A Generic Model for Interactivity-Intense Intelligent Tutor Authoring Tools

Danilo L. Dalmon and
Leônidas O. Brandão
Department of Computer Science,
University of São Paulo
{ddalmon,leo}@ime.usp.br

Guilherme M. Gomes and
Anarosa A. F. Brandão
Escola Politécnica
University of São Paulo
guilhermeg@usp.br,
anarosa.brandao@poli.usp.br

Seiji Isotani
Institute of Mathematics and
Computer Sciences
University of São Paulo
sisotani@icmc.usp.br

Abstract — Intelligent Tutoring Systems (ITS) provide many features that improve learning and teaching experiences. ITS are usually interactivity-intense and content-specific. Interactivity-intense assignments are recommended for scaffolding learning, while content-specific systems can offer low flexibility regarding its possible pedagogical approaches and its uses by teachers. In order to overcome this limited flexibility, there are systems which let content-specific interactivity aside to provide authoring tools, on which teachers can author intelligent tutored assignments without programming. The generic model proposed in this article intends to address this problem providing flexible authoring tools for interactivity-intense assignments with tutoring features, letting teachers benefit from the flexibility of content authoring tools as well as the interactivity usually restricted to content-specific ITS. We introduce an application framework which implements this model. Also, we present a proof of concept, an Interactivity-Intense Intelligent Tutor Authoring Tool on the domain of geometry. It is available as free software.

Keywords — Authoring Systems, Computer Assisted Instruction, Electronic Learning, Intelligent Tutoring Systems.

I. INTRODUCTION

Intelligent Tutoring Systems (ITS) are educational software that provide intelligent feedback in order to improve learning experiences for students [1]. ITS’ main benefits to students are, for instance, recognize misconceptions, guide them through the study, enhance the speed of learning procedures and helping or preventing teachers from repetitive work [1], [2], [3]. There are many types of ITS, which depends on its structure and on its way to help students. Some examples are the cognitive, collaborative, and inquiry tutors [2], [4], [5] and [6].

Two aspects of educational systems that we consider important are interactivity and flexibility of use. Interactivity can be defined in many ways, depending on the point of view [7]. We highlight the importance of interactivity conceptual frameworks, which can guide the development of educational systems [7], [8]. For teaching and learning sciences, mathematics and technology, there are evidences on the relevance of types of interactivity such as manipulate and co-construct interactivity [9], which we call interactivity-intense assignments. Examples of ITS that explore these types of interactivity are [10], [11] and [12]. Moreover, ITS can be used in many ways. Considering the examples above, [10] can be used on the Internet with steady assignments, [11] is used as a desktop application and provide adaptable assignments, and with [12], teachers or programmers can author assignments from scratch. When an ITS provides a set of steady assignments, teachers do not have the work of creating their own, but in this case teachers cannot adapt the assignments to their own purpose. When an ITS has flexible assignments or authoring tools, the opposite happens, teachers must spend time to create their assignments, however the latter has an additional advantage over the former: if the teacher cannot produce assignments (particularly interactive-intense ones) the system usually provides a set of assignments from a repository.

So, in one hand interactivity-intense assignments and authoring tools are features that enhance the learning and teaching experiences [8], [13], on the other hand, producing good assignment is time consuming. By analyzing existing systems, we found no ITS with interactivity-intense assignments and authoring tools for teachers, but systems with one feature or other.

Our goal is to provide means to facilitate the development of ITS with these two features together. We propose in this article a generic model of ITS in order to develop systems with interactivity-intense assignments and authoring tools, which we call Interactivity-Intense Intelligent Tutor Authoring Tools. This model consists of an abstract architecture with detailed description of key features that facilitates the co-existence of the proposed features.

As a concrete example of a system created using the model proposed, we present an application framework for building Interactive Learning Modules (iLM). With this framework, a proof of concept on the domain of interactive geometry was implemented. This instantiation of the generic model was used to analyze its influence on the development process. In addition, we highlight some lessons learned and key aspects of the model and the framework on the features development.

