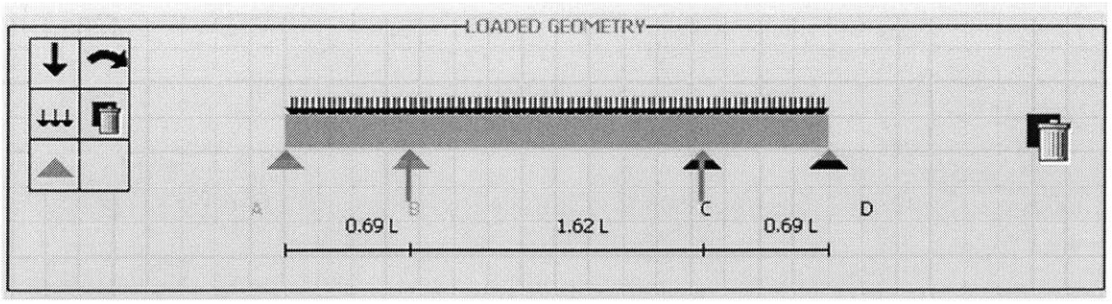


Figure 27 `ControlPanel` Display Pane

Along with other supporting information, `drawBeam` takes an `Input`, an `Output` and a `GraphicLimits` objects and prepares the data that will serve as the basis for the displays. `paintComponent` is the overridden method from the Java class `JPanel` that processes this information and draws the displays on the screen. `paintComponent` is not explicitly called. This method is internally activated every time the screen is presented for the first time, needs to be repainted or is "refreshed". `ControlPanel`, in addition, implements the Java language `MouseListener` and `MouseMotionListener` interfaces, which allow it to have interactive mouse activated click and drag-and-drop features. These features let the user add, delete or alter the position and the sense of the loads; add or delete supports, and change the span properties. These Drag and drop features were customized using the Graphics and Graphics2D Java packages Figure 28.

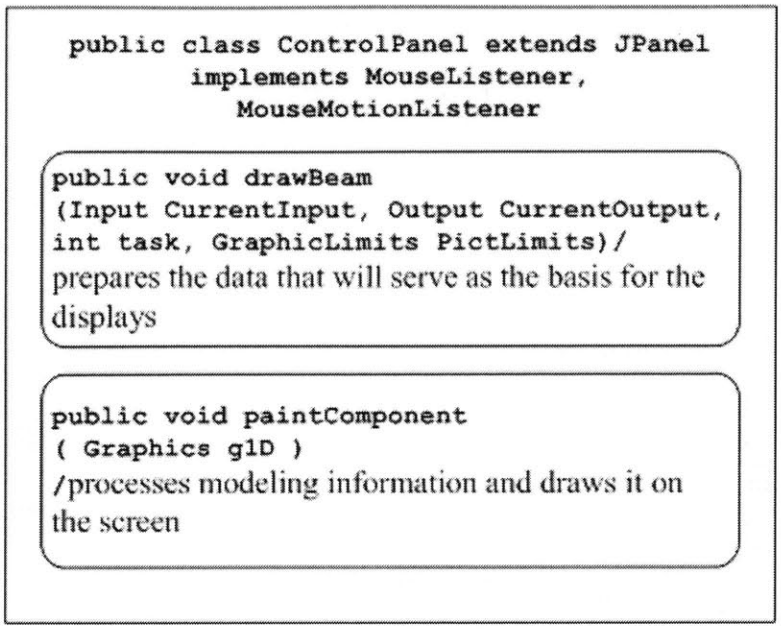


Figure 28 Main Methods of the `ControlPanel` Class

The results of the simulation are processed and displayed by objects of the class `BeamPanel`. This class extends the Swing Java class `JPanel` and implements the `MouseListener` and `MouseMotionListener` interfaces. Like `ControlPanel`, through `drawBeam`, `BeamPanel` takes an `Input`, an `Output` and a `GraphicLimits` objects and prepares the data that will serve as the basis for the displays. `paintComponent` is the overridden `Jpanel` method that processes this information and draws the displays on the screen. This method is internally called every time the screen is presented for the first time or needs to be refreshed. `BeamPanel` can illustrate the structural response of the beam in the form of shear, moment and stress-deformation diagrams. This last diagram type portrays the profile of the beam deflected shape along with a contoured illustration of the axial stresses acting on the beam (Figure 29).

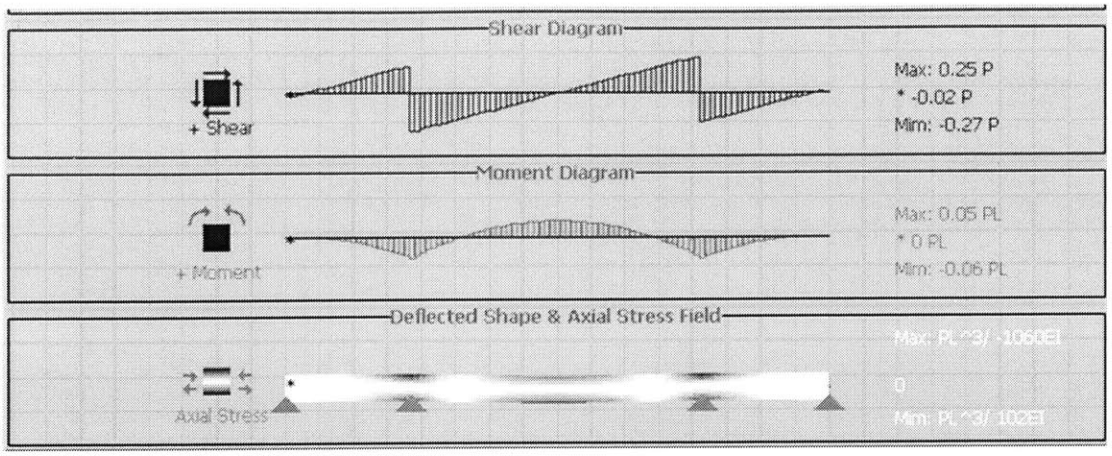


Figure 29 BeamPanel displays portraying a Shear, Moment and a Stress-Deformation Diagram

These interfaces facilitate the implementation of a mouse activated tool that shows the numerical value of any of the response measures by clicking on the desired coordinate of the corresponding diagram. This tool also, lets the user view the changes in the response by dragging the mouse cursor along the beam. (Figure 30).

```
public class BeamPanel extends JPanel
      implements MouseListener,
        MouseMotionListener
```

```
public void drawBeam
(Input CurrentInput, Output CurrentOutput,
int task, GraphicLimits PictLimits)/
```
prepares the data that will serve as the basis for the displays

```
public void paintComponent
( Graphics g1D )
```
/processes modeling information and draws it on the screen

Figure 30 Main Properties of the `BeamPanel` Class

6.1.5 THE CONTROL MODULE

The Control Module manages the interaction with the user, controls the flow of information between the various processing modules and organizes the screen output on the Graphical User Interface. This module consists, primarily, of an object of the class `iBeam`, which extends the `JApplet` Java class. When the learning environment is called, `iBeam` automatically generates a default structural model, generates the structural response and presents the beam layout and the structural response on the corresponding GUI display panels. Other items of the GUI such as control buttons and text areas are also instantiated and displayed on the screen. The user can then interact within the simulation environment or engage in assessment exercises.

SIMULATION MODE

In order to meet the requirements of the simulation environment, iBeam orchestrates the functions of the Problem Preparation Module, the Finite Element Engine and the Display Module. Whenever iBeam is called, the user goes into simulation mode or resets the learning environment; `iBeam` generates a new model by instantiating the `Input` object `MainInput`. The default-configuration constructor is called to create this model. At this point, `MainOutput` and `MainLimits` are also instantiated as `Output` and `GraphicLimits` objects respectively. `MainInput` and `MainOutput` are then passed to the `adjustInput` method of `Preprocess`. After `Preprocess` operates on the `Input` object and prepares the attributes of the `Output` object, `iBeam` calls methods of the static class `Calculate` perform the finite element calculations; `genNodes`, `genInitDisp`, `genElements`, `assembleMatrices` and `solve` are the methods called in this process. Then, the method `Calculate.setPictLimits` is called in order to define the display scaling factors according to the values of the attributes of `MainInput` and `MainOutput`. These factors are stored in `MainLimits`.

The illustration of the structural and loading configuration of the main model is performed by `BeamPict` an object of the class `ControlPanel` held by `ibeam`. When `BeamPict` is instantiated, the constructor of the `ControlPanel` class receives `iBeam` as a reference while the method `drawBeam` receives a reference to `MainInput`, `MainOutput` and `MainLimits`. The information contained in these objects is then processed and a representation of the loaded beam and the support reaction forces is displayed.

To display the calculation results three objects of the class `BeamPanel`; `ShearPict`, `MomentPict` and `DefPict`, are instantiated and held by `iBeam`. Similarly to `BeamPict`, these objects receive a reference to `iBeam` upon instantiation and each of their `drawBeam` methods receives a reference to `MainInput`, `MainOutput` and `MainLimits`. `ShearPict` displays the shear diagram, `MomentPict` displays the moment diagram and `DefPict` displays the deformed shape and the beam's axial stress field.

Whenever user alters the structural or the loading configuration of the beam, `iBeam` automatically changes the affected attributes of `MainInput`, calls the preprocessing methods, repeats the calculations and redraws the display panels automatically (Figure 31).

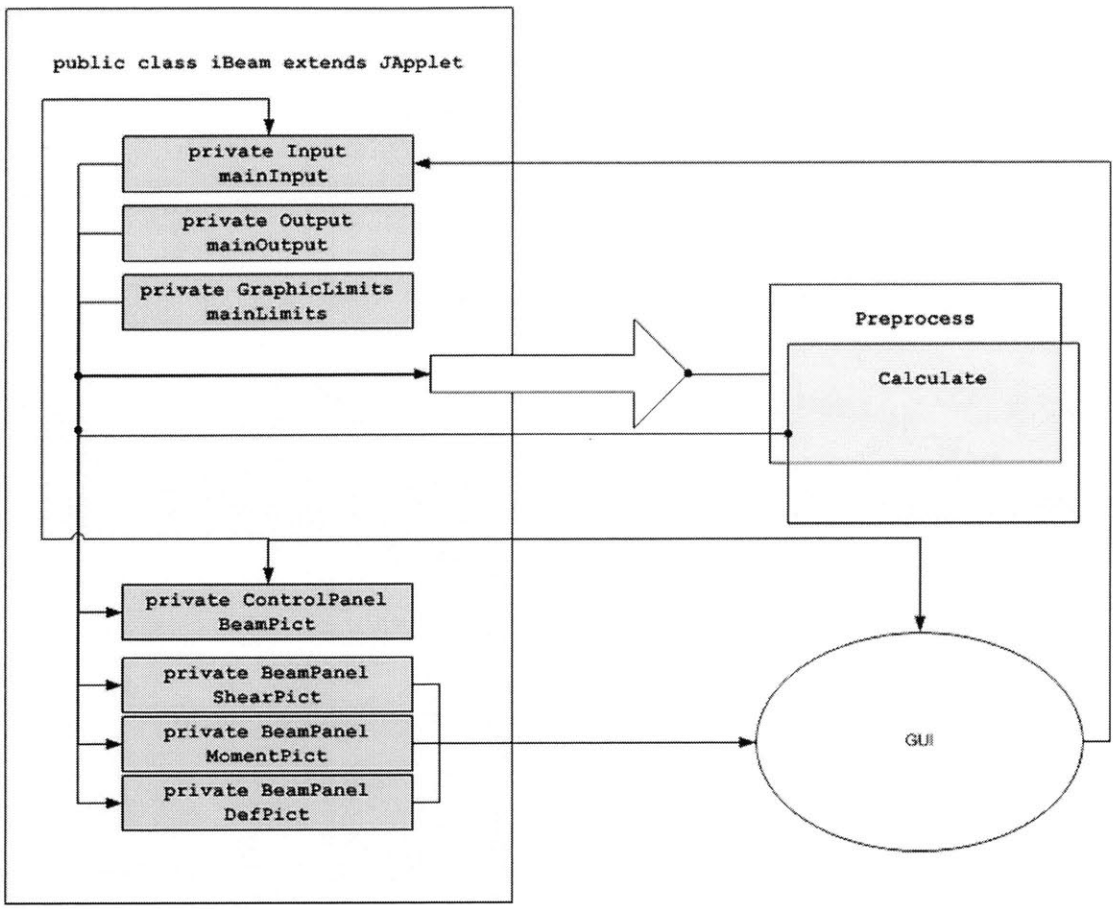


Figure 31 iBeam interactions in Simulation Mode

6.2 REASONING MODULES

When the user engages in assessment exercises, iBeam and PointLoad have to generate and present a problem, simulate the structural response of the beam models that represent this problem, assess the user's response, and provide feedback. For the most part, problem presentation and simulations require processing information in a predetermined sequence through algorithmic techniques that are effectively handled by deterministic programming. Assessment and feedback, on the other hand, are decision based and demand causal reasoning; situations that are better handled by inference mechanisms based on knowledge representation (Jackson, 1999). As a

result, in order to satisfy the functionally of the assessment exercises, these environments adopt a strategy that integrates Java object oriented programming with rule based reasoning schemes built in JESS, a Java compatible expert system shell.

In this approach, rules are written to capture the reasoning tasks associated with assessment and feedback. Each of these rules consists of a set of constraints and a set of actions. The constraints are matched against facts resident in a database. A rule is triggered if the facts resident in this data base satisfy its constraints. When this occurs, processes declared by the rule actions are enabled (fired). Since these inference mechanisms are used primarily as reasoning aids, these processes consist primarily of program instructions performed through calls to methods in the main Java program or changes to the attributes of Java objects.

A specialized search algorithm performs the matching of the rules against information contained in the data base. The infrastructure that facilitates this matching is often called a Search Engine while a related set of rules is commonly known as the Knowledge Base. JESS Version 5.2 is the Search Engine used by both PointLoad and iBeam. This version of JESS relies on an improved form of the Rete search algorithm. Rete searching operations are of order N*F. N is the number of facts in the database which is a function of control parameters. F is the number of rules; a function of the operations (Friedman-Hill, 2001). The programming of the rules is performed in the JESS programming language, a CLIPS-based declarative code. JESS Java classes facilitate the interface between the reasoning agents and the rest of the programming infrastructure.

In iBeam and in PointLoad, the fact data bases consist of information about the simulations and the student interaction with the assessment exercises. This information is passed as Java Bean objects. JESS allows the declaration of Bean Objects with dynamic attribute values. Whenever these types of values are modified, the fact data base is automatically updated so that if the search engine runs again, the

rules affected by the changes are revisited. Consequently, declaring relevant input attributes as dynamic, facilitates an interactive response between the simulating environment and the reasoning modules.

6.2.1 KNOWLEDGE REPRESENTATION

The functional relations described Table 6 through Table 17, are the fundamental conceptual knowledge entities that serve as the basis for determining the features of the simulation and the development of symbolic knowledge representation schemes that can be used for assessment and feedback. Although quite simple in themselves, in order to draw conclusions, these entities combine into complex interrelated networks and have to be manipulated among large data sets. In order to effectively use this knowledge and deal with this complexity, reasoning tasks are divided, categorized and each handled by a specialized reasoning agent. This approach reduces the number of rules in each agent's knowledge base and minimizes the number of items in each corresponding fact database; thus reducing the searching order and memory allocation requirements.

The knowledge that has to be represented in each agent can be divided into *core* knowledge and *process* knowledge. Core knowledge is a symbolic representation of the Conceptual Model. Consequently, it addresses fundamental relationships and does not depend on the task at hand. Therefore, it is embedded in all the intelligent agents and provides the constraint framework of the knowledge base. Process knowledge builds on core knowledge and is task specific as it represents paths that lead to a decision or to an action. As a result it shapes the processes declared by the rules of the agent.

KNOWLEDGE REPRESENTATION STRATEGY

Although some of the details may vary, PointLoad and iBeam use similar knowledge-representation strategies. In these applications, when the `genNodes()` method of

`Calculate` is called, it automatically initiates a process that creates a representation of the beam as a series of connected segments each describing a generalized interval (Figure 32). The algorithms involved in this process are contained by `Input` in the methods `sortConstraints()`, `createBConditions()`, `createIntervals()` and `assignIntervalCharacter()` (Figure 18).

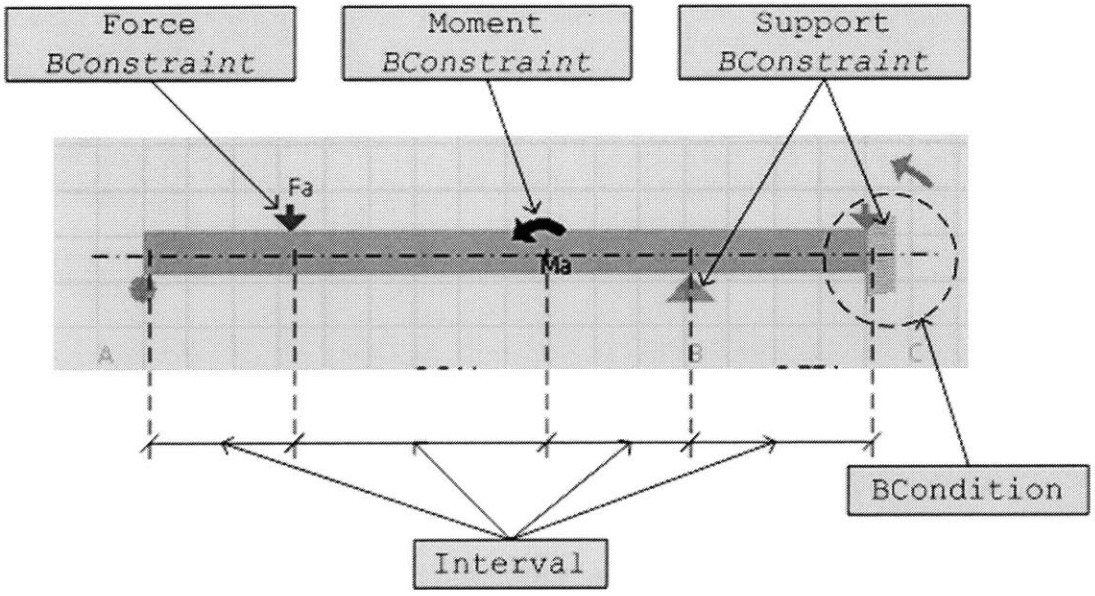


Figure 32 Representation of a Beam for Assessment and Feedback

An interval is a beam segment between two consecutive Interval Boundary Conditions and it is represented by a Java Bean object of the class `Interval`. An Interval Boundary Condition consists of one or more Boundary Constraints and is represented by objects of the class `BCondition`. `Force`, `Moment` and `Support`, whose instantiated objects represent concentrated loads, concentrated moments and structural supports respectively, constitute the Boundary Constraints considered in iBeam. Objects of these classes implement the `constraint` interface. This interface defines abstract Java Bean type methods that are independently defined within the implementing classes. Through this interfacing mechanism, all constraint objects can be controlled with the same calling methods,

while allowing each class to define its own response. Figure 33 shows the methods of the `constraint` interface that are associated with knowledge representation.

