Innovative Digital Learning

Ion SMEUREANU, Bucharest, Romania, smeurean@ase.ro
Gheorghe RUXANDA, Bucharest, Romania, ghrux@ase.ro

The new programming technologies allow for the creation of components which can be automatically or manually assembled to reach a new experience in knowledge understanding and mastering or in getting skills for a specific knowledge area. A Visual C# .NET implementation under development is discussed.
Key-words: *learning component, user control, automatic assembly, adaptor.*

Introduction

E-learning groups methods and techniques, traditional or computer aided (like multimedia processing, asynchronous or synchronous communication, web pages, large databases etc.) assisting the subject in the learning process.

In 1997 Maddux, Johnson and Willis provided a simplified approach for educational software classification; they defined two application type levels:

- **First level** includes software applications which are targeted for an easier, more intense and more efficient delivering of the same knowledge as in the classic method. The user involvement is **low**, the **software predetermines** almost all that is going on the display; **interaction** between user and computer is **pre-configured** by the software author; user contribution complies with a predetermined scenario; applications aim to acquire knowledge by memorizing.
- **Second level applications** mean new and more efficient learning techniques. They allow more active involvement of the user; **the user controls everything that happens**; user-computer interaction is driven by user at runtime; there is an extended range of inputs and actions accepted by the computer; **creative** actions **prevail**.

2. E-learning and innovation: component learning model

In practice, as in specialized literature, the concept of *learning object* is intensely used, but it is not strictly defined. A **learning object** is defined as any entity, digital or non-digital, that may be used for learning, education or training (IEEE, 2002); it can be reused or referenced any time in a computer-based learning process. Examples of such objects are prints, studies, exercises, texts, audio lessons, courses, curricula etc.

The **learning component** is an object implementing interfaces; these interfaces make it able to recognize other related objects with which it can interact **inside a semantic network**. Functionally, the components are small executable pieces which interact with each other **even during the design time of a client application**; they complete one another, they bring themselves new properties and references through which they connect, "coupling" and acting as a whole (this is where the term component comes from).

The success of new learning technologies is related to the paradigm-shift, from traditional content-centered and instructor-led models towards an interactive focus on the teacher/learner. **Component-based learning is the process of assembling existing software components in an application in such a way that they interact to get a predefined functionality.**

The task of **content management** is partially accomplished by learning components, in a way that innovative digital learning objects can be developed; they are capable to **interpret, contextualize and react depending on the architecture where they are assembled**.

The **semantic model** can be partially supplied by the human subject because learning components can be **assembled manually**, so as to provide a large opportunity both for students and professors to exercise their creativity and vision, and to conceive and develop learning resources by themselves. An important feature of e-learning systems based on learning components is that both **teachers**

and learners can become active producers of educational content. Tools for high quality content authoring are already available and anyone with enough creativity can compete in innovation. Component-based education requires active engagement of students' effort over an extended period of time, progressive and innovative. The students experiment, learn from failures, reply to the challenges and become deeply involved. Even the homework can be delivered as component-based activities.

Objects used for learning exist and cooperate at different levels of granularity. We no longer talk of individual objects, but of learning frameworks that can work in two interchangeable modes:

- **author**, when professor and student create and test training applications;
- **reader**, when already authored lessons are experienced.

The framework has to support two kinds of processes: **the decomposition of learning objects** into their components as well as the **automatic or manual assembly** of these components in real-world applications. It is not enough that learning objects satisfy some formal criteria of coupling/decoupling; the aggregation must also be pedagogically effective. The coupling /decoupling of learning objects is a considerable challenge, mixing ideas from pedagogy and software engineering [4]; the challenge is to attain new significance by composing reusable components; some advantages are revealed in the example below, which is about learning objects for mathematics.

Personalizing the learning process means creating a development plan that is perfectly adapted to the knowledge level, needs, expectations, personal pace and learning habit of the student. Intelligent e-learning systems aim to implement customized learning models; a personalized course can be configured by automatically selecting and sequencing the needed learning components.