This paper is organized as follows. Section II presents related work of ITS regarding interactivity and flexibility of use. Section III describes the proposed model and section IV the application framework and the proof of concept on
geometry developed using this model. The article ends with some conclusions and future work in section V.

II. RELATED WORK – INTERACTIVITY AND USE FLEXIBILITY IN ITS

This section analyzes briefly the literature on ITS regarding mainly the features of interactivity in assignments for students and flexibility of use or authoring tools for teachers.

A. Interactivity in ITS

Interactivity, as the interaction between user or learner and computer, is essential to improve learning and teaching through the use of computer [7]. In order to guide the use of interactive mechanisms on user interfaces of software systems some conceptual frameworks were developed [8], [9] and [14]. Using this approach, it is possible to highlight types of interactivity that enhance learning of specific domains, like sciences and mathematics [7,9]. We define interactivity-intense assignment as the one where user interfaces adopts different types of interactivity in order to enhance learning.

ITS, by definition, use at least two types of interactivity, one to allow users to enter some information and another for the computer to inform or guide her. In this context, interactivity-intense assignment should provide the user additional ways for interacting.

Examples of interactivity types in ITS are: flux control in Computer Aided Instruction (CAI) and similar systems [10], [15]; manipulate in electronic circuits simulation systems [16]; object interactivity in most systems [10], [11], [12]; also, ITS present many response interactivity mechanisms, evaluation [11], and extending learner’s understanding [16].

B. Use Flexibility in ITS

In the case of educational software, flexibility of use is a key issue [13]. It consists of allowing the teacher to adapt the system’s behavior in order to better suit her pedagogical approach. When teachers have good training and critical thinking regarding teaching a given domain, they may opt to not use the system if it is not consistent with their opinions.

Interactivity is an aspect well explored by ITS, which is not the case for flexibility of use. Most systems provide assignments for students in a fixed set of problems, which is the case for ER-Tutor and SQL-Tutor [10] and C-POLMILE [11]; or the assignments are created automatically using templates, which happens for SCI-WISE [6] and Sherlock [12].

There are intelligent tutor authoring tools, which intend to facilitate the elaboration of tutored assignments [17]. However, usually one needs high programming skills in order to use them [17]. An innovative approach is presented by CTAT (Cognitive Tutor Authoring Tools) [2]. It provides authoring tools for teachers without the use of programming skills with a new paradigm of intelligent tutoring – example-tracing tutors [18]. This paradigm inspired us to help spreading authoring tools for teachers to more ITS.

As this is not an exhaustive review of literature, it is not possible to assure the inexistence of an ITS that has the two features: authoring tools and interactivity-intense assignments. Although, this review shows that most ITS do not consider flexibility of use as a desired feature, and for systems that have it, it is difficult to provide interactivity-intense assignments.

III. A MODEL FOR INTERACTIVITY-INTENSE INTELLIGENT TUTOR AUTHORING TOOLS

In this section we present the model for developing educational software with functionalities of: (i) intelligent tutoring, (ii) interactive assignments and (iii) authoring tools for teachers. First we present the main goals and the development context, followed by a general and a detailed description of each of the model’s components.

A. Goals and Development Context

We propose a model with four main goals: (i) to serve as basis for the design and the development of ITS; (ii) providing means to allow and make easier the creation of interactivity-intense and domain-specific features; (iii) to implement assignment authoring tools; and (iv) to assure that these features are compatible with each other. These goals are derived from the literature review described in last section.

In order to attain these goals, the model is presented as a generic system architecture with a description and a structure to any functionality it intends to provide. By using this model, one can develop an ITS with the desired features following a cookbook and reusing the proposed architecture.

This model was created for the design and implementation of tutoring features in an Interactive Geometry System called iGeom [19]. iGeom is part of a family of educational systems called Interactive Learning Modules (iLM), which main features are interactivity-intense assignments, authoring tools for teachers and communication with Learning Management Systems (LMS) [20], [21].