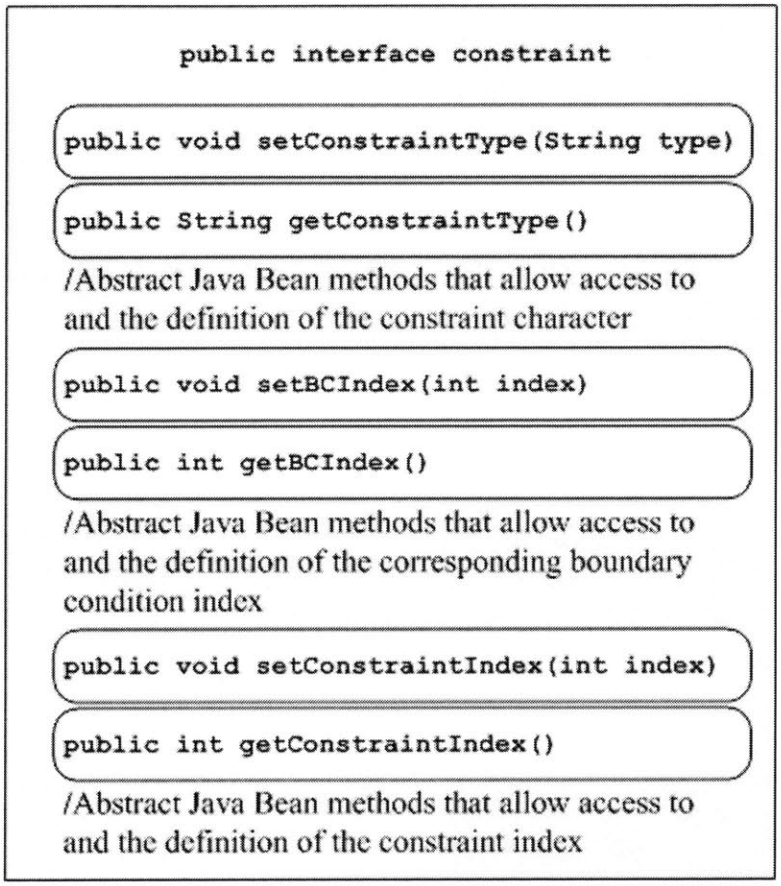


Figure 33 Main Abstract Methods of the `Constraint` Interface

Each `Interval` object contains two `BCondition` objects that represent the left end and right end boundary conditions of the interval and two `String` objects, `shearCharacter` and `momentCharacter`, that respectively identify the profile of the shear and the moment diagrams within the corresponding beam segment (Figure 35). These `String` objects depend on the make up of the boundary conditions and on the relative position of the interval along the beam. `ShearCharacter` strings can be defined either as "no shear condition" or as "constant shear". A "no shear condition" occurs when the shear along the interval is

zero while "constant shear" occurs when the value if the shear diagram is constant within the interval and not equal to zero. Similarly, `momentCharacter` strings can be defined as "no moment condition", "constant moment" or "linearly varying moment". A "no moment condition" occurs when the moment along the interval is zero while "constant moment" occurs when the value if the moment diagram is constant within the interval and not equal to zero. A "linearly varying moment" condition occurs when the value of the moment varies linearly which is normally associated with an interval of "constant shear" (Figure 34). The nature of these shape strings are consistent with information contained in Table 10 through Table 15.

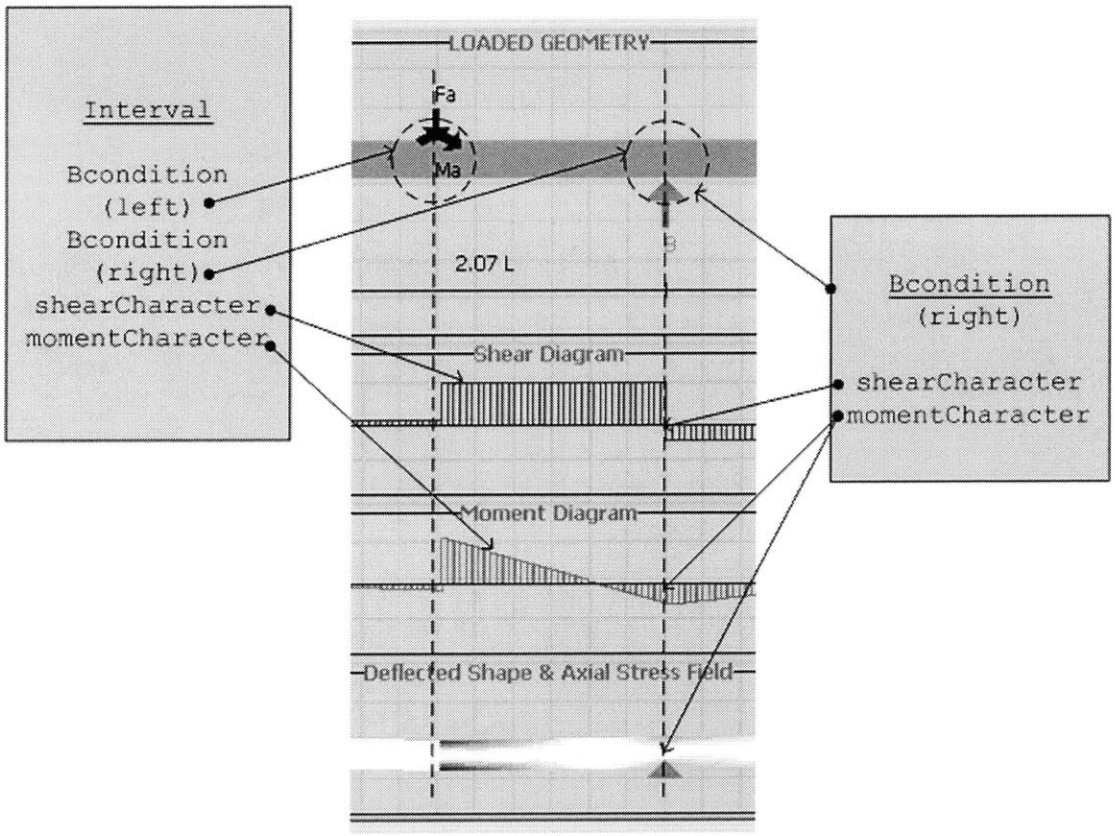


Figure 34 Interval Representation for Assessment and Feedback

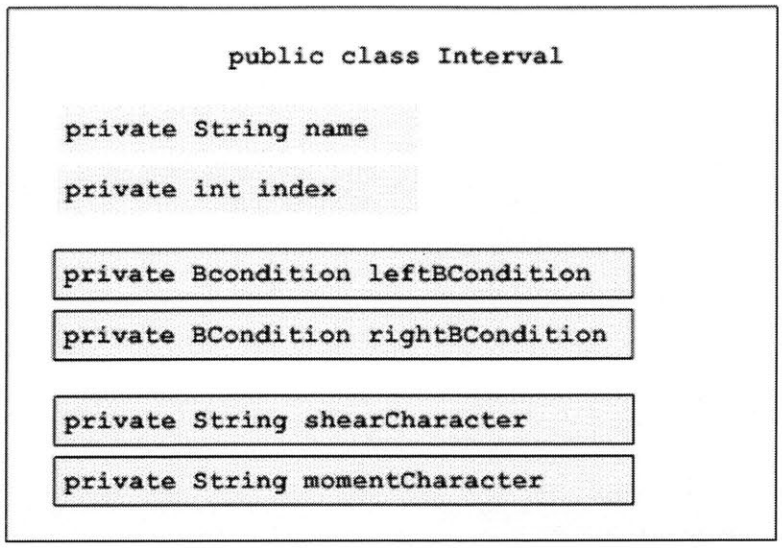


Figure 35. Principal characteristics of the `Interval` class

A `BCondition` contains descriptors that qualitatively characterize the behavior of the beam with respect to shear, moment, rotation and defection at the point where the boundary condition is defined. The descriptors associated with shear and moment consist of `String` key and `Boolean` state variable pairs that are contained in the Java `Hashtable` objects `shearCharacter` and `momentCharacter`. The keys describe characteristic structural response features while the value (`true` or `false`) of the state variables indicates their relevance at the boundary condition. The keys considered in `shearCharacter` are "sudden jump" "sign change" and "local extrema", while the keys considered in `momentCharacter` are "sudden jump", "local extrema" and "slope change". These identifiers depend on the boundary constraints that define the boundary condition and are consistent with information contained in Table 7. For example, if the boundary condition consists of an intermediate hinge support, the String key "sudden jump" with a state variable value equal to true is stored in `shearCharacter`, while the `String` " slope change " with a state variable equal to true is stored in `momentCharacter`. As all Java `Hashtable` objects `shearCharacter` and `momentCharacter` can accommodate any number of key/state descriptors. As a result a boundary condition

may be defined by more than one descriptor. For instance, if the shear diagram changes sign over an intermediate hinged support, the moment diagram would experience both, a local maximum/minimum and a change in slope. As a result, two descriptors, "local extrema"/`true` and "slope change" /`true` should be stored in `momentCharacter`

The descriptors associated with beam deformation consist of two `Boolean` variables, `rotation` and `displacement`. A `true` `rotation` value indicates that the boundary condition point may experience angular deformations while a true `displacement` value indicates that the boundary condition point may experience transverse linear deformations. Similarly to `shearCharacter` and `momentCharacter`, these values depend on the boundary constraints that define the boundary condition and are consistent with information contained in Table 7. For example, if the boundary condition consists of an intermediate fixed support, the value for both, `rotation` and `displacement` would be `false` (Figure 36).

When two or more constraints coincide at a point (within a predefined tolerance), they are assigned to a single `BCondition` object. In these cases an algorithm interprets the character of each constraint and, if necessary, assigns reasoned values to the various `BCondition` descriptors. For instance, a concentrated force is associated with jump in shear and a continuous moment. A moment support is associated with a shear release and a sudden change in the moment. If these two constraints coincide at the same boundary condition point, the descriptors of the `BCondition` object would contemplate a jump in shear and a jump in the moment. In the case where a concentrated moment coincides on a hinge support, the moment diagram would exhibit an abrupt change in value and a change in its slope. As a result the two descriptors, "sudden jump"/`true` and "slope change" /`true` would be stored in `momentCharacter`.

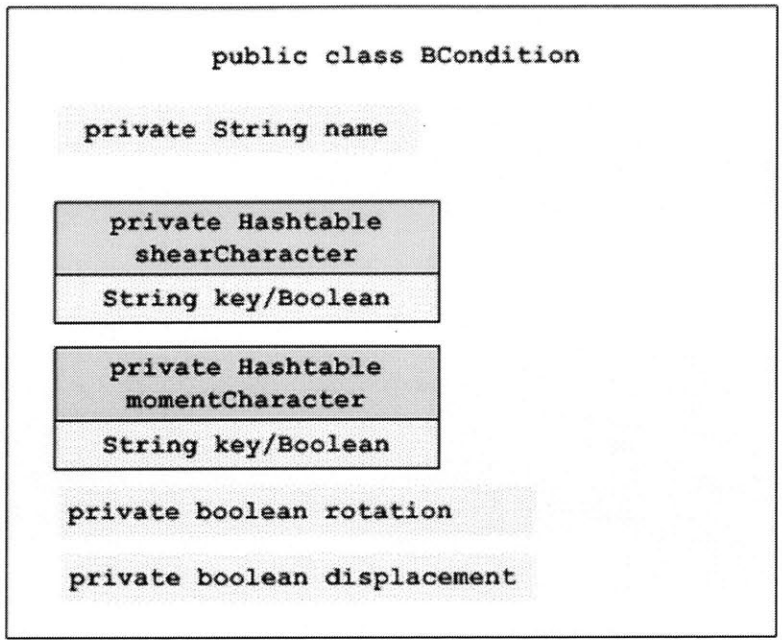


Figure 36. Principle Characteristics of the `BCondition` Class

KNOWLEDGE REPRESENTATION PROCESS

The `genNodes()` method of `Calculate` is called after a beam model is calibrated by the Preprocessing Module. After this method defines a precise representation of the beam model, it calls the following methods contained within the `Input` object of the model that is being calculated: `sortConstraints`, `createBConditions()`, `createIntervals()` and `assignIntervalCharacter()`.

`Input.sortConstraints()` finds and sorts the constraints of the model according to their position along the beam. Then, `createBConditions()` incorporates these constraints into a `Vector` of `BCondition` objects. This process consists of instantiating all needed `BCondition` objects and assigning their character strings according to the corresponding boundary constraints. `createIntervals()`, instantiates the `Interval` objects according to the content of the `BCondition Vector`. These objects are grouped in a `Vector` held by

`Input`. Finally, `assignIntervalCharacter()` defines the `shearCharacter` and `momentCharacter` strings of each interval.

In general, when the environment is in Assessment mode, information contained by `Input` objects is delivered to the reasoning agents through static methods of the Java Class `Evaluate`. These data are then matched against the rules of the knowledge bases. This process sets off inference processes that allow the agents to make decisions and to take action either by direct program intervention or by calling methods of other objects that are part of the application. The nature of these interactions depends on the application and on the type of assessment exercise presented to the student.

`Input` objects contain all the modeling information used by both the simulation modules and reasoning agents, however, they do not contain result data. This later information is contained by objects of class `Output`. Much like an experienced human tutor, these agents reason qualitatively about a problem without having to always rely on calculated numerical information. `Output` information is accessed by the reasoning agents only when they find it necessary such as when determining whether the student has been given sufficient information to solve a problem. This information is accessible through calls to methods contained in `Output` (Figure 37).

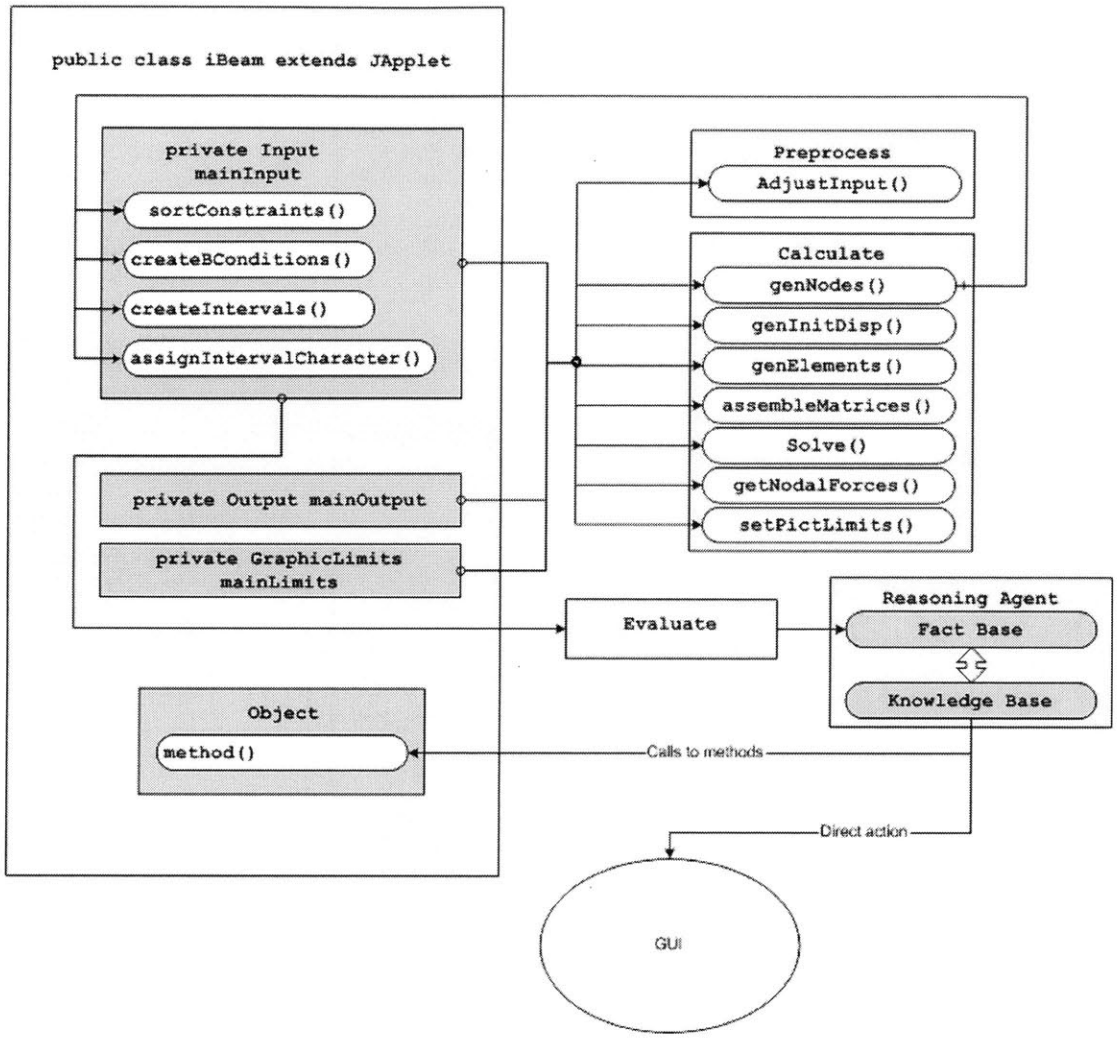


Figure 37 Knowledge representation procedures

6.2.2 PROBLEM PRESENTATION AND FEEDBACK STRATEGIES

The strategy used for problem presentation is consistent with the hierarchy of the Conceptual Model. Initially, problems presented to the student address the most general hypotheses. Gradually, exercises require from the learner to incorporate more specific knowledge. For instance, the first levels of assessment available to the student emphasize interval recognition and boundary-constraint/response-pattern identification. When these concepts are understood, the student is requested to recognize response pattern correlations and to integrate heuristic relations.

Feedback and assessment are also based on this hierarchy. When assessing a student's response, the program first checks for consistency in defining the intervals. If this is correct, it then addresses more specific issues such as whether the student was able to correctly identify the nature of the changes at the interval boundaries and the nature of the response diagram profiles along the intervals. If the student makes a mistake, feedback addresses exclusively the lowest hierarchical level where the answers are incorrect. For instance, if the student correctly identifies the beam segments where intervals occur but cannot identify the corresponding (interval) boundary constraints; feedback will hint solely on constraint recognition. Feedback concerning interval recognition (lower hierarchical level) or interval character recognition (higher hierarchical level) would not be given to the student. This strategy aims to eliminate repetition and avoids introducing concepts the student may know about but has not incorporated in her knowledge schemes.

