# System Engineering based Design and Control of Teletraining Sessions in an Open Environment

Witold Jacak
Department of Software Engineering
Upper Austria Polytechnic University
A 4232 Hagenberg-Linz, Austria
jacak@fhs-hagenberg.ac.at

Andreas Wintersteiger
Institute for Systems Science
Johannes Kepler University Linz, Austria
aw@cast.uni-linz.ac.at

Jerzy Rozenblit
Department of Electrical and Computer Engineering
The University of Arizona
Tucson AZ 85721 USA
jr@ece.arizona.edu

## Abstract

*In this paper we present a method and a tool for modelling a teletraining session in heterogenus, distributed open environments. We propose a mathematical notion for the training process. Therefore we divide a whole training session into presentation units, define some relations on these units and develop a controller for running the session. Units consist of multimedia objects, such as text, graphics, video and audio, which have to be displayed with time and space synchronisation and coordination.*

*In a related project at the Department of Software Engineering at the Upper Austria Polytechnic University a tool for modelling and running such teletraining sessions was developed. It is based on the herein defined formalism and compiles well defined unit models into JAVA™ code, which can be executed by usual WEB–Browsers.*

## 1 Introduction

In recent years an explosion of multimedia applications has been seen in many domains such as education, training systems, office and business systems etc.
Many multimedia applications will be designed to run on heterogeneous workstations or to be interconnected to offer a multimedia service. Some of these can be used directly to preparation of different teleeducation courses or telepresentation workshops [5].
However multimedia incorporation proves not sufficient for education system implementation. Unlike the computer aided instruction and intelligent tutoring systems the contemporary education systems are to be developed as interactive learning environments.
It is also important that students or workshop participants could not only conduct and record but also communicate their work. The advantages of teleeducation in an open environment mainly concern the shared databases, the improvment of collaborative work, interdisciplinary links and individualized learning. Additionally, multimedia offers more objective and expressive presentation of knowledge and makes simulation based learning possible.
The production of large quantities of document-information, necessary for the expansion of services is a significant investment and it is important that this information remains available and is not lost because of incompabilities in the data structures supported by the applications [1].
Moreover such information should be easy available for non professional software engineer (teacher, manager etc.), which additionally support systems for easy creation of education sessions or workshop flow and its control. For the typical applications in the domain of teleeducation and telepresentation a number of requirements can be identified [5, 1]:

- Portability in multi-vendor environment
- Media information to be able to group several monomedia entities into a single container



0-8186-7889-5/97 $10.00 

- Structure of information - the real time interactivity, including easy acquisition of multimedia data can be ensured
- Easy , user friendly composition and synchronization in space and time
- Specification of links and use of non multimedia objects
- Generation of interaction with different user classes
- Reusability of data in other documents
- Ability to update and manipulate sets of data or object elements

For the purpose of reusing different component multimedia objects or scripts in different presentations, learning sessions or activations, a clear standardization specification is needed. Moreover, the design of distributed hypermedia applications needs presentation preparation tools, which allow easy creation of real-time performed learning sessions or final remote presentations.
From the above presented reasons we develop new methods and software for computer supported creation of interactive multimedia telepresentations of education session in an open and distributed environment.
We consider the composer of such telepresentations having most different multimedia objects like text, graphics, movies, videos and also nonhypermedia objects such as executable mathematical, simulation software or design tools.
We propose a way of unifying these different objects into one standardized format, based on MHEG–Standard *Multimedia and Hypermedia information coding Expert Group* [3, 2]. Thus we call these objects *dynamic multimedia documents.*
Additionally, the contents specification should be developed. Each object will be modelled on two layers of specification: The static layer describing the standard document attributes and the dynamic layer characterizing the object's dynamic properties with interaction dependencies.
The presentation's objects will be stored in a distributed database, which elements (multimedia objects) can be stored somewhere in the open environment. An intelligent navigation system allows selection and preview of these objects from all over the world. One such document becomes part of the presentation. The multimedia objects can be used as a part of different sessions or presentations. The presentation is modelled in form of an event depending graph, representing the general flow of the education session. The nodes in the presentation's graph represent the equivalence class of units with equivalent contents. The arcs of the graph are generated from the unit's precedence relation and conditions assign relation. Such the structure of the session is used to generate the event based control system for the session execution automatically [4].
The session architecture will be synthesized to a standard source code, which can be executed by intelligent network browsers like NetScape Navigator. The telepresentation session application is based on a client/server–tool for creating the architecture and it will take control over all clients. This application deals with space and time synchronization such that the distributed telepresentation environment can be seen as a virtual electronic classroom where students can work and communicate interactively [4].