A new generic system model was created as a means to introduce tutoring features inside iGeom’s assignments, which do not consider tutoring features of suggesting or providing new assignments to students. This is a model for systems we call Interactivity-Intense Intelligent Tutor Authoring Tools, as it is intended for developing ITS with both features of interactivity-intense assignments and authoring tools for teachers, which iGeom was supposed to be the first [22].

Later, instead of providing an architecture specific to the Interactive Geometry domain, the model was generic enough for any of the existing iLM’s domain, and we expect it is generic for any domain whatsoever. Having developed this model, it was instantiated by implementing an application framework on the domain of iLM, which was firstly presented in [23]. The relationship between the proposed model and the iLM domain is detailed in section IV. First, we describe it in depth below.

B. General Description

As generic system architecture, this model is structured as functional components. These components are: (i) structural, (ii) domain, (iii) tutoring and (iv) assignment. The division on these components is derived directly on the model’s goals and intended features. On the following, we present briefly each component and then describe the communication among them.
- **Structural component** is responsible for initialization and setting the connections among the other components. It is also responsible for domain-independent features such as configuration and file management.

- **Domain component** provides domain-specific features. It is responsible for organizing the system model, e.g. geometry and calculus. Also, it implements the domain user interface, allowing the user to interact with domain-specific objects and providing the interactivity intensity required by the model.

- **Tutoring component** sets an architecture for intelligent tutoring features in a way compatible with domain and assignment components’ functionalities. It can react to students’ domain-specific actions or act on demand with messages or other tutoring features.

- **Assignment component** defines assignment authoring tools and assignment data management features. It assembles Domain and Tutoring components’ data in assignments for students and teachers.

Communication among these components include mainly:

(i) structural component initializing the other components that (ii) use domain-independent features provided by it; (iii) assignment component using domain and tutoring actions to create and manage assignments; and (iv) tutoring component reacting to domain actions. Fig. 1 presents a diagram with the four components of the model and arrows representing these data exchanges.

![Diagram of the model's components and communication among them.](image)

A system built with this model, can be used in two ways: (i) by teachers that author and edit assignments; and (ii) students that solve assignments. On each situation, the structural component initializes the system and sets specific features available to the user.

When a teacher authors an assignment, she must set domain features and define the assignment’s initial state and expected answer, using features provided by assignment and domain components. Then, she chooses the tutoring behavior and adds it to the assignment being authored, with assignment and tutoring components.

When a student solves an assignment, she gets the initial state defined by the teacher. Such state is provided by the assignment component. Then, she uses domain actions to solve the assignment (domain component), asking for tutoring actions when needed or using them when the teacher has specified (tutoring component).

**C. Model’s Components**

On the following, we describe the internal architecture of each component, list their main features and discuss how it can be used to develop a system on the proposed model.

1) **Structural Component**

This component has three main goals: (i) providing basic structure to the system; (ii) providing domain-independent features; and (iii) initializing and setting communication among other components. In order to implement these roles, its internal architecture must have sub-components responsible for them.

The basic architecture is defined by its conditions of use and its final user, such as Web-based, mobile, or desktop applications. This definition also sets most of domain-independent features, such as configuration, file management and communication with external systems. As these specific aspects of the system are essential to designing its architecture, the proposed model is restraint to define this component’s role. Moreover, the structural component must be able to initialize, connect and provide these features to the remaining components. This can be done with separate sub-components for initial procedures and for setting public interfaces.

During development, this component is usually the first to be created, as the other components depend on it. Also, third party frameworks are frequently used for this role. This component has no influence on the educational aspect of the system, so it can be designed as basic software [24], in contrast to the other ones.

2) **Domain Component**

Domain-specific features must be developed using the organization proposed by this component. By domain-specific features we mean functionalities that are particular to the educational domain of the system. For instance, in the domain of interactive geometry, such features are: create point, create line, move point, reflect point, etc; in the domain of calculus, there are: create function, draw function, calculate integral or derivative, etc; in mechanics, create body, set force, set gravity, view free-body diagram, etc.