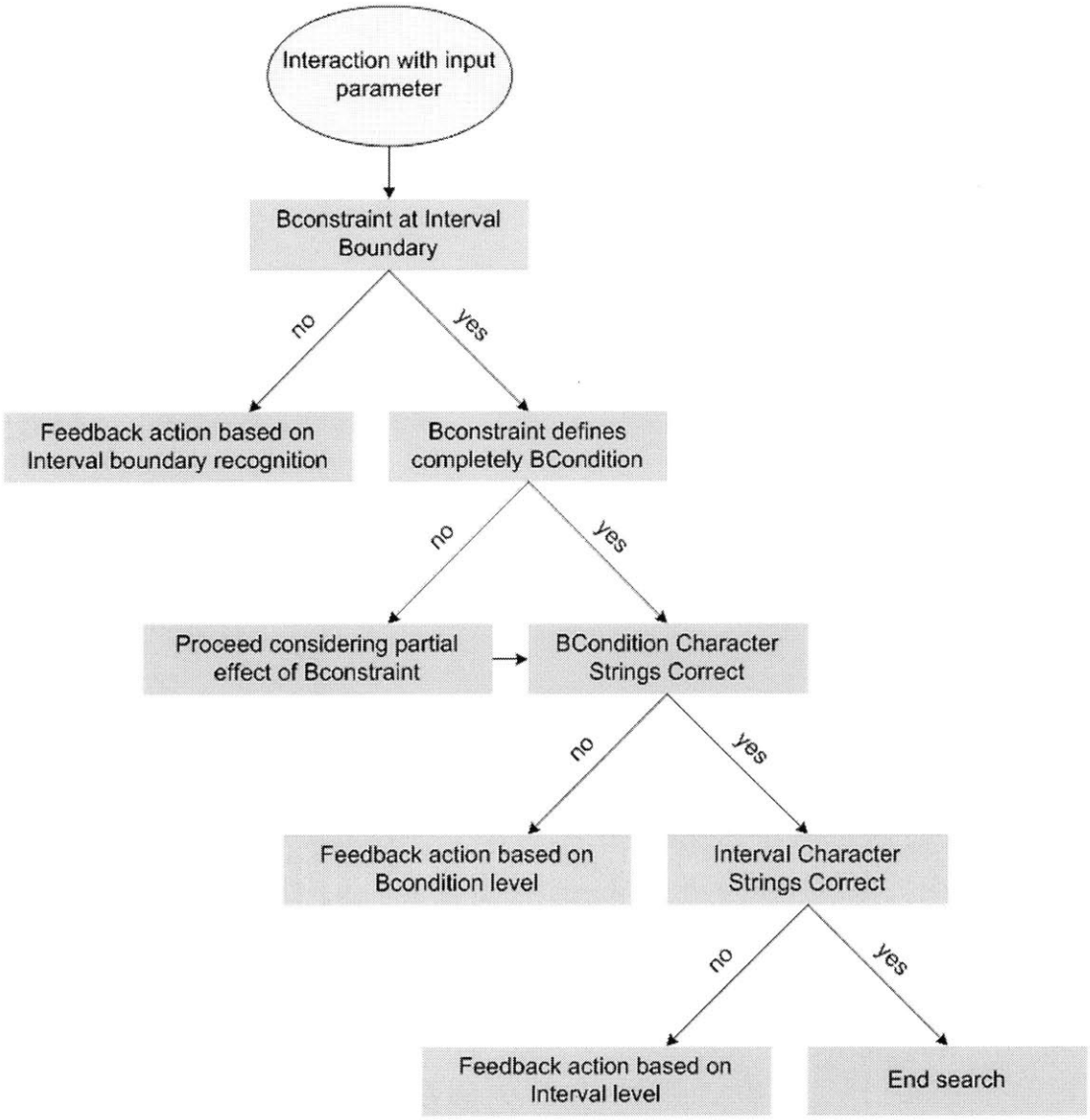


Figure 38 Simplified overview of Assessment and Feedback Strategy.

CHAPTER 7. POINTLOAD

PointLoad is an application that supports the teaching of concepts related to beam Bending Action. This application was developed to calibrate the strategies of the methodology proposed for the development of simulation based Experiential Learning Environments. PointLoad was developed as an Applet using the Java programming language and JESS a Java based Rete expert system shell developed by Ernest J. Friedman-Hill of Distributed Computing Systems at Sandia National Laboratories in Livermore, CA (Friedman-Hill, 2001). The following sections describe scope of PointLoad, and explain the interaction and the pedagogical purpose of its functionalities.

7.1 EXPERIMENT HYPOTHESIS SPACE

PointLoad's scope is limited to handle a single span beam loaded with a concentrated force (Figure 39). Since the beam constitutive relations reveal linearity with respect to the loads any loading condition can be viewed as a linear combination of concentrated forces (concentrated moments can be viewed as two equal opposite forces acting close to each other). As a result, for purposes of assessing the methodology, this limitation narrows the domain with little loss of generality. In addition, a single span beam is capable of incorporating all the relevant boundary support conditions and has to address most of the continuity relations associated with bending action.

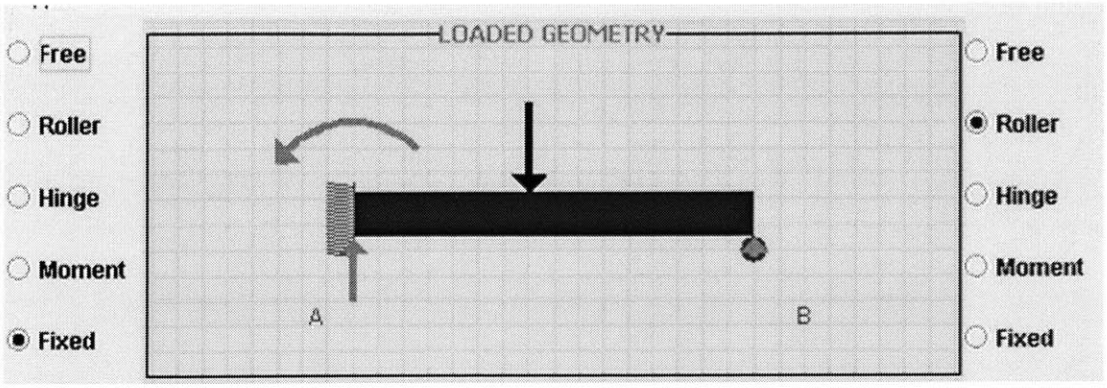


Figure 39 PointLoad: Experiment Hypothesis Space

7.1.1 THE GRAPHICAL USER INTERFACE

As it was mentioned in Chapter 6, Pointload orchestrates the functionalities of its three main components, an interactive simulator, a virtual tutor and a student model, through a control module via a graphical user interface (GUI). This interface consists of a single frame divided into two sections: the *Task Window* and the *Control Window* (Figure 40). The Task Window placed on the lower half of the GUI, consists of an explanatory text area flanked by two columns of menu buttons located along the window margins. These buttons allow the user to access and update the student model, navigate between simulation and different assessment modes and support the interactivity of these two tasks. While in simulation mode, the explanatory text area displays information about the structural model and interaction process. During assessment, the text area displays feedback and post evaluation information.

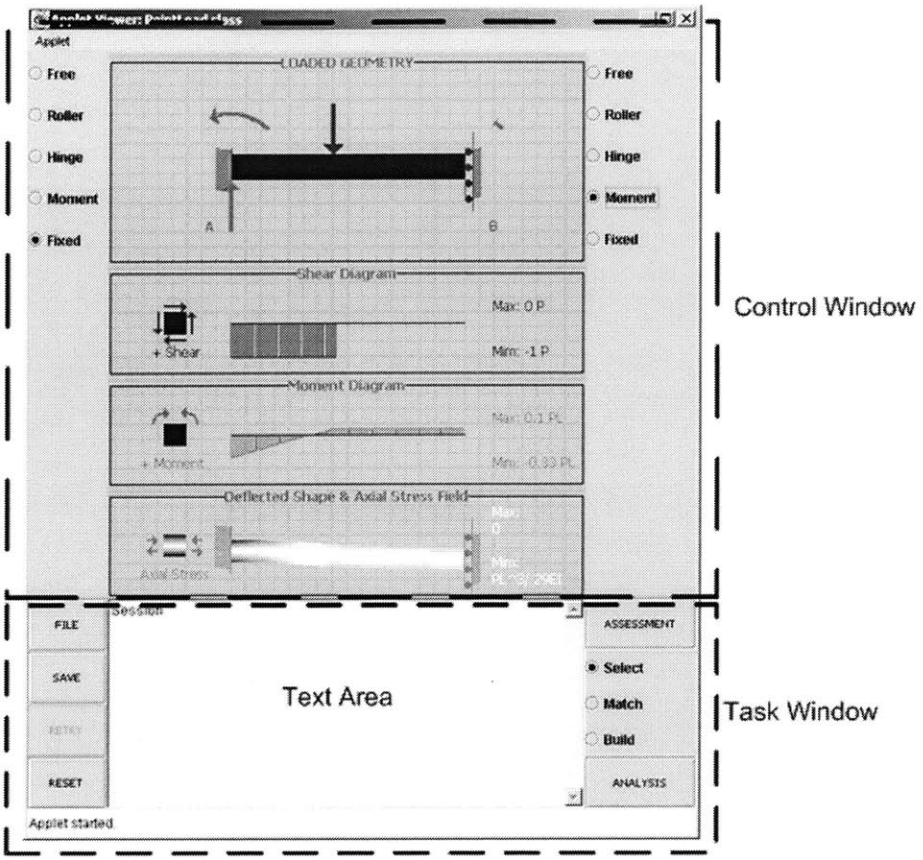


Figure 40 PointLoad: Graphical User Interface in Simulation Mode

The Control Window, located above the Task Window, is configured according to the different modes supported by PointLoad. It is, however, always organized such that control buttons are always located along the right and left margins while the center portion corresponds to the main graphical display panels that provide pictorial information of the beam and its structural response. The graphical capabilities and the graphical user interface items such as buttons and text areas make use of Java Swing classes.

Ease of use and minimum distraction are the principles that guide the design of Pointload's graphical user interface. The idea is for the user to always have immediate access to the features of the system with out having to, for instance, search within submenus. In addition, interaction with the interface has been designed to be

intuitive; for example, controls are always along the margins, while information is always displayed in the middle of the main frame.

7.1.2 THE SIMULATION

The simulator is at the core of the learning environment. Its primary function is to interactively model and illustrate the structural response of single span beams subjected to a concentrated force. The simulation responds to changes of the support boundary conditions at each end of the beam and to the position and sense of the load. The structural response of the beam is illustrated through visual representations of the free body diagram, shear diagram, bending moment diagram and stress deformation diagram. These diagrams are arranged into a column of four display panels placed at the center of the Control Window in the GUI (Figure 41).

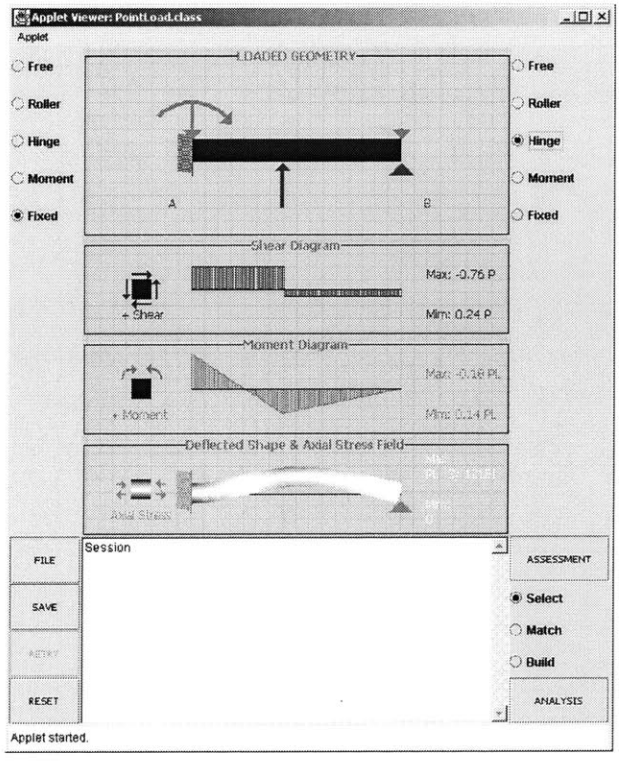


Figure 41 PointLoad Screen in Simulation Mode

The topmost diagram consists of a `ControlPanel` with the configuration of the loaded beam and the corresponding support reactions. A mouse activated click-and-drag control feature lets the user interact with the load and change the its sense and its position along the beam. Two sets of radio support buttons flank this panel. These click activated buttons facilitate the selection of the desired support boundary conditions at either end of the beam.

The panels bellow the `ControlPanel` consist of `BeamPanels` that display the shear, moment and stress-deformation diagrams. In addition illustrating the deflected shape, the stress-deformation diagram depicts the axial stress distribution along the beam; red is used to indicate compression and blue for tension. The color intensity indicates the relative stress magnitudes.

The beam configuration and the response measure diagram displays are drawn on a common horizontal scale. This dimensioning and the arrangement of the display panels is intended to facilitate the recognition of the functional relationships that exist between the input parameters and the response measures. In addition, having the structural response patterns on top of each other facilitates establishing the relationships that exist between the response measures themselves.

The vertical scale of each diagram is determined according to the size of the panel. This automatic scaling is necessary for practical purposes and it does not affect the didactic effectiveness of the simulation

THE INTERACTION

The pedagogical strategy in simulation mode is to engage the student in explorative action. To facilitate this engagement, the interaction between the user and the simulator is designed to require minimal effort. For instance the structural model is automatically generated and displayed as soon as the application is deployed.

Changes to the model boundary conditions such as support types and load position are accomplished entirely through easy to use buttons and drag and drop features. In addition, having graphical displays endowed with these features allows the user to modify of the structural configuration of the virtual model in an intuitive manner. Furthermore, since the simulator responds instantaneously to these changes, learners can gain an intuitive sense or confirm reasoned predictions while freely exploring structural and loading configurations. Moreover, to avoid information overload, only changes in structural parameters that are essential to the conceptual understanding of beam Bending Action are enabled. For instance, the relative distribution of the response measures embody the functional relations that make up the hypotheses of the Conceptual Model. These distributions are revealed by the shear, moment and deformation diagrams. Through reflective observation, these diagrams are assimilated into pictorial and symbolic representations of structural behavior. The magnitude of these representations is not as significant as it is their outline. Consequently, for conceptual understanding, the absolute numerical value of the response measures is not as important as it is their relative distribution. In a single span beam, the position and sense of the loads and the nature of the support boundary conditions have an important impact on both the magnitude and the relative distribution of the response measures. Consequently these are the parameters that control the simulation. On the other hand, the magnitudes of the loads and of the span length have an effect on the scale of the response measures (shear, moment, deformation, reaction forces) but not on their relative distribution. As a result, their values remain constant and can not be accessed. Finally, keeping the bending rigidity fixed further simplifies the structural model while maintaining the capacity to illustrate most the important relationships of the Experiment Hypothesis Space.

Discovering typical response patterns and establishing the links between response and input parameters is a process facilitated by comparison and contrast of cyclical simulation output. As a result, to effectively support the development of meaningful

iconic and symbolic representations of structural behavior, the visual output is richly detailed and the interactive response of the simulator is on real time. In order to meet these requirements, PointLoad relies on three main elements: the Problem Preparation Module, the Finite Element Engine and the Display Module.

7.1.3 THE EXPLANATORY INTERFACE

In addition to providing simulation output, the tutoring environment is capable of associating the beam structural configuration and response to theoretical principles relevant to beam Bending Action. This capability can be turned on or off by the user. If turned on, this information is relayed to the user through as explanatory text in the Text Area panel located underneath the simulation displays. Symmetry/Anti-symmetry and Determinacy are the two primary principles that the tutoring environment aims to recognize. The identification of these principles is automatic and occurs concurrently with the simulation. The information is also provided along with the simulation output (Figure 42).

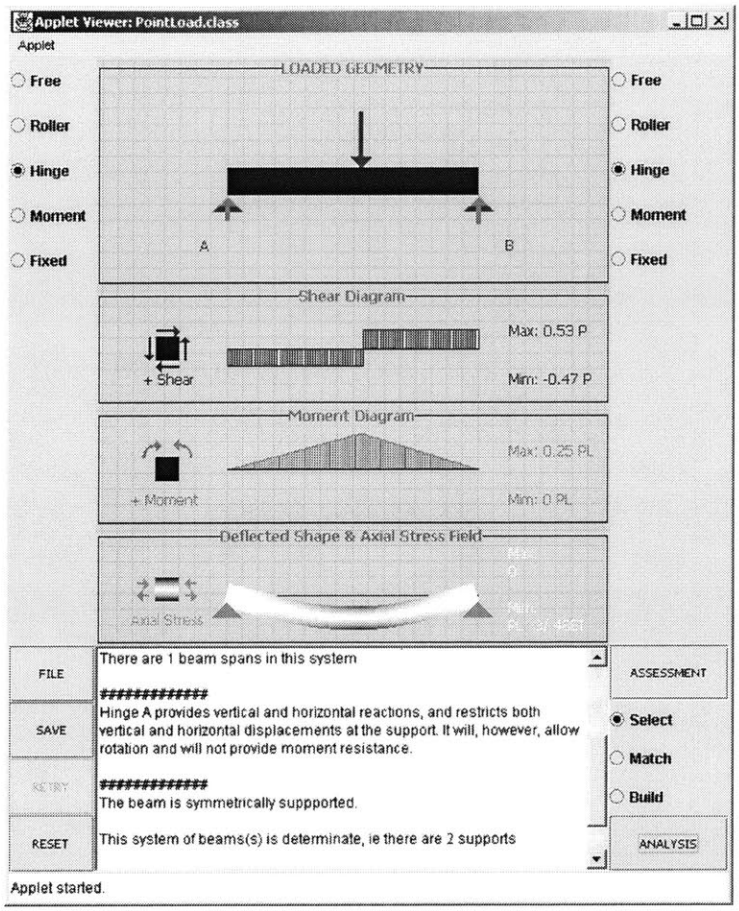


Figure 42 PointLoad: Simulation Screen with Explanatory Interface

The explanatory interface consists of an interfacing Java class and a JESS based inference mechanism that identifies the make up of the structure and associates it to relevant theoretical concepts. A reasoning agent performs the association of modeling data to theoretical concepts. This agent receives information on the structural make-up of the simulation model and processes it through an array of production rules. These rules constitute a declarative virtual representation of the conceptual knowledge held by these concepts.

7.1.4 THE VIRTUAL TUTOR

In order to elicit reflective action, PointLoad depends on a strategy that combines qualitative problem solving with evaluation and feedback. Problem presentation introduces Epistemic Conflict while feedback facilitates the reflective process. These tutoring actions are implemented by the Virtual Tutor through three assessment exercises: Select, Match and Build. Each of these exercises experiments with different approaches to problem presentation and feedback strategies that are designed to show a logical progression in terms if their complexity and interactivity. Select exercises request from the learner to reflect upon the nature of the relationships between the input parameters and the structural response. Tutoring support consist of delayed text-based explanatory feedback. Engagement in Match exercises allows the student to explore and reflect upon the correlations that exist between the structural response measures. Tutoring support consists of passive feedback based on calibrating the nature and difficulty of problem presentation. Build exercises promote reflective action by requesting the learner to interactively reverse engineer a problem. Tutoring guidance is also interactive and consists of immediate feedback offered in response to each of the input steps the learner makes toward constructing an answer.

The pedagogical strategy on which the selection of these exercises is based, consists of implementing a setting where the student is allowed to progressively built the capacity to reason and discover the rules of the domain. Select exercises address the most fundamental relations of the Conceptual Model, namely interval recognition and the relationships between the input and the character of the response. Match exercises focus on the more complex interactions relative to the relationships that exist among the response measures. Finally Build exercises require the student to reflect upon all the possible relations that describe the domain.

In addition, these exercises require from the learner to progressively get more involved in the solution process. In Select type of exercises, the student needs to select an answer. Match exercises require from the student to assemble an answer. Finally, Build exercises require form the student to actively get involved in constructing the solution.

To accomplish these tasks, the Virtual Tutor interprets the learner's level of knowledge, generates and proposes exercise problems, assesses the response, selects and implements feedback strategies and controls the interaction while in assessment mode. These actions are achieved primarily by two modules, the Student Model and the Assessment/Feedback module.

STUDENT MODEL

The primary purpose of the student model is to furnish the Virtual Tutor with a reference basis to guide and motivate the student during assessment. To accomplish this task, the user interactions on each assessment exercise are profiled in a record which consists of a Java object file of the class `StudentRecord`. A user who whishes to engage in problem solving activities, must first create a personal `StudentRecord`. This file is accessed and read by the Virtual Tutor to determine the level and kind of assessment exercises the student should get. During assessment, the `StudentRecord` is modified. This password protected file can then be stored and repeatedly accessed (Figure 43). The information accumulated in this file is used by the Virtual Tutor as a measure of the student's ability to identify and understand the knowledge that governs the structural behavior of a generalized beam interval. This section presents the structure of the `StudentRecord` and its subclasses. The way in which this file is used will become apparent when the Assessment/feedback module is described.