## 2 Basic Notions

When beginning the synthesis of a system for teletraining design, one specifies the family of educational or training tasks which can be developed and next realized with this system.

### Training task specification

In this section we focus our concern on the issues involved in the education process. The *education process* or *training process* may include various linearly ordered educations sessions i.e. the education process is the linear sequence of sessions.

$$Edu\_P = (Ses_1, Ses_2, ..., Ses_n)$$

The basic components of each education session are *education units.* An education unit is a complete portion of the knowledge needed to present one learning or training item. The unit is characterized by the unchanged gross contents, which can be expressed with many different tools such as multimedia objects, on–line transmitted experiments, on–line control etc.
The presentation process is critically dependent on the education session representation. Here, we use a general description of an education session, formally specified as follows:
Education session $S_i$ is the weakly ordered set of education units and is represented by a three-tuple:

$$Ses_i = (U^i, \prec^i, \equiv^i) \quad (1)$$

where:
$U^i = \{u^i_k | k = 1, ..., L^i\}$ is the set of education units connected with $i$–th education session,

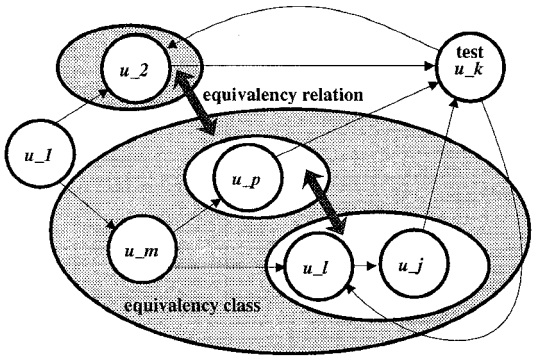


**Figure 1. Education Session**

$\prec^i \subset U^i \times U^i$ is the unit precedence relation, and $\equiv^i \subset 2^{U^i} \times 2^{U^i}$ is the unit equivalence relation.
The partial order represents a presentation precedence, i.e., $q \prec^i u$ means that the unit $q$ is to be completely presented before the unit $u$ can begin the presentation. Let $Q$ and $W$ be the subsets of $U^i$ i.e. $Q, W \subset U^i$. $Q \equiv^i W$ means that the sequence of units $q \in Q$ can be substituted by the sequence of units $u \in W$ during the presentation of the education session. The units $Q$ and $W$ have equivalent contents.
Especially, if $\{q\} \equiv W$ then the unit $q$ can be substituted by the sequence of units from the subset $W$.
An example of an education session is illustrated in Fig. 1. To present such an education session in an open environment the contents of session units should be implemented in form of different documents. The contents of every education unit $u_k^i$ is realized by the presentation of an adequate presentation unit.

## 3 Presentation Unit: Modelling and Control

The presentation unit is the multimedia based implementation of education unit contents. The composition of different multimedia documents, which present the contents of education unit creates the complex multimedia object. This object forms the *presentation unit* equivalent to the *education unit* and will be formally used as education unit representation in the education process.
Let $MM$ be a set of different multimedia objects such as text documents, figures, images, movies etc. To create the presentation unit $p\_u_k^i$ for a given education unit $u_k^i$ from the education session $Ses_i$ the composition function should be given. The composition function

$$C^i : U^i \to 2^{MM} \tag{2}$$

assigns the sets of simple multimedia objects $m \in MM$ to every education unit $u \in U^i$