Components are running since client application design, therefore they can automate a significant part of component-coupling, in the way that the application requires. Part of the properties is browsable and therefore can be visually modified by mouse right-click actions, without writing code or with minimal programming effort.

Components benefit of abstractization using interfaces, which standardize communication and force components to respect minimal communication rules that are generally accepted (like *IComponent* inherited by an user control, *IDataObject* for drag and drop action).

```
public class EvalExpresCtrl : System.Windows.Forms.UserControl, IDataObject
{
  public EvalExpresCtrl()  {InitializeComponent();}

[Browsable(true),EditorBrowsable(EditorBrowsableState.Always),Category("Custom")]
  public string ExpressionLatex
    { get{ return latExpression;}  set   {latExpression=value;}   }
  public string[] GetFormats(bool autoConvert)
    {
     if (autoConvert)  {  return new string[] { "Math", "Latex","Bitmap", DataFormats.Text};   }
      else  {   return GetFormats(); }
    }
/* .... */
}
```

By running in a virtual machine the components are no longer platform-dependent; the components adapt themselves to new versions, inconsistencies due to versioning thus vanishing. Components can be easily reused and rapidly integrated in new applications, without the need of implementation details, only by knowing the coupling interface. A schematic structure of an educational component is represented in the figure 1.

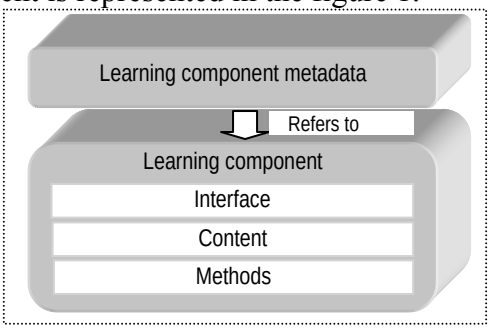

Fig. 1 Learning component structure

Learning objects are *self-contained*, bearing information about themselves, which allow

to be independently dragged in another location where they are perfectly integrated. It is because of this that every learning object encapsulates metadata; these metadata ensure: resource identification (by title, version, resource type), indexing for fast searches (keywords, author), exposing owned formats or newly available formats using converters, publishing accepted facilities etc.

From the point of view of educational strategies, having a warehouse with such components is not enough; using metadata and interfaces it is very hard to depict all possible semantic relationships because of the diversity of educational processes.

Visual .NET environment offers at the moment, one of the most interesting models for components aggregation and communication. In addition, it facilitates objects storing (ADO.NET), Web forms design (ASP.NET) or distributed services requesting (Web Services). .NET provides a **dynamic publishing and subscribing mechanism**. The .NET components are executables, therefore they know (or they can find about) their interoperability characteristics **at runtime** in a determined context; consequently they can publish their interface. A classic example is a graphing component that can subscribe to a varying number of external mathematical functions or, in his turn, ask for data from a table object.

3. Practical approach

A telling example can be an application for mathematics training; it raises enough problems so that the definition, the integration and learning object handling will be made clear. A schematic interaction of objects used in such case is represented in figure 2.

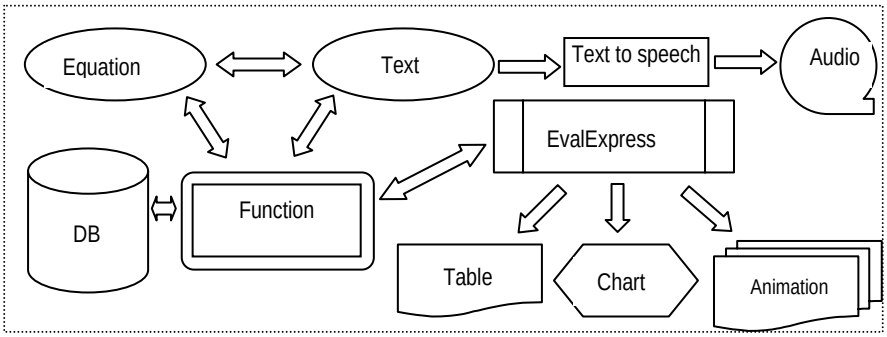

Fig. 2 Component interaction in figuring out the function concept

Learning object diversity is extremely large; we just point out the most representative types of components, in order to easier understand component assembling ways and typical application structures.