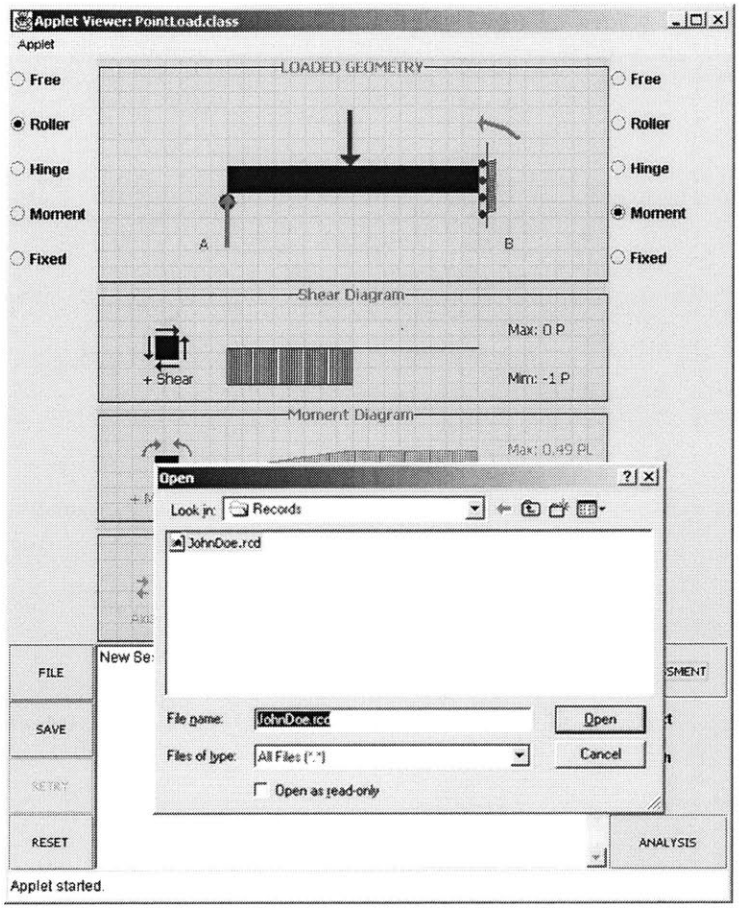


Figure 43 PointLoad: Student Record File Screen

`StudentRecord` contains personal data such as the student's name and password, and performance information on the three types of assessment problems implemented by PointLoad. This information is contained in Java objects of the class `Exerice1`, `Exercise2` and `Exercise3` which correspond to Select, Match and Build exercises respectively (Figure 44).

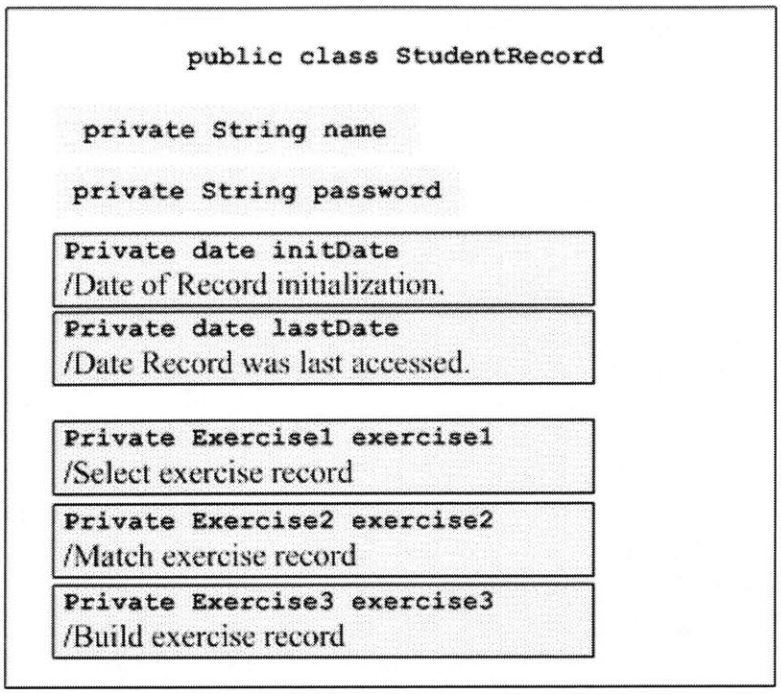


Figure 44 `StudentRecord` Class Structure

Each `Exercise` (exercise1, exercise2, exercise3) object contained be the `StudentRecord` file is made of one or more `TrackingRecords`. `TrackingRecords` are objects that monitor the student's answers on the each of the assessment exercises. `Exercise1` objects contain three `TrackingRecords`, one for each type of response pattern: shear diagram, moment diagram and stress deformation diagram. `Exercise2` and `Exercise3` objects compile the answers in one record so they contain one `TrackingRecord` each, see figures Figure 45,Figure 46 and Figure 47.

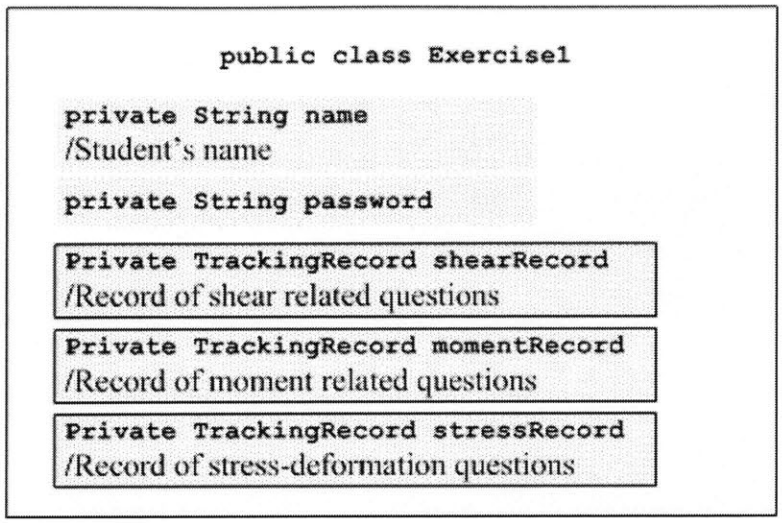


Figure 45 `Exercise1` Class Structure

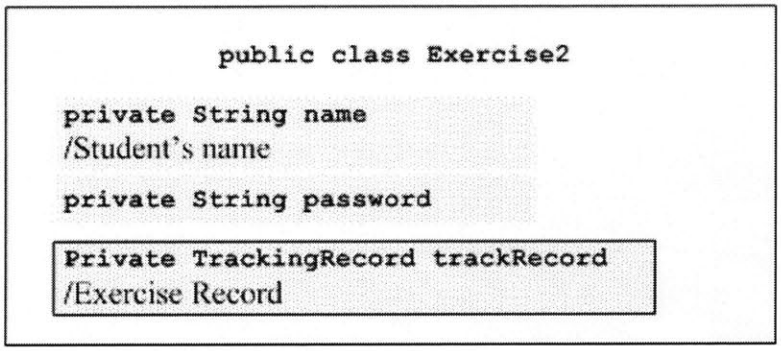


Figure 46 `Exercise2` Class Structure

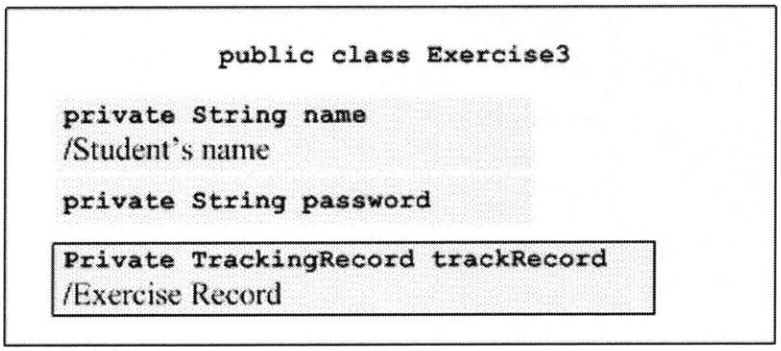


Figure 47 `Exercise3` Class Structure

Each TrackingRecord consists of an object of the Java class Hashtable and an object of the class ForceRecord. This Hashtable has five keys, each describing a support boundary condition. An object of the class SupportRecord is linked to each these keys (Figure 48).

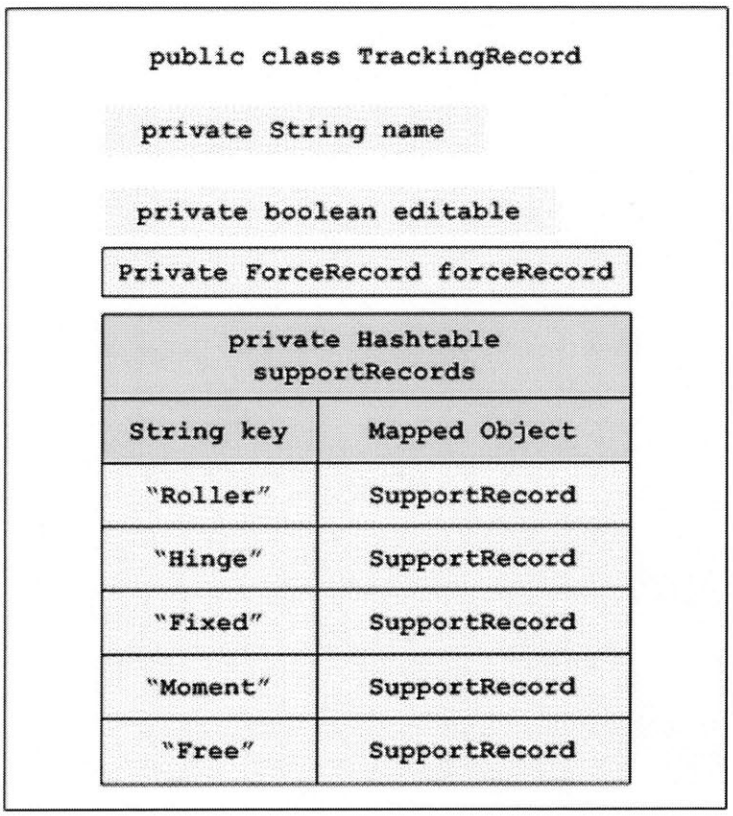


Figure 48 TrackingRecord Class Structure

SupportRecord objects contain a string attribute BC, for boundary condition, and integer counters for the number of right and wrong attempted trials (Figure 49). It will become clear in the next section how this Hashtable acts as a dynamic index of the student ability to appropriately identify the boundary conditions of the various support types.

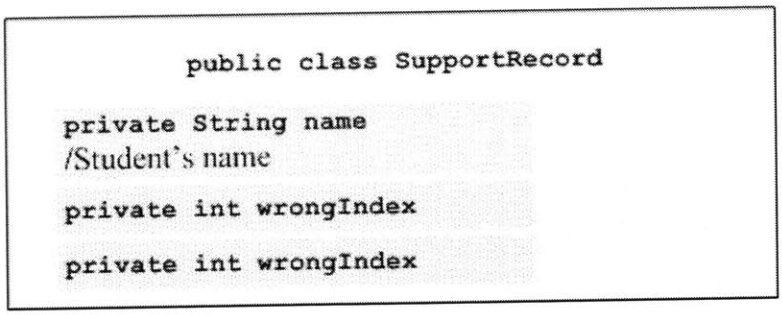


Figure 49 `SupportRecord` Class Structure

The `ForceRecord` objects have Boolean values indicating the validity (true or false) for load position and load sense and integer counters for the number of right and wrong attempted trials for both load position and sense (Figure 50) .

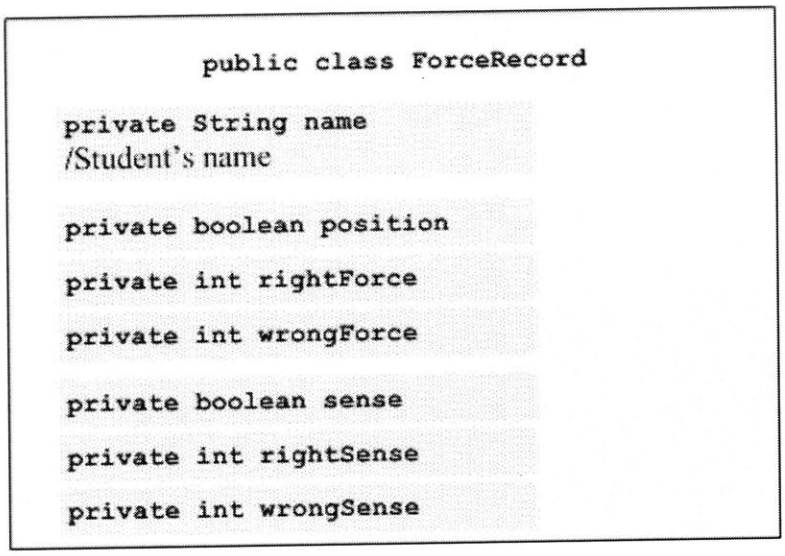


Figure 50 `ForceRecord` Class Structure

ASSESSMENT/FEEDBACK MODULE

The Assessment/Feedback module consists of three elements: the *Select* component, the *Match* component and the *Build* component. Each of these components performs three essential tasks, problem generation, evaluation and feedback which support each of the three assessment modes available in PoinLoad. In order to perform these tasks,

the Control Module first reads the information contained in the `StudentRecord` file. This information is used to generate a problem that is presented to the student. Reasoning using qualitative domain knowledge, the answer provided by the student is then processed and feedback content is selected and presented.

These components consist of a combination of Java classes and JESS rule based reasoning systems. These Java classes are interfaces that feed data from the simulating environment to small dynamically generated databases to be used by the rule based inference mechanisms which embody declarative virtual representations of the knowledge and procedures pertinent to each assessment/feedback modes. These mechanisms evaluate the student responses, update the student record, generate and provide guiding advice and control the interaction by editing attributes of java objects of the general programming infrastructure. The underlying strategy that supports these inference mechanisms consists of forward and backward chaining. The programming of the rules is performed in the JESS programming language.

7.1.5 SELECT EXERCISES

Select exercises are multiple choice type questions whose objective is to get the student to ponder upon and draw conclusion about the associations that exist between the fundamental character of the response patterns, and their corresponding boundary conditions and input parameters. The pedagogical objective of these exercises is to focus the reflective process on the primary relationships of the domain (See Table 7 through Table 15). Since this is the first level of assessment, feedback is narrative and seeks to inform to the user about basic heuristics and rules that would allow her/him to identify the correspondence between response patterns in shear, moment and stress-deformation diagrams and a specific set of input parameters.

INTERACTION

When the Select mode of assessment is chosen, the student is presented with four displays. The top figure corresponds to the *main beam* while the diagrams bellow it correspond to shear, moment or stress-deformation diagrams. The student is then asked to choose the diagram that she/he thinks correspond to the main beam Figure 51. The main beam and the question diagrams correspond to loaded configurations dynamically generated and solved by the simulator.

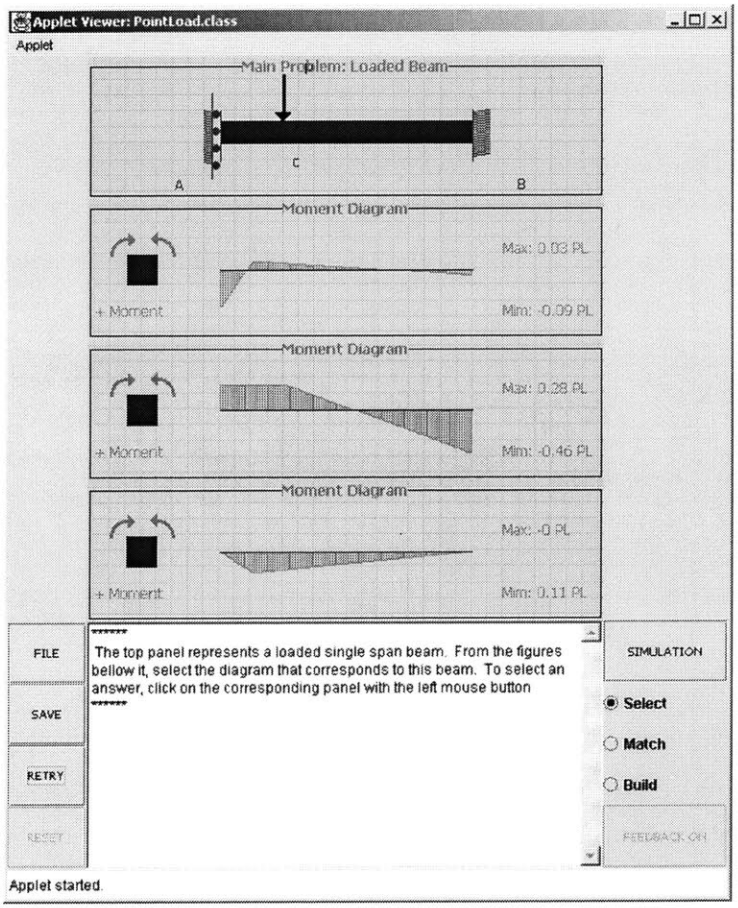


FIGURE 51 POINTLOAD SELECT EXERCISE.

When the answer is selected (by clicking on the chosen display panel), the program feeds the information contained in the input attributes of both the main problem and

the response problem to the Select agent. These inputs are then used to generate a fact database that is used by the agent's set of rules to evaluate the answer and provide feedback. To evaluate an answer, the system first determines if the beam intervals are correctly matched by comparing the position of the load. It then checks the consistency of the load sense and then it checks for correctness at the supports. Correctness exists if an input attribute of the response problem matches the corresponding value on the main problem. If an attribute is incorrect, the system triggers an inference mechanism that can establish the nature of the mismatch and provides textual guidance to the student. This guidance consists of a comparative description between the structural behavior of the selected and the main beams. Although this description focuses on the incorrect parameter, the system considers the overall configuration of each of these beams when matching the rules that describe their structural behavior (Figure 52).

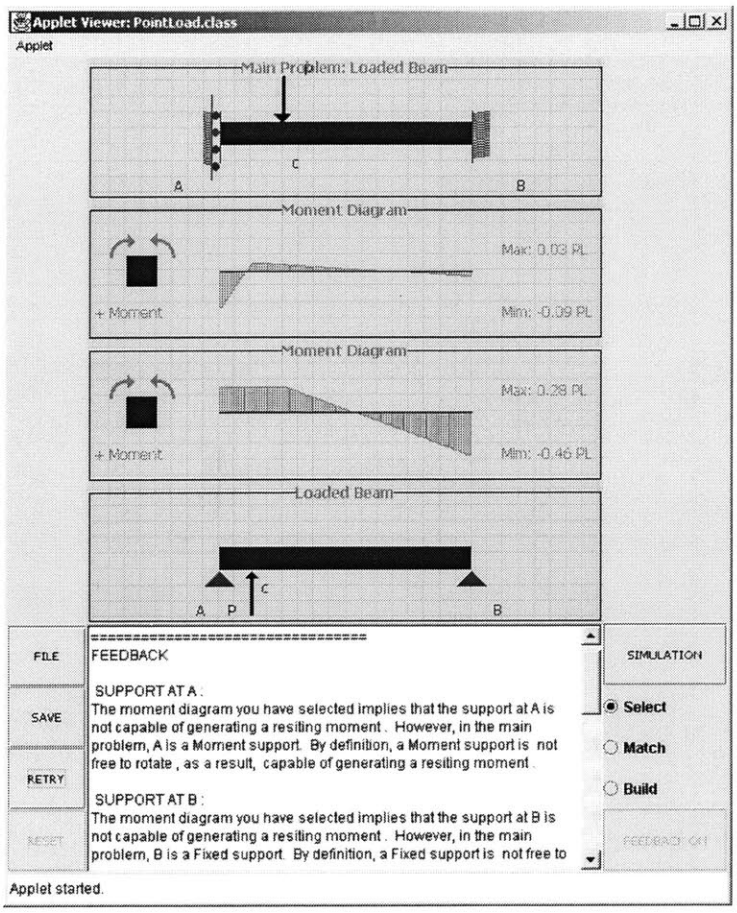


Figure 52 PointLoad Feedback Screen: After the student selects a diagram, the corresponding panel shows the configuration corresponding to selected answer. In addition the Tutor provides narrative feedback comparing the selected beam and the main beam.