$$C^i(u_k^i) = \{m_j | j \in J_k^i\} = M_k^i \subset MM \tag{3}$$

The structured set of multimedia objects and documents $M_k^i$ is called *a presentation unit* (p_Unit) of an education unit $u_k^i$ from $Ses_i$.
Then, the presentation unit is composed of different simple objects belonging to different multimedia object classes. Each one of such objects is described by two basic forms:
– the first one describes a static property of the multimedia object using the SGML formalism,
– the second one presents the object's controller needed to its properly presentation during the execution and transmission of the session.
The presentation unit as a composition of simple objects creates a so called complex multimedia object. The proper presentation of the unit's components needs a time and space coordination and synchronization. It causes a multilevel control system for p_Unit execution. This control system is realized on two levels:
– the execution level performs the presentation of each simple object component of the presentation unit,
– the coordination and synchronization level scheduling the presentation of unit's components according to the presentation scenario.

### Execution controller of simple multimedia object

The presentation of the given object $m$ from the set $MM$ is controlled by the following dynamical system [6, 7]:

$$Contr\_m = (S_m, X_m, Y_m, \delta_m^{ex}, \delta_m^{in}, \lambda_m, \tau_m) \tag{4}$$

where:
- $S_m = Space\_Pose \times Logic\_State$ is the state set of object $m$ and $Space\_Pose$ describes the location of the object on the screen,
- $X_m$ is the external (control signal) event set,
- $Y_m$ is the output (message) event set,
- $T$ is the own time base,
- $\delta_m^e : (S_m \times T) \times X_m \to S_m$ is the external state transition function,
- $\delta_m^i : S_m \to S_m$ is the internal state transition function,
- $\lambda_m : S_m \to Y_m$ is the output function,
- $\tau_m : S_m \to \mathbb{R}$ is the time_activity function.

The controller is in the form of discrete event based dynamical system DEVS, developed by Zeigler [7].

### State space of object $m$

The position component of the object state determines the location and zooming of the object on the screen during presentation. In the easy case

$$Space\_Pose = \{(x_{ref}, y_{ref}) \mid x_{min} \leq x_{ref} \leq x_{max} \wedge y_{min} \leq y_{ref} \leq y_{max}\} \quad (5)$$

and the pair $(x_{ref}, y_{ref})$ gives the screen position of a reference point of the object.
The second component of the state presents the current status of the object during its presentation on the screen.

$$Logic\_State = \{\text{Resetting, Idle, Preparing, Prepared, Starting, Running, Pausing, Terminating, Terminated}\} \quad (6)$$

These components are used to control the object's presentation. Some of this states are active, i.e. the object comes to the next state without additional external signal, and some are passive which need an external signal for transition.

### Time activation function

For every active state the life time should be given while for passive states the life time is equal to infinity. The activation function describes the life time for each state.
$\tau(\text{Idle}) = \infty$, $\tau(\text{Preparing}) = t_{prepar}$, $\tau(\text{Prepared}) = \infty$,
$\tau(\text{Starting}) = t_{start}$, $\tau(\text{Running}) = t_{execut}$,
$\tau(\text{Pausing}) = \infty$, $\tau(\text{Terminating}) = t_{terminat}$,
$\tau(\text{Terminated}) = \infty$, $\tau(\text{Resetting}) = t_{reset}$.

### State transition - the execution control

All possible state transitions are presented in Fig. 2. During the presentation, any object passes through a sequence of states. State transition occurs either due to the reaction upon the receipt of control signals - external events, or due to internal events. In each state , the object performs certain actions, and reacts only to specific control signals. Upon certain state transitions, messages are emitted to indicate the completion of the corresponding actions. The life cycle shows that the control signals are accepted only in states not later than the indicated one. Once an object arrives at the