A. **Content** – is the skeleton of an application, offering generic support for an architecture aiming a clear learning goal; usually consists in a hierarchical/graph network for logic navigation among inter-related knowledge sets. It expresses the relationship between learning objects and the syllabus, the course or other higher organizing structure in which they are delivered.

B. **Elementary objects** usually placed as leaves in the tree and having specialized editors; Text / RichText, Equation, Sound, Graph, Image, Animation, Video are the most used elementary objects.

C. **Matching mechanism** as an abstract class managing logical associations between objects such as belonging to an object set or one to one/multiple-choice quizzes.

D. **EvalExpress – mathematical expression evaluator** providing support for **runtime compilation** and work with parameterized functions. The **EvalExpress** acts as a mini-compiler, doing syntactic validations, memory allocation and dynamic evaluations, during the whole execution of the client application.

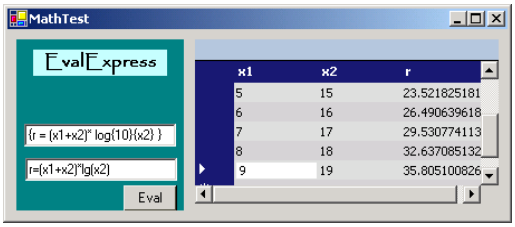

Fig. 3 EvalExpress – Data Grid interaction

E. **Converters and adapters** aimed to adapt the outputs to the coupling interfaces, usually calling an overloaded cast operator. The diversity of elementary components requires bringing them to a common format; even in this case, problems remain to be solved, such as: what conversion is preferred when faced with multiple choices, which component initiates conversion at coupling time etc. For instance, the output of **Equation component** working in a visual form, can be a Latex string, easily managed, compressed or reentered in a visual format for updating; often we need conversions from/to formats accepted by a wide spread editors, like MS Word. Text to speech could be another usual converter.

Another common adapter is a **database adapter** charged with data compression and persistence of the objects. It offers a built-in mechanism for storing the state of an activity or students' work using component serialization. Serialization writes at low-level the binary representation of the .NET component content.

Another usual converter is an **XML converter**; it offers a structured and text-based format for storing and retrieving the state of **a component aggregation** as a support for cross platform portability.

F. **Standalone Application –** is an entity able to be executed in a standalone play regime on a specific platform. It includes also sub-categories "script" and source application, written in a programming language and becoming platform dependent after compiling and linking.

G. **Function** – covers a mathematically important data type, a continuous function represented by a method that takes a numeric argument and returns a numeric value. In addition, other attributes of a function are important, such as the domain over which it is defined. The function object offers the possibility to handle mathematical functions by analytical expression or by pointer to a library code; an expression evaluator control allows syntactical validation of the analytical form. Functions can be viewed using multiple components, such as graphs, visualizations or tables**.**

The new approach is to automate the coupling of components, building an adaptor which forces the system to expose **only a set of safe or desired interfaces** for a specific context. By exploiting the metadata and partial specification of the learning goalthat must be enforced, it can automatically and progressively build a centralized adaptor. This adapter will mediate contextual interactions among components by both performing the specified behavior and simultaneously managing possible deadlocks.

Bibliography

[1] Cleborne D. Maddux, D. LaMont Johnson, Jerry W. Willis, Educational Computing: Learning with Tomorrow's Technologies, Allyn & Bacon; 2001.

[2] Wiley, David, A. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy, Utah State University, Digital Learning Environments Research Group, 2002.

[3] Smeureanu, I., Reveiu, A., Dârdală, M., Educational Technologies Based on Software Components, *Informatica Economică*, vol. X, nr. 3, Editura INFOREC, Bucureşti, 2006.

[4] Tom Boyle, Design principles for authoring dynamic, reusable learning objects, in *Australian Journal of Educational Technology*, 2003, 19(1).

[5] math.hws.edu;portal.acm.org;ocw.mit.edu; www.epsilonlearning.com.