The system understands the information that can be read on each of the response diagrams presented to the user. For instance, in a set of shear diagrams, a student should be able to recognize a fixed support from a free end. A fixed support is capable of providing a vertical reaction so it can be associated with a non-zero shear value while a free end would have to show a zero shear value. A shear diagram, however, would not have enough information in it to allow the student to differentiate between a hinge and a fixed support. In both cases, these supports have the potential to show non-zero values for shear. As a result, to determine a correct answer form an incorrect answer, the system can not simply compare directly the attributes of the `Input` object of the main beam against the attributes of the answer `Input` object.

To check for correctness, the system has an inference mechanism that allows it to reconstruct input boundary conditions from the information present in the diagram proposed by the student and compares this information against the attributes of the main input.

As the student interacts with the assessment module, the Student Model keeps track of her performance. Exercise1 of `StudentRecord` has one `TrackingRecord` for each type of response diagram. At every Select exercise trial, the student has to choose from a set of shear, moment and/or stress-deformation diagrams. The Tracking Record that corresponds to the type of diagrams presented in the exercise is modified upon receiving the user response. This modification consists of identifying the `hashtable` keys that match the support boundary conditions of the main beam and assigning the response boundary conditions to the `BC` attribute of the `SupportRecord` object held by these keys. If the `hashtable` key is equal to the `BC` attribute, then the support has been correctly answered. A correct correlation exists when the key in the `hashtable` matches its corresponding object's `BC`. If this correlation is correct the right index is increased by one, otherwise the wrong index is increased. To account for the load position and sense, a similar procedure modifies the `forceRecord` objects.

These records are not revealed to the student. They are used to dynamically calibrate the level of difficulty of the exercises and to determine the assessment options available to the user. The difficulty level of each exercise is adjusted by controlling the similarity between the attributes of the main beam and the models in the question set. The closer the attributes the more difficult it is to provide an answer. For instance, it is more difficult to differentiate between the response of two beams whose only difference is the sense of the load than it is between beams whose supports are also different.

When the student has successfully correlated all the items in the tracking records corresponding to shear, moment and stress-deformation of `Exercise1`; the program will introduce mixed diagrams (shear, moment, and stress-deformation) from which to select the one that corresponds to the main beam. This adds to the level of difficulty while allowing the student to stay in the Select assessment mode. At this point, the student is also allowed to access the Match and Build set of exercises.

7.1.6 MATCH EXERCISES

Match exercises are combinatory multiple-choice type questions that are presented to the student with the objective of determining whether she/he has understood the correlation that exists among structural response patterns and the relation between these patterns, and the input parameters and boundary conditions. Since at this level, the student is supposed to have already understood the "fundamentals", feedback is not as explicit as it is in the Select set of exercises. Feedback takes the form of controlling the presentation and generation of new problems.

When the matching mode of assessment is chosen, the tutoring environment dynamically generates ten inputs and ten corresponding FEM generated outputs. The program randomly assigns one of these inputs to the main beam. The student is then presented with four display panels. The top figure corresponds to the main loaded beam while the figures bellow it correspond to scrollable panels containing the ten shear, moment, and stress-deformation diagrams. The student is then asked to scroll through each of these panels and select a shear-moment-stress/deformation set of diagrams that she/he thinks correspond to the boundary conditions and input parameters of the main beam. The student can also access a side panel showing the full set possible answers (in random order) for each type of diagram by double clicking on the corresponding scrollable panel. A single click on the side panel selects the answer under the mouse pointer to be drawn on the main display window.

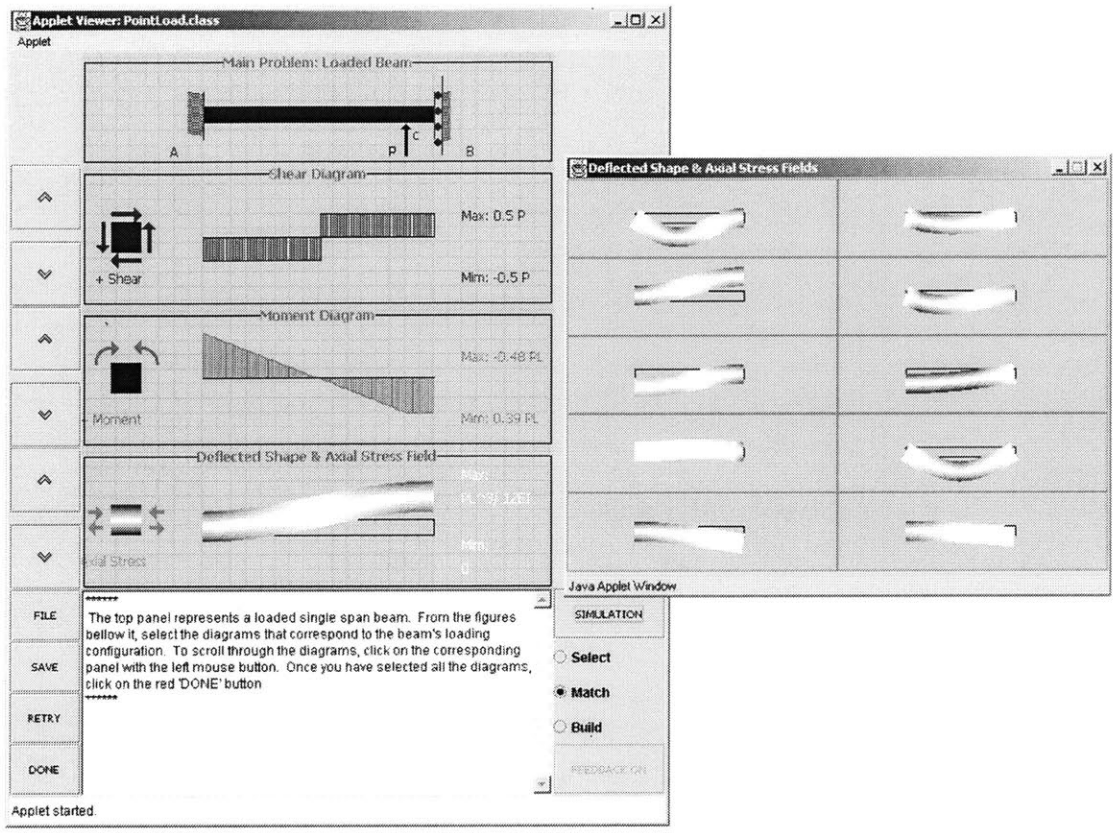


Figure 53 Match Exercise

When a complete set of diagrams is selected (by clicking on the DONE button), the information corresponding to the input of the main problem and the three response diagrams is passed to the Match exercise agent. This information is used to generate a small fact database that is used by the agent's rule based inference mechanism to evaluate the answer and provide feedback. This mechanism reads the information in each of the selected diagrams and checks for both consistency and for correctness. Consistency exists when the three selected diagrams exhibit characteristics that allow the reconstruction of physically possible boundary conditions and input parameters. For instance, a hinge would be consistently inferred if at the same support point, the shear diagram shows a non-zero value, the moment diagram shows a zero value and the stress/deformation diagram reads no displacement and displays a rotation. Correctness would apply if the reconstructed boundary condition matches the

corresponding parameter in the main problem. In this description a hinge or a roller would have to be present in the main problem for the support to check for correctness (Figure 54).

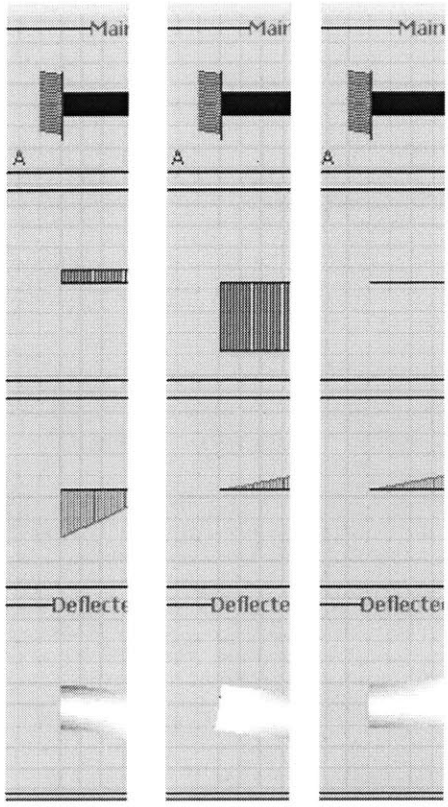


Figure 54 Left Diagram: Consistent and correct with respect to a Fixed support. Center Diagram: Incorrect, but consistent with respect to a Hinge support. Right Diagram: Inconsistent.

As in the Select type exercises, the system first works with the point where the load is applied and the sense of the load. It then tries to reconstruct the supports. If any of these items is inconsistent, the system triggers a set of rules that can establish the nature of the mismatch and provides a short textual explanation about the mistake being made. The student is then encouraged to try and solve the same problem again. The response choices on the scrollable panels, however, will be different. If, the

answer is consistent but not correct, the system replaces the boundary condition(s) of the main problem with the consistently inferred but incorrect boundary conditions. No textual feedback is provided. The student is, however, encouraged to try the exercise again with the modified problem. If on this try, the student makes the inverse consistent/correctness mistake made on the previous try, it can be inferred that he/she is mistakenly associating a set response patterns with an incorrect boundary condition. As a response, the system provides a thorough explanation of the behavioral characteristics of the two parameters that the student seems to be "mixing up". For instance, if the main problem shows a downward load and the student selects a set of diagrams that is correct except for that they correspond to an upwardly loaded beam; the system would switch the sense of the load on the main problem from downward to upward. If the student's answer to the modified problem shows a set of diagrams that correspond to a downward load, it can then be concluded that he/she has the sign convention reversed and that an explanation is needed.

Similarly to Select exercises, the Student Model is updated at every Match exercise trial. The `TrackingRecord` contained in `Exercise2` of `StudentRecord` is modified upon receiving the user response. Inconsistent support conditions are considered incorrect. The `SupportRecord BC` value assigned to an inconsistent support is nil. Consistent supports are processed as in Select exercises. To account for the load position and sense, a similar procedure modifies the `forceRecord` object contained in `Exercise2`. This exercise is completed when all the `hashtable` keys and `forceRecord` objects are correctly correlated. At this point the `Exercise3` is no longer updated.

7.1.7 BUILD EXERCISE

Build exercises are designed to reinforce the reflective process of correlating patterns of structural response to input parameters. While still being multiple-choice, they allow the student to engage in a more active role in building an answer, thus

encouraging a constructivist approach to learning. In this strategy, hints are provided instantly as the student selects the response parameters. As a result feedback becomes a guiding mechanism rather than a post evaluation advisor.

The Build assessment strategy consists of displaying a dynamically generated set of consistent response diagrams and then asking the student for the corresponding boundary conditions (Figure 55). The same graphical user interface layout used in simulation mode is recreated for this exercise. The controls used to manipulate the simulation are also used to build the exercise answer. As the student selects the input parameters for the answer, this information is instantly relayed to the Build agent. This utility then compares the characteristics of the selected input item with the characteristics of the main beam. Nothing happens if there is a match. Otherwise, the agent identifies the nature of the mismatch and provides textual and graphical instantaneous feedback. For instance, if an incorrect load position is selected, the load arrow on the main display panel flashes red, hinting the student about the mistake before she/he commits to an incorrect answer. In addition an explanation is given in the text panel. If the student keeps on repeating the mistake, the Build assessment agent reduces the exercise complexity by fixing the load to its correct position and encouraging the student to proceed with the other parameters. This process continues until the student arrives at the correct answer either by choosing the correct input parameters or by triggering corrective intervention of the Build agent. Rather than being a post evaluation advisor, this type of response turns feedback into a guiding mechanism that helps build an answer.

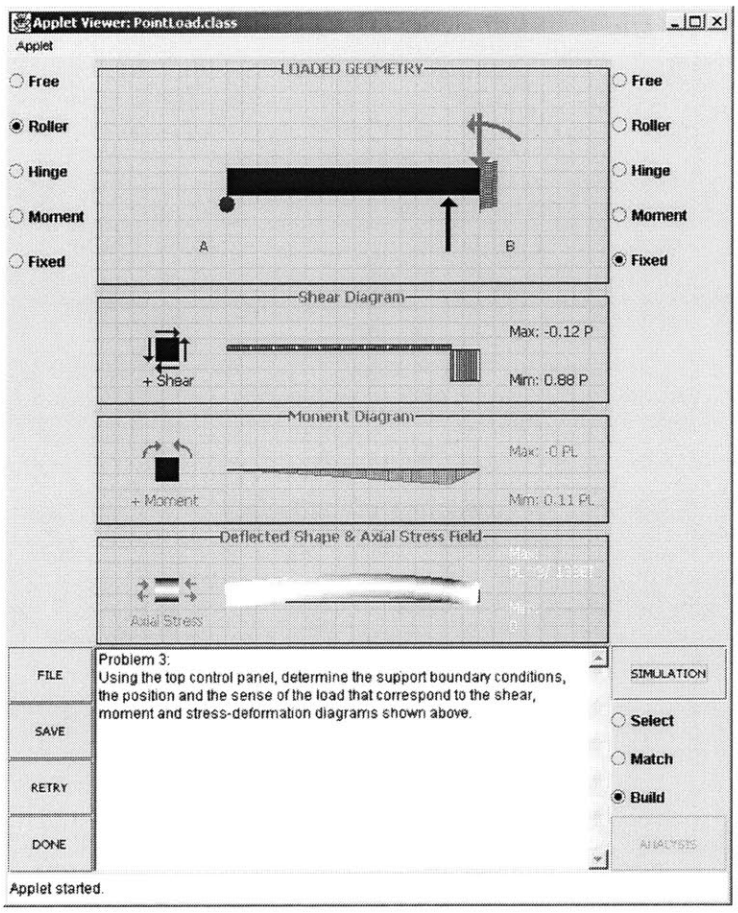


Figure 55 Build Exercise

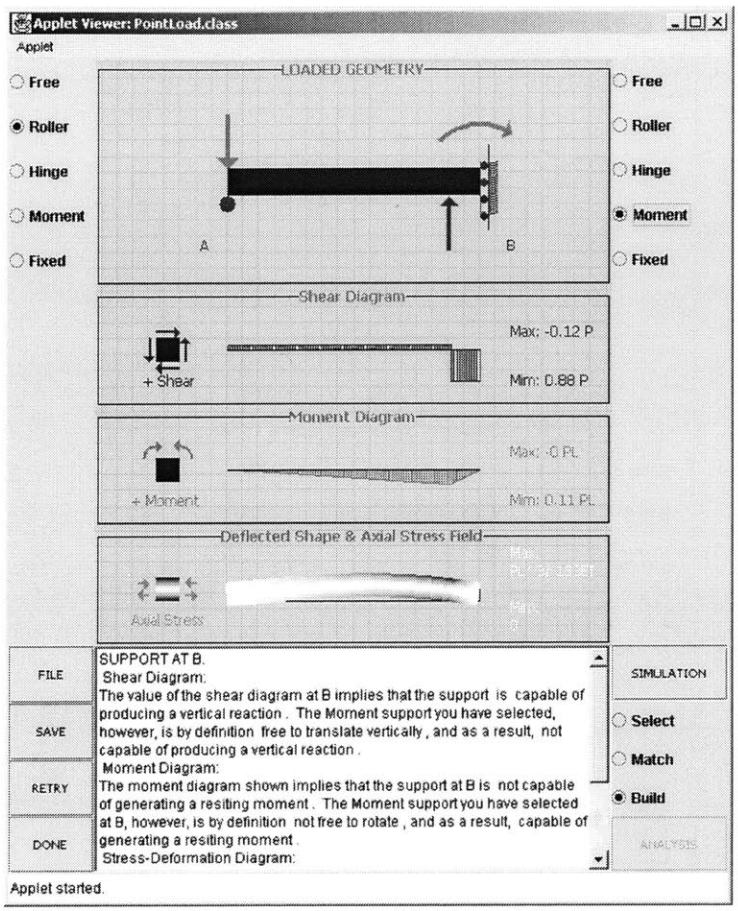


Figure 56 Build Exercise: Immediately after the student interacts with an input item, the environment provides both text based and graphics based feedback.

The Student Model is updated in a similar fashion as in Select exercises with the difference that there is only one tracking record to process. The exercise is completed when all the `hashtable` keys and `forceRecord` objects are correctly correlated. At this point the Exercise3 is no longer updated.

7.1.8 OBSERVATIONS

Usability tests performed on PointLoad. The sample set consisted of 22 subjects, of whom 7 were undergraduate Civil Engineering sophomore and senior students, and the rest were first and second year graduate students. All of the students had taken at least one course in structural mechanics and an analysis course. The tests consisted

of first allowing the students to freely interact with PointLoad in simulation mode until they felt that they had become familiar with the environment and "understood" the structural response of all possible load/support combinations. Then, the students were asked to comment on usability aspects such as control features and ease of use. Finally the students were observed as they engaged in assessment exercises.

The results of this test revealed that users appreciated and were engaged by the interactive capabilities of the simulation environment. The graphical user interface was found logical and simple to use. The manner in which the information is displayed was also appealing. It was, however, suggested that in addition to qualitative information provided by in the display diagrams, quantitative relational information be given.

This survey also revealed that users did not engage in a reflective process until question problems were presented to them. Most students experienced difficulties answering assessment questions that were very similar to the exercises they had interacted with during simulation. However, once they had been assessed on a particular item, other exercises posed no further difficulties. In addition, the Explanatory Interface was of little interest and in some cases was found distractive. This last result further confirms that unless there is an explicit motivation, learners do not engage in reflective action. As a result, uncalled conceptual information instead of being helpful may actually be detrimental during simulations.

Also, when not able to respond to a question, students preferred going back into simulation mode to justify the answers rather than wait for explanations. In many instances, even after seeing the answer, despite their background knowledge and restricted domain, they had to refer back to simulations in order to confirm what was offered by the feedback.

Because of the lack of interactivity, in most instances, select exercises were not favored as much as Match and Build exercises. Furthermore, long textual feedback was not as effective as short immediate and/or graphical feedback. It did not seem to be a question of not wanting to read the explanations that were provided or that the students did not understand or agreed with their content. The reply from most students went along the lines of *I agree with it, but I don't think that way about these problems*. This confirms that although the basic knowledge entities are generally accepted, the way each user builds explanations to complex conceptual problems is unique. As a result, it is best to create a setting where the student discovers the rules of the domain rather than to imbue ready made and previously justified answers.

Between Match and Build exercises, the later were perceived as more effective at engaging the student and in providing timely feedback. In addition, because of the multiplicity of screens Match exercises were sometimes characterized as graphically overwhelming.

CHAPTER 8. IBEAM

The experience gained in the development of PointLoad led to the creation of iBeam. This application extends the simulation capabilities and fine-tunes the assessment and feedback strategies used by PointLoad. These enhancements assimilate the results of the usability tests performed on PointLoad.