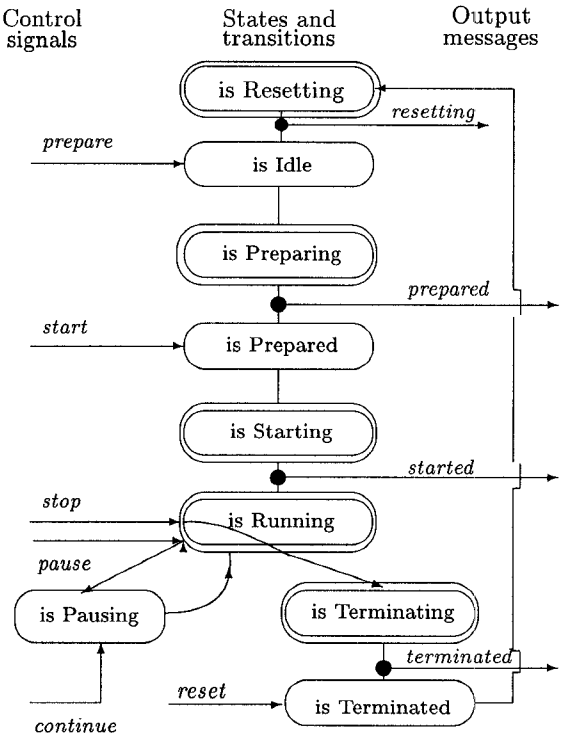


**Figure 2. The object's life cycle (state transition)**

state "Terminated", it cannot be presented anymore, as long as if it is not reset, either implicitly or explicitly by receiving the "reset" signal, which is accepted in any state other than "Idle". In addition , an object's states contains one further component "Pause", which is controlled by the signal pair "pause"/"continue". These signals are accepted at any time, but take effect only in the "Running" state.
The states "Preparing, Starting, Terminating, Resetting" allow adjustment of hard/software caused delays. Signals can be implemented as some kind of logical/boolean object attributes which are initialized to false and cause the object to send a message to all waiting ones when switching to true. Setting such an attribute (signal) once more when already true does not cause anything to happen. The "reset" signal sets it's value back to false. A "reset" command can imply, for example, that the following actions have to be performed:
first, the object state is set to "is Resetting";
second, any actions necessary for terminating the presentation are performed;

third, all signals are set to false; fourth, all attributes are re-initialized and updated respectively, depending on their meaning; and
finally , the object state is set to "Idle".
This model presents only standard states, signals and messages. Besides those signals shown in Fig. 2, the controller may accept further signals and emit additional messages. It can be extended easily on application defined states and signals. Application defined signals behave in the same way, but can be set and unset explicitly by the application. For example, iterating the object will stop the current iteration and start the next or previous one when receiving "next" or "previous" signals, respectively.

## Coordination and Synchronization of p_Unit Presentation

As we have previously mentioned, the presentation unit is a composition of the simple objects. In one special case, the p_Unit can contain only one object.
The presentations of each unit's component have to be coordinated and synchronized in time and space. For the synchronization of the component's execution, a higher level of control is introduced. This level will be called the *coordination and synchronization level.*
To present the contents of multimedia objects, a coordinator of the p_Unit realization must exist. That implies that a component object not coordinated by any p_Unit will not be presented. Generally, p_Unit should perform the coordination and synchronization of different objects presentation.
Let $C_i(u_k^i) = M_k$ be the set of multimedia objects assign to this p_Unit. Formally, a coordinator and synchronizator of the p_Unit for the given education unit $u_k^i$ is represented by the following structure:

$$\text{Syn p\_Unit}_k^i = (SU_k, IQ_k, AX_k, RB_k, Akt_k, Gen_k) \tag{7}$$

The components of the synchronizator p_Unit have the following meaning:

- $SU_k = X\{S_m | m \in M_k\}$ is the global state set of presentation. The vector $(s_1, s_2, ...s_M)$ describes the current status of the presented unit's components,
- $IQ_k = \bigcup\{Y_m | m \in M_k\}$ is the messages queue, generated by controllers of unit components in execution level,
- $AX_k$ is the authoring external event. The external event $x \in AX_k$ is used to initialize, activation or deactivation of particular unit's flow parts,
- $RB_k$ is the rule base; logical synchronization of p_Unit performance,
- $Akt_k : IQ_k \rightarrow SU_k$ is the actualization function which updates the global state register based on coming messages from the unit's components,
- $Gen_k = \{g_m(RB) : SU_k \times AX_k \rightarrow X_m | m \in M\}$, is the set of generation functions of control signal (external event) for each object's controller

To present the objects from the p_Unit it is necessary to produce a presentation scenario for this unit. The scenario describes the composition and scheduling of objects during the presentation flow. Generally, a scenario can be represented by a set of rules, which determine the running of a given object depending on the current global status of the unit.
The rule base contains such synchronization rules in form of

*IF conditions THEN actions.*

One rule determines execution of an object or object group depending on the current global state of the presentation. An example of such rule is presented below:

```
IF     Object_i is Running
   AND Object_j is Prepared
   AND Hardware for Object_j is Idle
   AND Authoring_Event is Button_Toggle
THEN   Control_Signal X_j is Start
   AND Hardware for  Object_j is Busy
   AND (Time(Object_j.started) + 20 Time-Units)
   AND Control_Signal X_j is Pause
```

The p_Unit structure is shown in Fig 3.

## Time synchronization engine of p_Unit flow

The p_Unit synchronization mechanism work in following steps:

1. The messages from $IQ$ are used by the function $Akt$ to change the global state. The function $Akt$ changes the register $SU$ of the current status of each object.
2. The new global state activates the rules form $RB$ which preconditions are satisfied in this state together with an external authoring event.
3. The active rules are used by the generator $Gen$ to generate the control signals - external events for objects controllers. These signals activate the actions of local controllers.

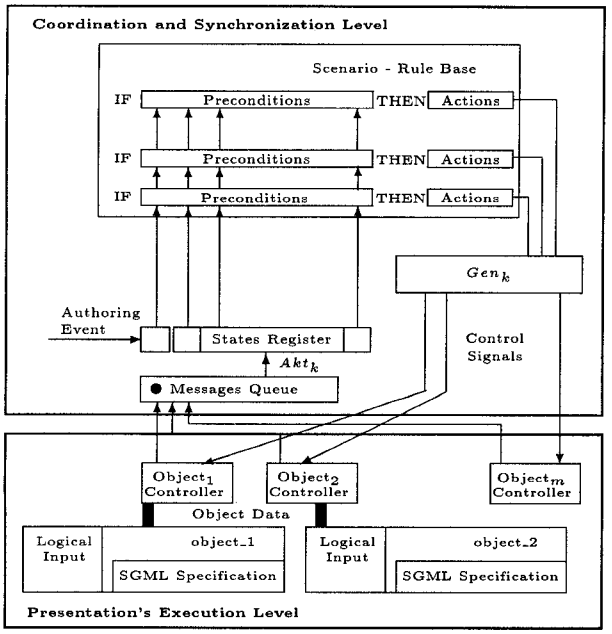


**Figure 3. Control structure of presentation unit**

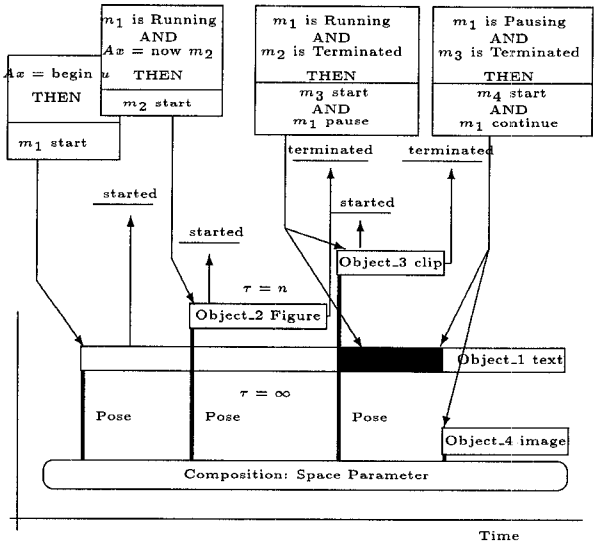


**Figure 4. Time synchronization of presentation unit execution**

4. The object's controller changes the status of the presented object, executes the object's state transition, and sends the message to $IQ$.