The most significant changes are in the expansion of the Experiment Hypothesis Space, which becomes evident in the greater simulation capabilities of iBeam. In addition, the explanatory interface was not implemented as it was found that providing narrative information during simulations was distractive and of little interest to users.

Of the three assessment exercises, an expanded version of the Build Exercise was implemented in iBeam. This exercise was selected as it became apparent that this mode of assessment provided the best opportunities for interaction and provided the possibility of providing immediate and diagnostic feedback. The nature and form of feedback was changed from somewhat extensive text based explanations to controlled simulation output and short hints.

To address the programming issues, iBeam implements the same methodology and benefits from the basic architecture and technical solutions utilized in Pointload. The expanded capabilities of iBeam, however, demanded significant enhancements from the basic operational modules. Assessment and feedback strategies were much more challenging to implement given iBeam's expanded domain. In this respect, the adopted Artificial Intelligence strategy proved invaluable as it would have been very difficult to implement the necessary functionalities otherwise.

8.1 EXPERIMENT HYPOTHESIS SPACE

In contrast to PointLoad, iBeam's scope is quite extensive. This later application can handle multiple spans, and multiple and different types of loads. In addition, in simulation mode, the bending stiffness and length of each span can be independently adjusted. Furthermore, in response to input from the usability tests, quantitative relational input and output information is given to the user in addition to the qualitative relational graphic based information offered in PointLoad (Figure 57).

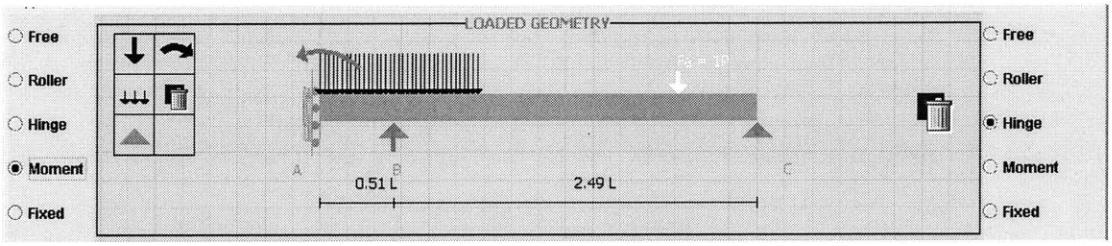


Figure 57 iBeam: Experiment Hypothesis Space

8.2 THE GRAPHICAL USER INTERFACE

iBeam's main graphical user interface consists of a single display frame divided into two sections: the *Task Window* and the *Control Window*. The Task Window placed on the lower half of the GUI, consists of an explanatory text area flanked by two columns of menu buttons located along the window margins. These buttons allow the user to access and update the student model, navigate between simulation and assessment modes and support the interactivity of these two tasks. While in simulation mode, the explanatory text area displays information about the structural model and interaction process. During assessment, the text area displays feedback and post evaluation information (Figure 58).

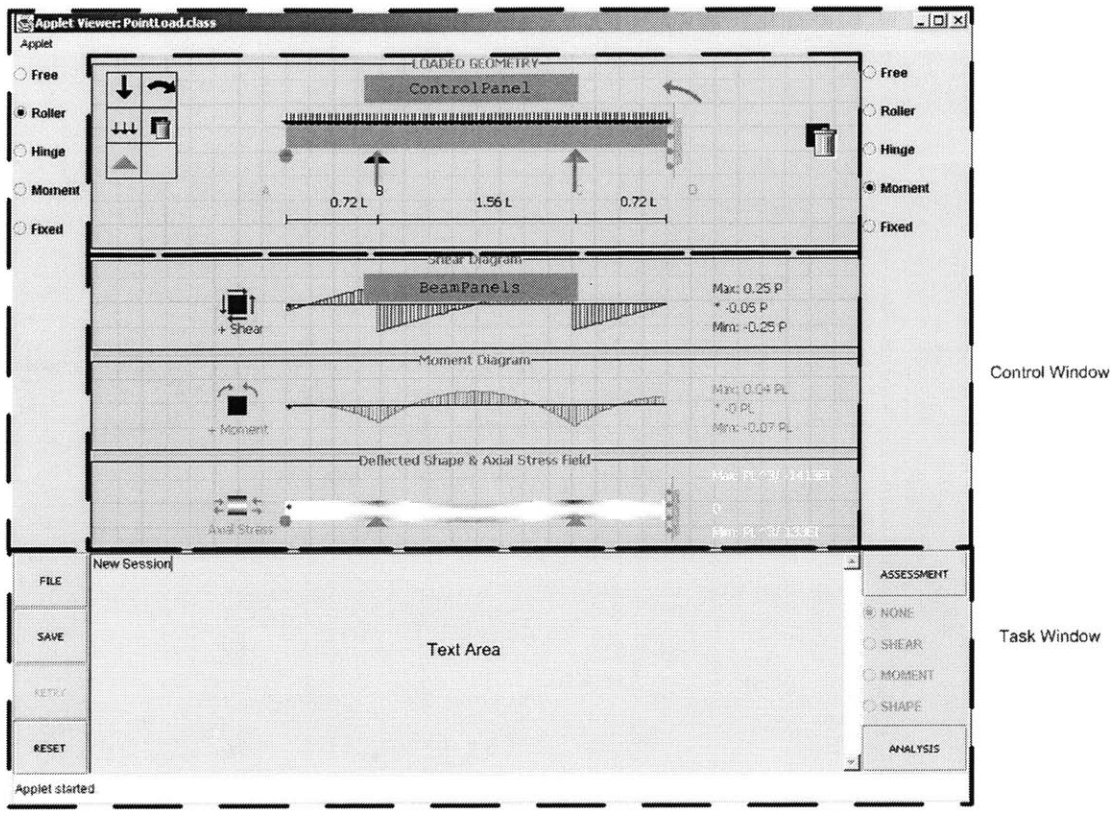


Figure 58 iBeam: Graphical User Interface in Simulation Mode

The Control Window, located above the Task Window, is configured according to the different operating modes supported by iBeam. It is, however, organized such that control buttons are always located along the right and left margins while the center portion corresponds to four graphical display panels. The topmost display corresponds to the Control panel, which consists of a `ControlPanel` pane that depicts the structural and loading configuration of the beam and facilitates the modification of the model through a menu of built in drag and drop control buttons.

The shear diagram, moment diagram and stress deformation diagrams are arranged bellow the Control Panel. These diagrams consist of instances of `BeamPanel` panes and provide graphical and numerical information relevant to the structural response of the beam.

As it was the objective in PointLoad, ease of use and minimum distraction are the principles that guide the design of iBeam's graphical user interface. Access to the features of the system is immediate and an effort has been made to minimize control features without limiting the capabilities of the environment. This approach proved successful in the development of PointLoad.

8.2.1 THE SIMULATION

As in PointLoad, iBeam's simulator is central to the learning environment. Its primary function is to interactively model and illustrate the structural response of a continuous multi-span beam. The main objective of the simulation is to engage the student in explorative action. To facilitate this engagement, the definition of the beam's structural and loading configurations is designed to require minimal effort and the structural response to changes made to the model is instantaneously computed and illustrated. The structural response is illustrated through visual representations of the free body diagram, the shear diagram, the moment diagram and a combined diagram of the deflected shape and axial stress contour. These diagrams are arranged into a column of four display panels placed at the center of the Control Window. This arrangement aims to facilitate the recognition of the functional relationships between the input parameters and the response patterns and between the response patterns themselves. The topmost panel corresponds to the configuration of the loaded beam. The intermediate and end support reactions are also shown on this panel. The panels bellow display the shear, moment and deformation diagrams. In addition, the deformation diagram depicts the axial stress distribution along the beam; red is used to indicate compression and blue for tension. The color intensity indicates the relative stress magnitudes (Figure 59).

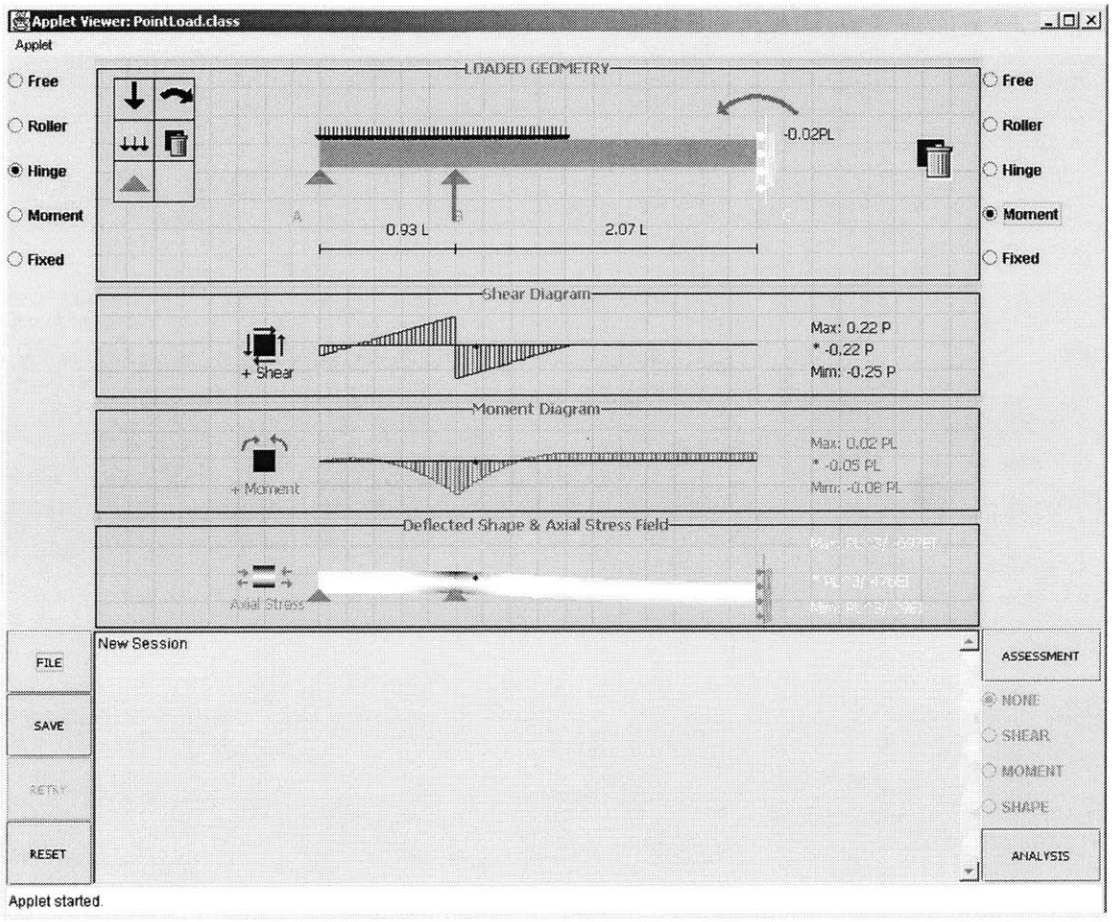


Figure 59 iBeam Screen in Simulation Mode

Developing meaningful iconic and symbolic representations of structural behavior, requires discovering typical response patterns and establishing the links between input parameters and the nature of the response. The comparison and contrast of simulation output can greatly foster this process. As a result, the visual displays are richly detailed and the interactive response of the simulator is on real time.

The information provided by the response diagrams consists of both a graphical illustrations and quantitative relational information. To avoid cluttering of the `ControlPanel` display, the magnitudes of the loads and reactions are displayed only when the mouse cursor is directly on top of a load or a support. The `BeamPanel` displays show the maximum and minimum values of the response

measure associated with the diagram each illustrates. In addition, by clicking on a coordinate of any response diagram, `BeamPanels` provide the value of the corresponding response measure at that point.

The beam configuration and response-pattern displays are drawn on a common horizontal scale. The vertical scale of each diagram is automatically determined according to the size of the panel and the maximum and minimum values of the diagram being rendered. This scaling is necessary since the magnitudes of the response measures vary widely according to the structural and loading configuration and the boundary conditions.

THE INTERACTION

Whenever the learner accesses the simulator, the beam is deployed with a default configuration. The user can then proceed to modify the input parameters until the desired configuration is obtained. The simulation responds to changes of the support boundary conditions at each end of the beam, to modifications in the layout of the intermediate supports and to alterations made to the loadings.

Changes to the end supports are made by clicking on the desired boundary condition on a set of radio buttons located on the GUI margins next to each of the beam ends. Five support-type choices are available: Free, Roller, Hinge, Moment and Fixed. Other modifications are accomplished through the `ControlPanel` (Figure 60). The intermediate support layout can be modified by adding, deleting or moving supports. To add an intermediate support, the user clicks on and drags the icon of a hinge support drawn on a tool menu located within the ControlPanel display. Once the icon is on the desired position, the mouse is released and the support has been added. To move or to remove an intermediate support, the user has to click on the desired support and drag it to the new position or to a "recycling" container located to the right of the ControlPanel display.

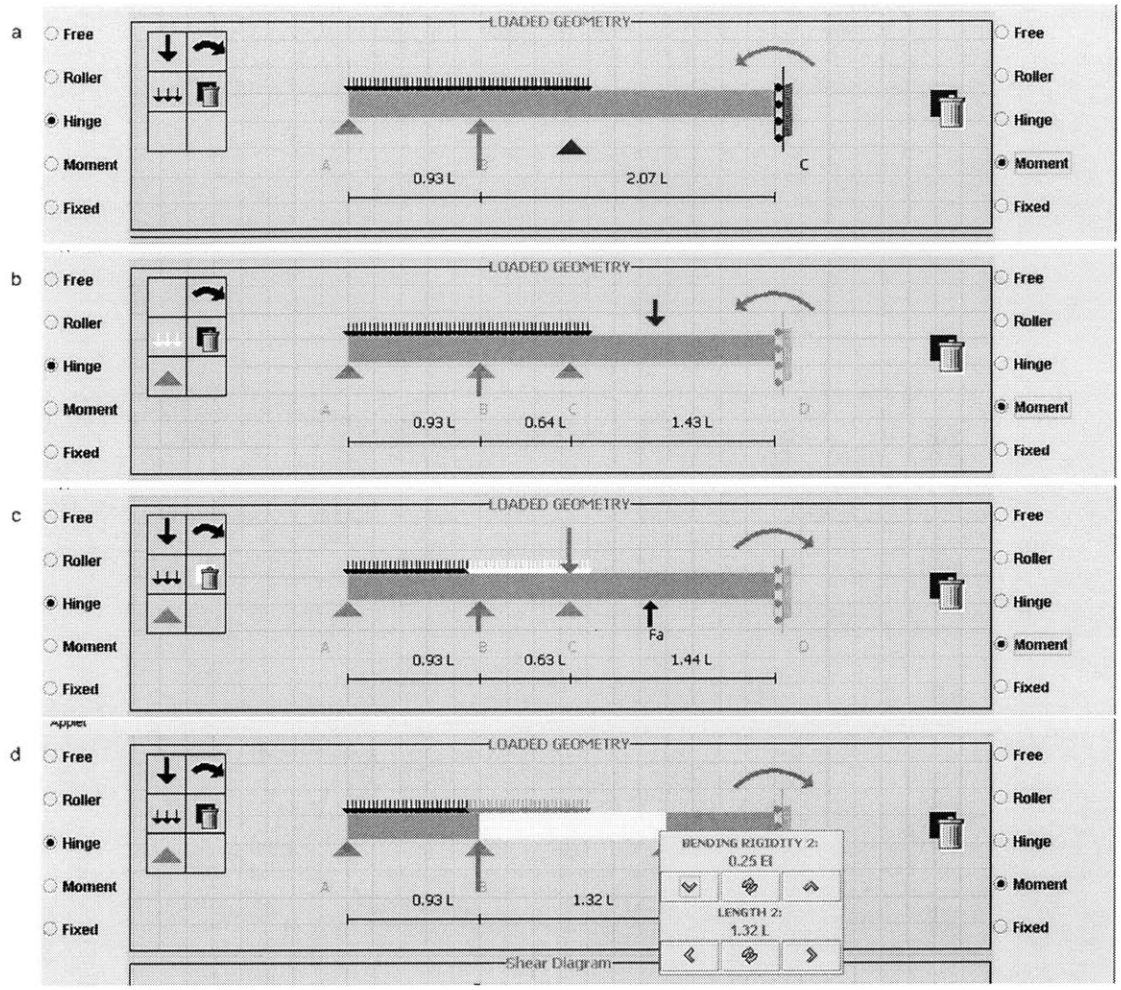


Figure 60 Control Panel : a. Intermediate support moving into palce. b. Concentrated load being applied. c. Distributed loads being removed (white). d. Center span being modified through pop-up menu.

Similar procedures are employed to add, move or remove concentrated forces and concentrated moments. The icon for a force is an arrow whose length represents the load magnitude. The standard value by which a load is added is equal to P. The icon for a moment is a curved arrow whose sweep angle represents the magnitude of the torque. The standard value by which a moment is added is equal to P*L. Whenever the position of two loads (force or moment) is close within a tolerance, they merge into one load whose magnitude is equal to their sum. The sense of a load can be reversed by double clicking on it.

Distributed loads are added by clicking the multiple-arrow icon on the ControlPanel display tool menu. The load can then be "spread" by clicking and dragging the mouse cursor over the desired beam interval. Performing this operation over a previously loaded interval, results in a distributed load whose value is equal to the sum. The distributed loading can be removed by clicking on the small trash container icon on the ControlPanel tool menu and subsequently dragging the mouse cursor over the distributed load arrows. The loads selected will turn white. By clicking on the trash container once more, the selected loads are deleted.

The length and bending rigidity of each span along the beam can be modified through a pop-up menu that is activated by right clicking over the desired member. Two arrow buttons facilitate increasing or decreasing the bending rigidity which may vary from EI/4 to 2*EI. A center allows resetting the span length to the default value, which is equal to EI. Similarly, two arrow buttons facilitate increasing or decreasing the span lengths. There is not a predefined upper bound. Supports, however, merge when close within a distance. As a result, the shortest span length is limited to be at least as large as this tolerance. The pop-up menu allows the user to reset the length of a span to the default length of L.

8.2.2 THE VIRTUAL TUTOR

In order to elicit reflective action, iBeam depends on a strategy that combines qualitative problem solving with evaluation and feedback. These tutoring actions are implemented by the Virtual Tutor through an interactive assessment exercise whose level of complexity and extent progressively adapt to the learner's level of knowledge. The problem presentation strategy consists of offering a solution and asking the student to recreate the corresponding structural and loading configuration. Tutoring guidance is given interactively in the form graphics based and short textual feedback

The pedagogical strategy used in problem presentation consists of gradually increasing the scope and complexity of the exercise, thus allowing the student to progressively discover the hypothesis of the domain. In addition, the interactive nature of the exercise lets the learner engage in exploratory action when defining a solution.

In order to accomplish these tasks, the Virtual Tutor needs to interpret the learner's level of knowledge, generate and propose exercise problems, assess the response, select and implement feedback strategies and control the interaction while in assessment mode. These actions are achieved primarily by two modules, the Student Model and the Assessment/Feedback module.

STUDENT MODEL

The primary purpose of the student model is to furnish the Virtual Tutor with a reference basis to guide and motivate the student during assessment. To accomplish this task, the user interactions on each trial are profiled in a record which consists of a Java object file of the class `StudentRecord`. A user who whishes to engage in problem solving activities, must first create a personal `StudentRecord`. This file is accessed and read by the Virtual Tutor to determine the level and kind of assessment exercises the student should get. During assessment, the `StudentRecord` is modified. This password protected file can then be stored and repeatedly accessed (Figure 61). The information accumulated in this file is used by the Virtual Tutor as a measure of the student's ability to identify and understand the knowledge that governs the structural behavior of a generalized beam interval. This section presents the structure of the `StudentRecord` and its subclasses. The way in which this file is used will become apparent when the Assessment/feedback module is described.