For an example of the control of the presentation flow refer to Fig. 4

# 4 CAD System for Education Session Development

The proposed CAD system allows the user to design and specify a multimedia telepresentation in a very comfortable way. The system compiles a generalized source code of all documents into JAVA language, a standard for internet applications. It envolves the use of control models, presented in the previous section, to generalize the presentation flow including interactivity.

## Graph based design of presentation session

The telepresentation design bases on semantic graphs. Each dragged object becomes a node of the graph, which will be meshed to other nodes by arcs defining the input/output–relationship. The whole graph then defines the session flow of the telemedia presentation. The nodes of the graph will be represented by a presentation unit controller and synchronizer.

## Node and arc specification based inference–engine for session control

The inference–engine compiles a complete discrete event specified model out of the graph, where each node, specifying a component document is meant to be an atomic model and the graph itself can be interpreted as a coupled model. The control specification of the presentation flow becomes input to the *coordination and local control system*. The data base of presentation unit is presented in Fig. 5.

## Automatic generation of platform independent telepresentation session source code

The presentation unit controller stores all object's models and allows easy changes and additions to the session flow. It includes a control interpretation system coded in JAVA, which interprets selected objects. It's inputs are a logic query plan (interactivity) and rules from a knowledge system (rule–base) for decision making and state accepting in order to run the local discrete control.

## Client–Server system development tool based session application and real–time realization

The final result — the telepresentation — will run in a client/server–environment. This environment consists of three major components: the *presentation*

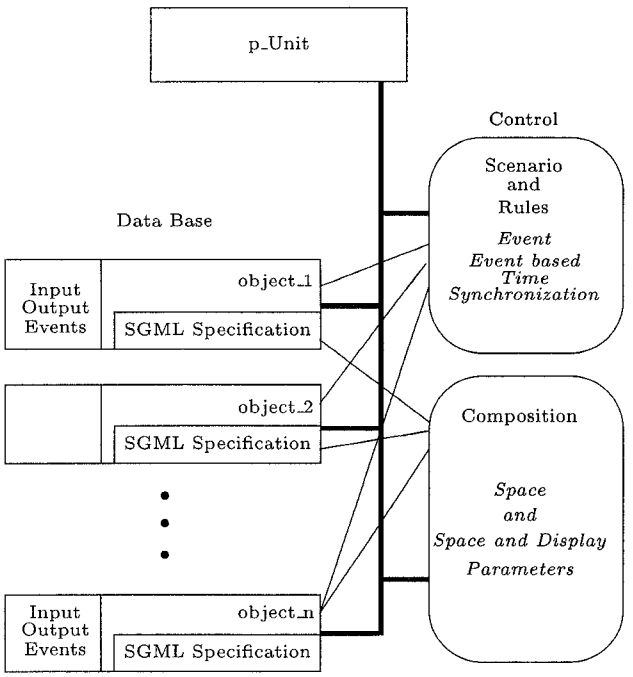


**Figure 5. Data base of p_Unit**

*server*, the *moderator client* and the *participant client*. All components are connected to the database and a mail–tool for exchanging electronic mail or simple notes.

### Presentation server

The presentation server is meant to be the central point for the presentation. It provides application programs (e.g. for exercises), the presentation's source and the control mechanism. Most of the control input comes from the moderator (by the moderator client) and it controls the participant clients.

### Moderator client and participant client

The moderator client (also called teacher client) controls the presentation server and both directly and indirectly the participant clients (also called student clients). The participant (student) clients can run in two major modes. The first is called the *learn mode*, where the participant (student) only has restricted access (read–only) to specified documents, therefore it is said to be remote controlled. The participant's client has a local notebook for writing own remarks, which is stored local in the participant's home directory. In this mode the moderator client can bring up the participants' screen contents in a window of his display.
The second mode is called the *teach mode*, where participants work interactively. By example the moderator sends exercises via e–mail to be done by the participants. In this case the presentation server provides application programs for the participants clients, for example a compiler system or Mathematica.