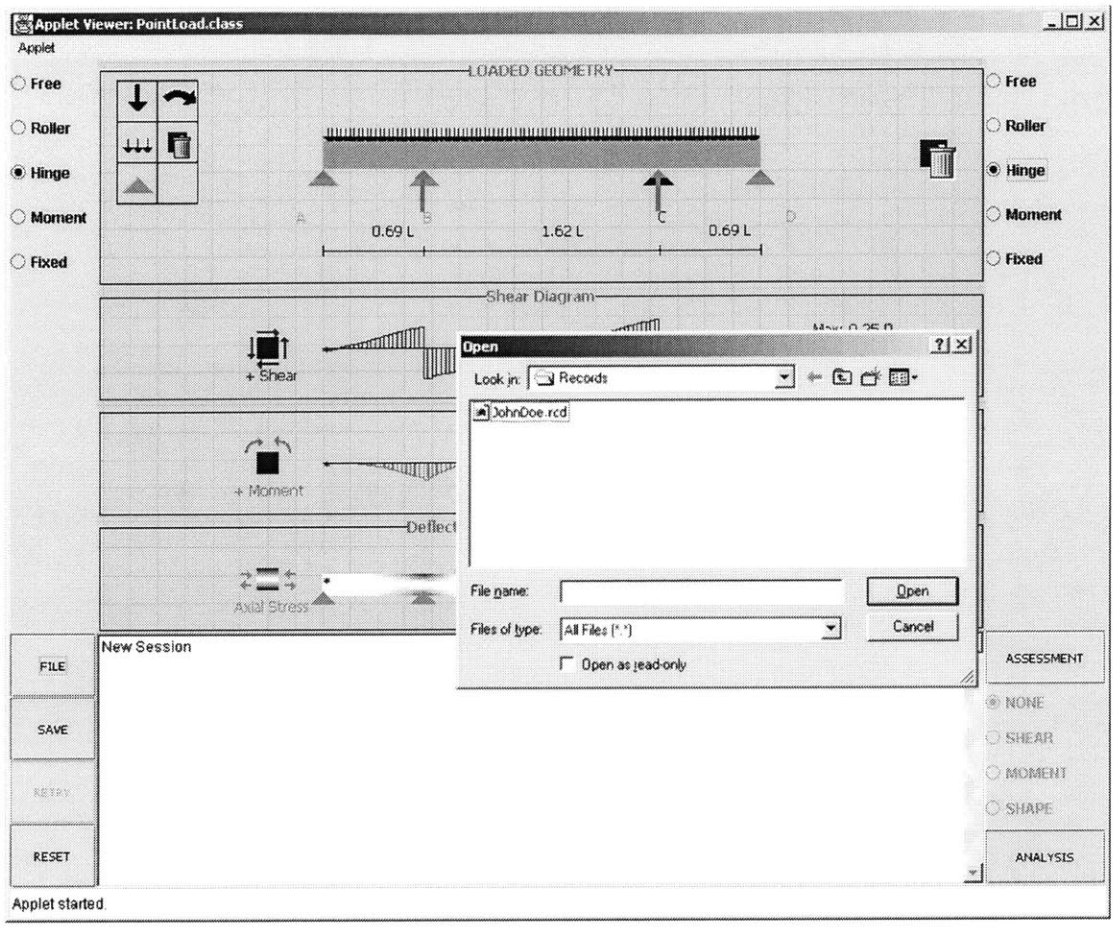


Figure 61 iBeam: Student Record File Screen

`StudentRecord` contains personal data such as the student's name and password, and performance information on the assessment problems implemented by iBeam. This information is contained in `Exercise` (Figure 62).

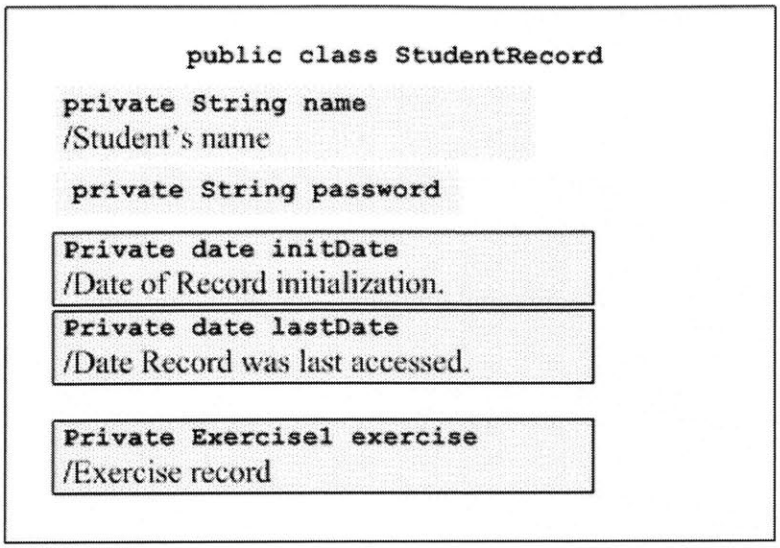


Figure 62 `StudentRecord` Class Structure

`Exercise1` consists of a Java `Vector` of `TrackingRecord` objects. `TrackingRecords` are objects that monitor the student's answers on the each of the assessment trials and essentially consist of stripped down versions of an `interval` object. `TrackingRecord` contains two `BCRecord` objects and two `CharacterRecord` objects. The `BCRecord` objects are associated with the left and right and boundary conditions, while the `CharacterRecord` objects are for the shear and moment character (Figure 63).

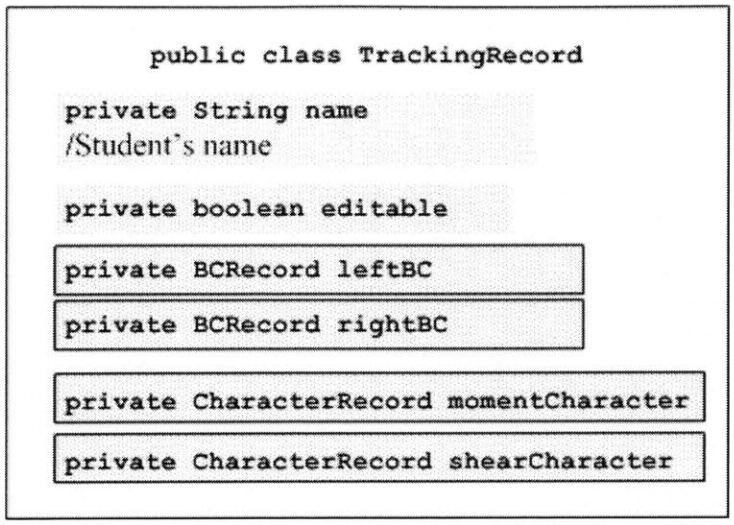


Figure 63 `TrackingRecord` Class Structure

`BCRecord` has a `String` descriptor for the boundary condition and a `Vector` of `Boolean` values. This `Vector` keeps a record of the correct and incorrect attempts made by the student at identifying the corresponding `BCRecord` boundary condition (Figure 64).

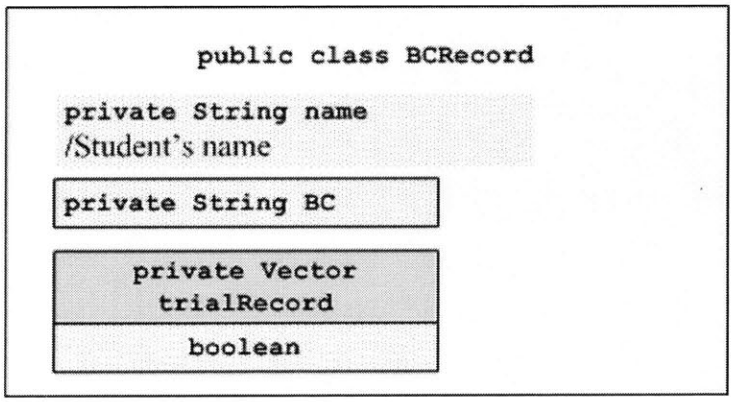


Figure 64 `BCRecord` Class Structure

Similarly `CharacterRecord` has a `String` descriptor for the interval shape character and `Vector` of `Boolean` values to keep a record of correct and incorrect attempts (Figure 65).

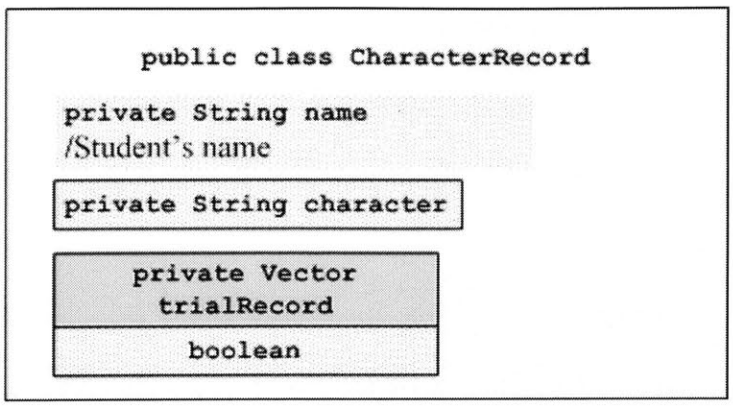


Figure 65 `CharacterRecord` Class Structure

ASSESSMENT/FEEDBACK MODULE

The Assessment/Feedback module performs three essential tasks, problem generation, evaluation and feedback. In order to perform these tasks, this module first reads the information contained in the `StudentRecord` file. This information is used to generate a problem that is presented to the student. This process consists of creating two `Input` objects, `MainInput` and `RespInput`, whose level of complexity is calibrated by the information read in the `StudentRecord` file. The structural response for the `MainInput` model is then calculated by the Finite Element Module and presented to the student on the GUI display panels while the structural configuration of the `RespInput` model is presented in the `ControlPanel` display pane. The student is then asked to interact with the input parameters of this model and establish the structural configuration that corresponds to the response diagram shown (Figure 66). Then, reasoning using qualitative domain knowledge, each of the student's interactions is processed and feedback content is selected and presented to the student.

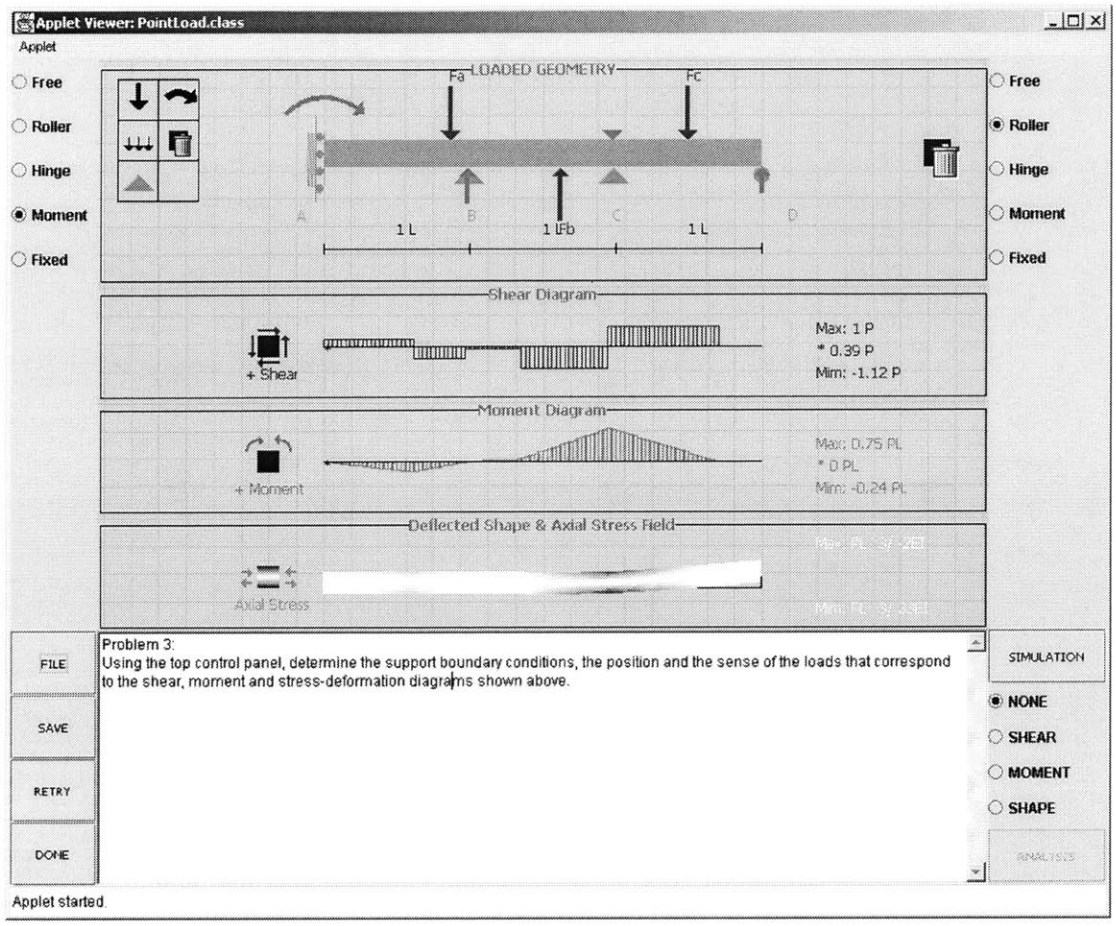


Figure 66 iBeam Screen Entering Assessment

Similarly to PointLoad, iBeam's Assessment/Feedback reasoning module consists of a combination of Java based procedures and JESS rule based inference mechanisms. These inference mechanisms consist of sets of related rules whose constraints continuously match the contents of `MainInput` against the contents of the student response stored in `RespInput`. The rule actions, when triggered, update the student record, generate and provide guiding advice and control the interaction by editing attributes of java objects of the general programming infrastructure. The underlying strategy that supports these inference mechanisms consists of forward and backward chaining. The programming of the rules is performed in the JESS programming language and are contained a script file.

The Assessment/Feedback Java procedures are responsible for creating and updating a fact data base against which the constraints of rules of the inference system are matched. Two methods of the static Java class `Evaluate` are involved in the generation and the update of this database, Assess3 and reloadAssess3. Assess3 creates the database with information related to the assessment exercise. While reloadAssess3 continuously updates the fact in the database as the student interacts within the assessment environment.

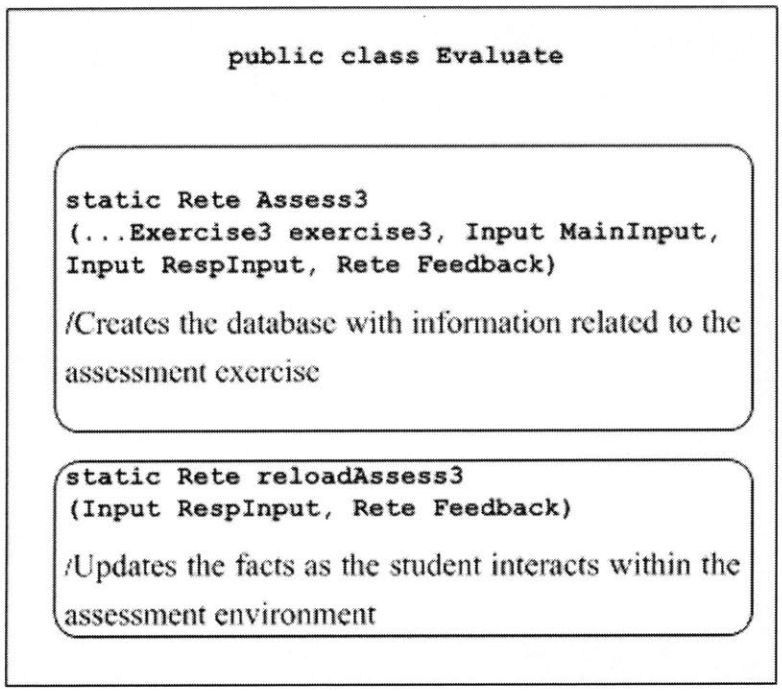


Figure 67 Methods of the `Evaluate` Class

8.2.3 THE EXERCISE

The assessment exercise implemented in iBeam consists of an expanded version of the Build exercise implemented in PointLoad and is designed to reinforce the reflective process of correlating patterns of structural response to their appropriate input parameters. In addition, this kind of problems engage in a rule discovery process by requiring from the student to actively build an answer, thus encouraging a

constructivist approach to learning. In this strategy, hints are provided instantly as the student selects the response parameters. As a result feedback becomes a guiding mechanism rather than a post evaluation advisor.

As mentioned in previous section 8.1, the assessment strategy consists of displaying a dynamically generated set of consistent response diagrams and then asking the student for the corresponding boundary conditions. The same graphical user interface layout used in simulation mode is recreated for this exercise. The controls used to manipulate the simulation are also used to build the exercise answer. As the student selects the input parameters for the answer, this information is instantly relayed to the reasoning agent, which then the compares characteristics of the selected input item with the characteristics of the corresponding response input. If there is a match the systems accept the partial answer. Otherwise, the reasoning agent identifies the nature of the mismatch and provides textual and graphical instantaneous feedback. For instance, if an incorrect load position is selected, the load arrow on the main display panel flashes red, hinting the student about the mistake before she/he commits to an incorrect answer. In addition a brief explanation is given in the text panel (Figure 68). The purpose of this action is to introduce conflict and insist on its resolution (flashing feature). If the student keeps having difficulties with the same parameter, the reasoning agent shows the student a feedback frame that displays the structural response of the proposed beam configuration and hints as to the nature of the mistake (Figure 69). The student can then compare and contrast these displays against the response of the main problem. As soon as the student interacts with any of the input parameters, the feedback frame disappears. This action is intended to guide the student along the right path but not disclose the answer. The strategy is to help the student build her own answer. If the problem persists, the exercise complexity is reduced by fixing the parameter to a correct state or eliminating it if it is not possible to find a match. The learner is then encouraged to proceed with the other parameters. This process continues until the student presses the DONE button

or until the student arrives at the correct answer either by choosing the correct input parameters or by triggering corrective intervention of the reasoning agent. At this point the environment shows the student the correct response and allows further simulation interaction with the model. Rather than being a post evaluation advisor, this type of response turns feedback into a guiding mechanism that helps build an answer.