### Interaction and communications of clients

In both modes the moderator (teacher) client controls the participant (student) clients. Anyway, the participant must be able to interrupt the presentation for questions etc. and the moderator client has to react by accepting or rejecting interrupts. Most communication among clients is to be done by e–mail, participant interaction has to be defined by the moderator (teacher). In most cases the telepresentation session will run in a homogeneous network. However, it has to be considered that hardware equipment inside the presentation network is not unique. In that case the presentation server has to provide application program executable for the specified hardware platforms, what will be a major problem for the design.

### Time and space control (delay problem) and synchronization (predictive control)

Communication among clients becomes complicated in teach mode, where participants do not sit in an electronic classroom at local but are distributed around the world. First we have to consider the time–delay for long distance connections via internet, where a synchronization of remote processes in order to obtain a virtual electronic classroom effect becomes necessary. A way to a solution is to use different traces for each client derived from one default–trace.
The multilevel structure of telepresentation unit design system shown Fig. 6.

## References

[1] O. Beltscheva, I. Georgiev, "The Development of a Multimedia System for Education", in *Multimedia in Open Environment*, W. Herzner (ed.), Springer Verlag, Vienna, New York, 1995

[2] T. Casey, J. Fromont, "MHEG, Scripts, and Standardization Issues", in *Multimedia in Open Environment*, W. Herzner (ed.), Springer Verlag, Vienna, New York, 1995

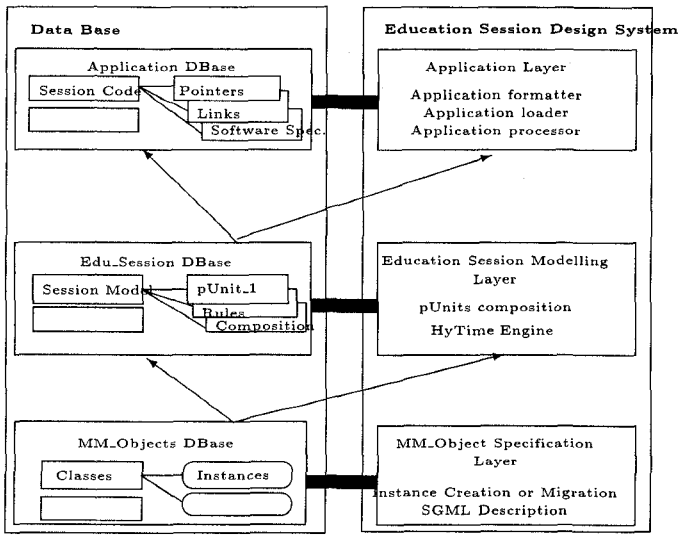


**Figure 6. Multilevel structure of p_Unit design system**


[3] F. Colaitis, F. Bertrand, "The MHEG Standard: Principles and Examples of Applications", in *Multimedia in Open Environment*, W. Herzner (ed.), Springer Verlag, Vienna, New York, 1995

[4] W. Jacak, "CAD System for Development of Telelearning Session in Open Environment", *Proc. of Conference TELEMEDIA'96*, Hagenberg, Austria, 1996

[5] M. Marshall, W. Samson, P. Duguard, "A proposed Framework of predicting the Development Effort of Multimedia Courseware", in *Multimedia in Open Environment*, W. Herzner (ed.), Springer Verlag, Vienna, New York, 1995

[6] W. Jacak, J. Rozenblit.: CAST Tools for Intelligent Control in Manufacturing Automation, *Lecture Notes in Computer Science*, **763**, Springer Verlag, pp. 203-220, 1994

[7] B.P. Zeigler.: *Multifacetted Modeling and Discrete Event Simulation,* Academic Press, 1984