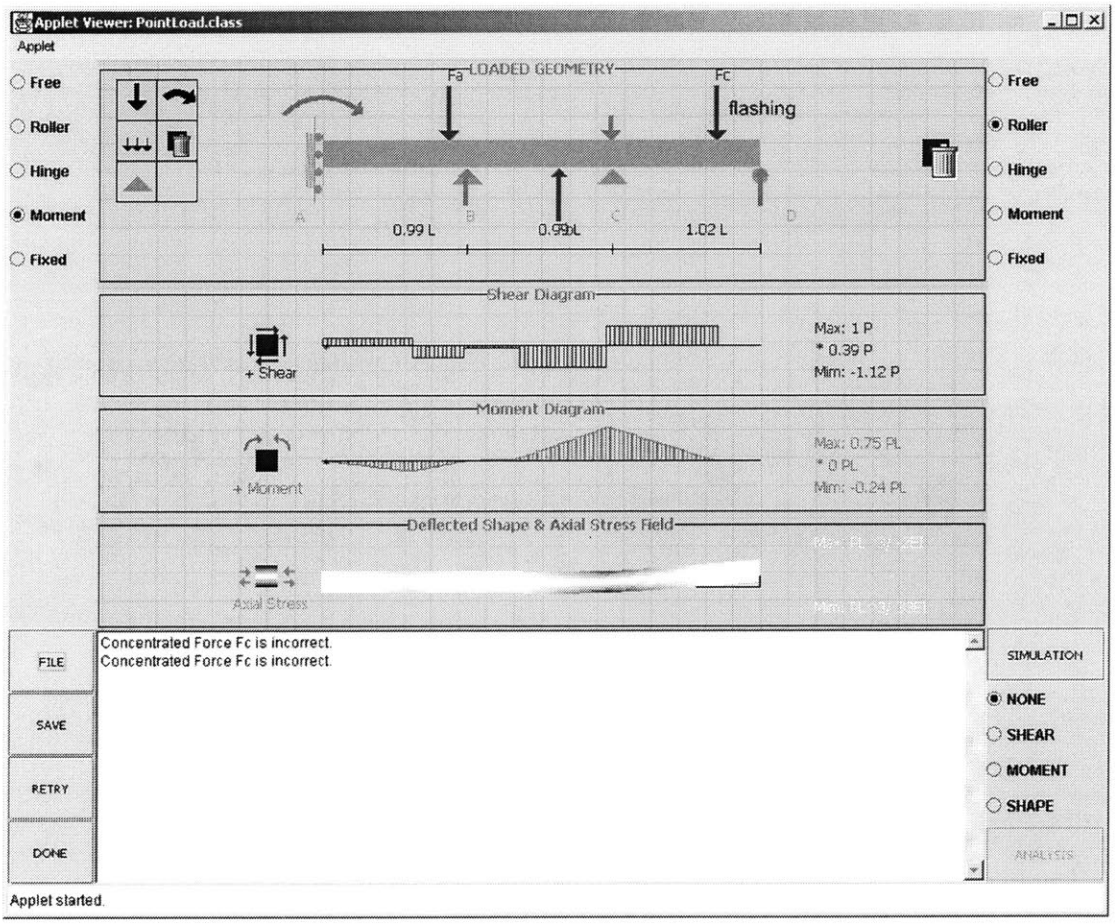


Figure 68 iBeam Screen Showing Immediate Feedback.

At any time during the solution process, the student may choose one kind of response diagram to see the structural response her proposed configuration (Figure 70). Unlike the feedback panel, this diagram is available through for the entire problem. However, once a diagram is chosen for display, the student can not access any other

diagram during that particular exercise. Only partial credit is given when the student chooses this kind of help. This strategy is intended to reduce complexity and provide additional guiding references to the student.

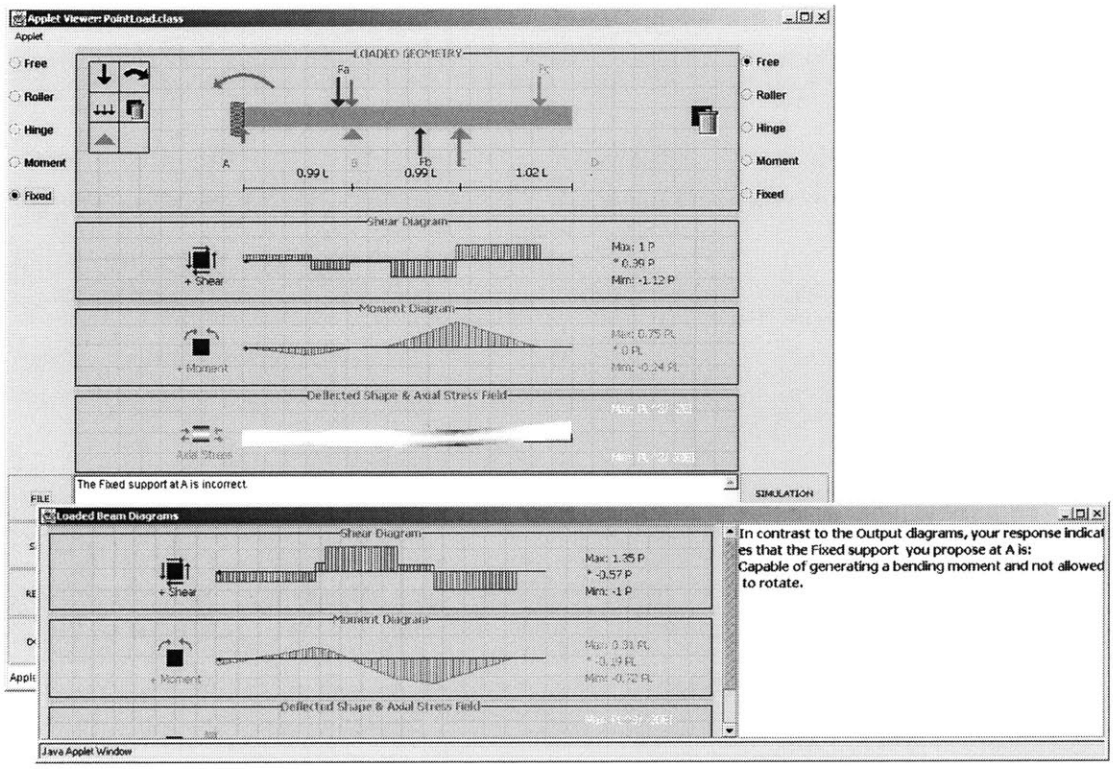


Figure 69 iBeam Screen Showing Feedback Panel.

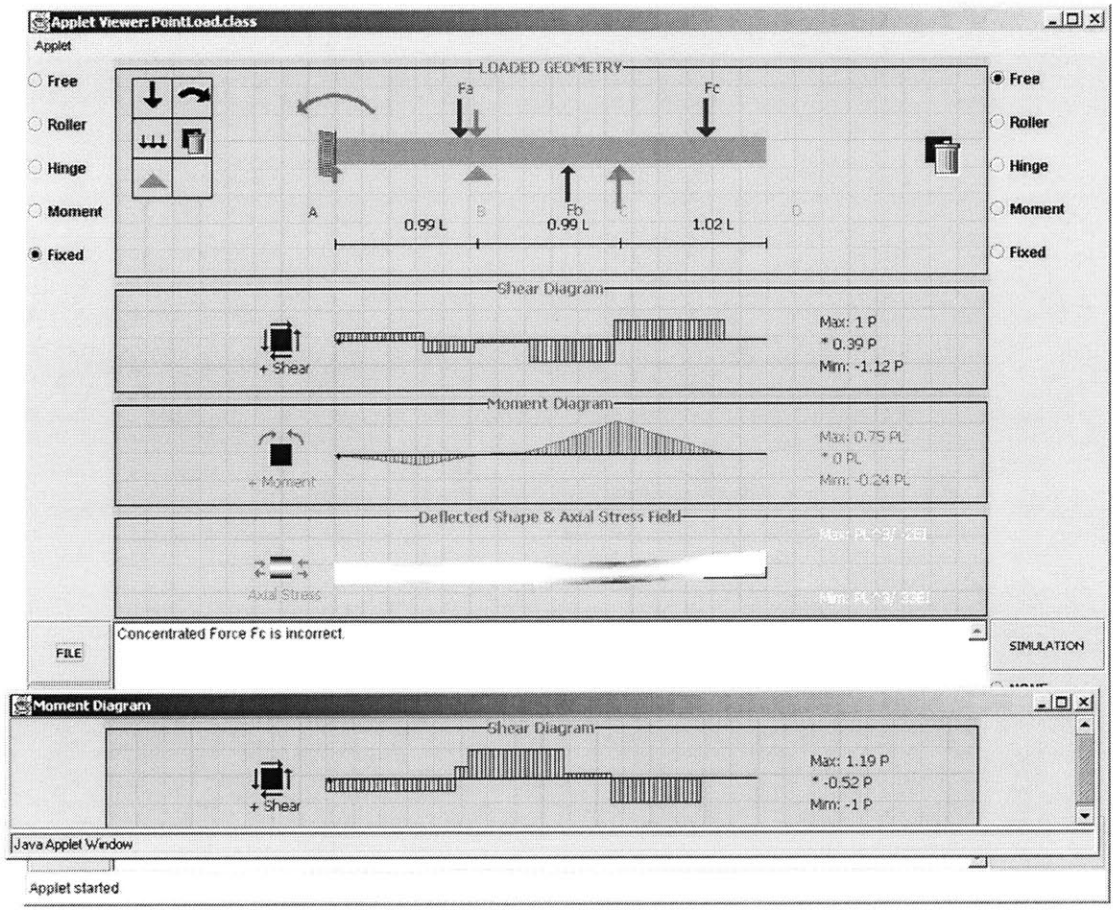


Figure 70 iBeam Screen Showing Help Shear Panel.

When the student finishes, the reasoning agent first identifies if there are matching interval boundaries between the main problem and the proposed solution. If this is the case, for each matched interval a `TrackingRecord` object with similar interval characteristics to the main problem is retrieved form the `StudentRecord` file. If this object cannot be found, one is created. Then if any of attributes of the answer interval correspond to those in the main problem, a true value is added to the corresponding record Vectors, a false value is added otherwise. If any of the interval boundary conditions was given by feedback intervention, the interval is not considered for update in the `StudentRecord`.. The exercise is completed when all the meaningful interval configuration are correctly matched and held by `StudentRecord`

PROBLEM SELECTION

The problems presented to the student vary according to the content of the `StudentRecord`. The first level corresponds to single span, single load exercises. When the record shows that the student has correctly identified interval configurations that include a concentrated moment and concentrated force boundary conditions, second level exercises are offered. This level consists of single span multiple load exercises. When intervals in which both boundary conditions include all three possible combinations of concentrated force and moment have been correctly identified, level three is reached. At this final level the multiple span exercises are offered. At this level combined boundary conditions are allowed.

In all instances, the beam in the question exercise and in the initial setup of the `ControlPanel` display have the same overall length and cannot be altered while solving the problem. In addition, the span pop-up menu are deactivated so changes to the bending rigidity values are not possible. These limitations are justified since changes in length and in cross sectional properties would lead to confusion and would foster quantitative rather than qualitative reasoning.

8.3 OBSERVATIONS

Initial reactions to iBeam indicate that the extended simulation capabilities make this environment more appealing and relevant than PointLoad. In addition, the quantitative relational information (available during simulation and during assessment) was appreciated as it gives additional references which are necessary to qualify the structural response of the more complex systems handled by iBeam. Most importantly, however, the assessment exercise was considered challenging while being "fun". This is precisely what motivates learning and what has been an objective of the research effort.

CHAPTER 9. SUMMARY, CONCLUSIONS AND FURTHER WORK

Computers have not only revolutionized the way engineers do their work. They have also changed the nature of structural engineering. Procedural tasks and skills that were essential in the not so distant past have now become obsolete whereas conceptual knowledge that was once viewed as helpful has now become the foundation of structural engineering work. Structural engineering is now measured by its ability to propose problem-solving novel structural configurations and not by its rote efficiency. These changes demand that educators review the nature of structural engineering education.

It is our view that computers can provide the answers. However, it is essential to first formulate the questions. The work described in this thesis has put forth some of the questions and has provided a few of the answers.

This work first identifies that it was needed to develop the cognitive framework for structural engineering knowledge so as to formally characterize the nature of structural behavior understanding as conceptual knowledge that is acquired by reflecting upon information gathered from direct observation and from theoretical principles. The epistemological study (centered on conceptual understanding) that followed then revealed that interactive computer simulations combined with qualitative problem presentation, assessment, and feedback would support the transformations needed to engage in these reflective processes. The Tutorial Cycle then facilitated the compilation of these principles into a methodology that provided the basis for the implementation of an experimental Experiential Learning Environment to support the understanding of the structural behavior of beams.

The development and implementation of this application led to the conclusion that, while interactive simulations facilitate exploration of a domain, they do not foster reflective action unless additional tutoring support is provided. It was also found that qualitative problem solving was most effective when it allowed the learner to interactively construct a solution rather that when the student had to readily provide an answer. Finally, it was learned that rather than trying to provide narrative explanations emulating human interaction, in a virtual setting, feedback should capitalize on the unique capacity that computers have to rapidly simulate and illustrate structural response. This is the capacity that has greatly benefited structural engineering practice and, we believe, is the one of the keys to effective learning environments. The other ingredient consists of identifying pedagogical strategies that lead to reflective action through epistemic conflict. Although we chose to use a scheme based on qualitative problem presentation, other scheme may be equally appropriate in other domains.

The implementation of the methodology revealed that the development of an Exploratory Tutoring Learning Environment demands resolving three main technological issues: interactive simulations, engaging user interfaces and the symbolic representation and manipulation of conceptual and heuristic domain knowledge. In most cases the first two issues can be resolved through algorithmic deterministic programming in high-level languages such as Java or C++. These languages are adequately fast and support utilities that have robust of numerical, graphical and interfacing capabilities. In addition, there is ample information on the development of engaging interfaces (Chabay & Sherwood, 1992). The approach to knowledge manipulation, on the other hand, requires more careful deliberation, and, in most instances it will be necessary to implement an Artificial Intelligence strategy. Most of the theory of structural engineering can be characterized by formulae and rules. These rules often describe very complex associations among concepts and procedures. As result, a knowledge based system (KBS) based on production rules

would be a suitable choice for knowledge representation and manipulation. The fragility and low performance speeds traditionally associated with large KBS should not be a major concern since the domain is limited by the topic of the lesson objective. This condition minimizes the number of rules necessary to represent the domain and controls the size of the facts in the knowledge base. In addition, *intelligent* tasks can be subdivided among small agents, further reducing the processing size and complexity.

Future work should seek to identify ways in which to better integrate the various components of a complete learning environment: classroom based instruction, reading assignments and computer based learning support. Student performance on these computer environments could become a valuable source of timely feedback information that could point to strengths and/or deficiencies in teaching learning styles or lecture material. Further research on ways of administrating the student record could, in addition, turn these learning environments into better means of evaluating a student than traditional tests.

Finally, although simulations based of the FEM can readily illustrate the structural response of most systems, developing symbolic knowledge representations of most topics is still a challenge and an essential feature of an effective computer based learning environment. As a result, implementing the methodology for Experiential Learning Environments in other domains should still be viewed as a valuable research contribution.

REFERENCES

Beck, J., Stern, M., & Haugsjaa, E. (1996). Applications of AI in Education. In: *ACM Crossroads.* 3(1):http//www.acm.org/crossroads/xrds3—1/aied.html.

Billet, S., & Rose, R. (1999). Securing Conceptual Development in Workplaces. In: P. Murphy (ed.) *Learners, Learning & Assessmnet.* pp. 329-344. Thousand Oaks, CA: Sage Publications Inc.

Boder, A., & Cavallo, D. (1991). An Epistemological Approach to Intelligent Tutoring Systems. In: M. Yasdani & R. Lawler (eds.) *Artifitial Intelligence in Education: Principles and Case Studies.* pp. 203-218. Norwood, NJ: Ablex

Chabay, R. W., & Sherwood, B. A. (1992). A Practical Guide for the Creation of Educational Software. In: J. Larkin, R. W. Chabay (eds.) *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared issues and complementary approaches.* pp. 151-186. Hillsdale, NJ: Lawrence Erlbaum Assoc.

Cumming, G. (1993). A perspective on learning for intelligent educational systems. In: *Journal of Computer Assisted Learning.* 9(4), pp. 229-238.

Dalal, N. P., & Kasper, G. M. (1994) The design of joint cognitive systems: the effect of cognitive coupling on performance. In: *International Journal of Human-Computer Studies.* 40, pp. 677-702.

Duffy, T. M., & Jonassen, D. H. (1991). Constructivism: New Implications for Instructional Technology? *Educational Technology,* 31(5), 7-12.

Elsom-Cook, M. T. (1993). Environment Design and Teaching Intervention. In: D. Towne, T. de Jong & H. Spada (eds.), *Simulation-Based Experiential Learning,* pp. 165-176. Berlin: Springer-Verlag.

Forman, G., & Pufall, P. B. (1988). Constructivism in the Computer Age: A Reconstructive Epiloge. In: G. Forman & P. B. Puffal (eds.), *Constructivism in the* Computer Age. Hillsdale, NJ: Erlbaum.

Friedman-Hill, E. (2001). Jess, The Java Expert System Shell. In: *Distributed Computing Systems, Sandia National Laboratories.* Livermore, CA: http://herzberg.ca.sandia.gov/jess.

Gere, J. (2001). *Mechanics of Materials.* Pacific Grove, CA: Brooks/Cole.

Jackson, P. (1999). *Introduction to Expert Systems.* Harlow, England: Addison-Wesley.

Jonassen, D. H. (1991) Objectivism versus constructivism: Do we need a new philosophical paradigm? In: *Educational Technology Research & Development.* 39, 5-14.

Kashihara, A., Matsumura, K., Hirashima, T., & Toyoda, J. (1994). A cognitive load application approach to tutoring. In: *Proceedings of the Fourth International Conference on User Modeling,* 163-168.

Kinshuk, Patel, A. (1996a). Artifitial Intelligence in Learning and Assessment – A Supportive Software for the Teaching of Accounting. In: S. Overmyer & J. Gornostaev (eds.) *Proceedings 6th East-West International Conference on Human-Computer Interaction.* ICSTI, Moscow, pp.1-11

Kinshuk, Patel, A. (1996b). Intelligent Tutoring Tools: Redesigning ITSs for Adequate Knowledge Transfer Emphasis. In: C. Lucas (ed.) *Proceedings of 1996 International Conference on Intelligent and Cognitive Systems.* IPM, Teheran, pp. 221-226

Kolb, D. (1984). *Experiential Learning.* Englewood Cliffs, NJ: Prentice-Hall.

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. In: *International J. of Artificial Intelligence in Education.* (1999), Vol. 10, pp. 98-129.

Njoo, M., & de Jong, T. (1993). Supporting Exploratory Learning by Offering Structured Overviews of Hypothesis. In: D. Towne, T. de Jong & H. Spada (eds.), *Simulation-Based Experiential Learning,* pp. 207-223. Berlin: Springer-Verlag.

Opwis, K. (1993). The Flexible Use of Multiple Domain Representations. In: D. Towne, T. de Jong & H. Spada (eds.), *Simulation-Based Experiential Learning,* pp. 77-89. Berlin: Springer-Verlag.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books.

Sellman, R. (1992). Gravitas, Unpublished PhD thesis. Open University.

Shepherdson, E. (2001). *Teaching concepts utilizing active learning computer environments.* MIT Thesis C.E. Ph.D.

Sleeman, D., & Brown, J. S. (1982). Introduction: Intelligent Tutoring Systems. In: D. Sleeman & J. S. Brown (eds.) *Intelligent Tutoring Systems.* pp. 1-11. New York: Academic Press.

Teodoro, V. D. (1993). A Model to Design Computer Exploratory Software for Science and Mathematics. In: D. Towne, T. de Jong & H. Spada (eds.), *Simulation-Based Experiential Learning,* pp. 177-189. Berlin: Springer-Verlag.

van Joolingen, W., & de Jong, T. (1993). Exploring a Domain with a Computer Simulation: Traversing Variable and Relation Space with the Help of a Hypothesis Scratchpad. In: D. Towne, T. de Jong & H. Spada (eds.), *Simulation-Based Experiential Learning,* pp. 191-206. Berlin: Springer-Verlag.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., & Evens, M. W. (1999). What Should the Tutor Do When the Student Cannot Answer a Question? *Proceedings of the Twelfth Florida Artificial Intelligence Symposium.* Orlando, FL.