These features are the essence to the system’s interactivity. If a system has these features and allows the user/student to interact and manipulate them in a way specific to the domain, the learning experience can be richer [7], [8].

In addition, this component sets the organization of the domain-specific features, which is composed of:

- **Domain Object** is the most basic component of a domain. Every concept that can be manipulated by the student’s actions must be a domain object.

- **Domain Actions** are the manipulations the user can do on the domain objects. These are abstractions that encapsulate the action but don’t execute them. This is important to allow the triggering of other features, such as tutoring and assignment components’.
• **Domain Model** is the implementation of the rules of a domain. It is responsible for executing the domain actions and changing the domain objects’ states.

• **Domain Converter** is necessary to translate domain specific data structures to domain independent ones so other components of the system are able to manipulate and storage them.

• **Domain User Interface** benefits from the underlying structure to enhance the interactivity. It must call domain actions whenever the user interacts with it and, as stated before, provide interactivity to enhance the learning experience.

By providing these sub-components to the tutoring and assignment components, domain component allows the model to assure that all features are compatible with each other.

3) **Tutoring Component**

Intelligent Tutoring features in the model for Interactivity-Intense Intelligent Tutor Authoring Tools are provided by this component. Its main goals are: (i) provide the student with intelligent tutoring actions; (ii) manage how these actions are being accessed by the student; and (iii) allow the teacher to define the tutor behavior. We define four elements for this component:

• **Tutoring Action** is similar to domain action on the domain component. It encapsulates an action that can be taken by the tutor. It can be a text message, highlight a domain object, provide access to external resources, etc.

• **Tutor Sensors** are observers of domain actions and other features, such as a “help button” that stimulate the intelligent module. For instance, a tutoring action can be executed proactively after a domain action or when the student presses the “help button”.

• **Tutor Authoring** lets the teacher define some behavior of the intelligent module. It is used by the assignment component to add tutoring data to the assignment structure.

• **Intelligent Module** is the intelligent component of this architecture. When stimulated by the tutor sensors, it considers data from tutor authoring and the assignment current state and calculates the best tutoring action to be executed.

This internal organization uses domain component’s features and offers features to the assignment component. It can be adapted from domain-specific ITS to add authoring features or from ITS authoring tools to add the organization from the domain component that allows interactivity-intense assignments.

4) **Assignment Component**

The assignment component has three main goals: (i) model an assignment; (ii) allow a teacher to author an assignment; and (iii) allow a student to solve an assignment. These features can be implemented by a component that manages assignments and is capable of presenting them to the user with specific configurations. The simplest one is a configuration that defines features for students differently that it does for teachers.

One of the main contributions of the proposed model is the organization of an assignment. This organization is specific for the compatibility with domain and tutoring features, which is one of the model’s goals. This component sets the definition of an assignment as composed of:

• **Initial State** is the state on which the student receives the assignment. It contains a set of domain objects and is defined during authoring.

• **Current State** is the state that the student is currently in. It contains a set of domain objects defined by the initial state and the domain actions done by the student.

• **Expected State** is a state that the teacher sets as the expected for the student to arrive at the end of the assignment. It contains a set of domain objects and can be used to automatic assess the solution sent by the student.

• **Configuration** is a set of instructions to the system that changes its behavior as defined during authoring by the teacher.

• **Tutoring Data** are instructions that define the behavior of the tutoring component when the student solves the assignment. Instead of using domain object or actions, this information uses tutor-specific data.

A method for authoring an assignment can be directly derived from this description. A teacher, in order to author an assignment, must define all five characteristics.

Domain actions and objects must be observed by the assignment component. During the assignment authoring, it is necessary to facilitate the attribution of objects to the assignment’s states and to manage the actions. During solving, it is necessary to trigger tutoring features.

This concludes the description of the proposed model for developing Interactivity-Intense Intelligent Tutor Authoring Tools. Next section describes how this model was used to implement an application framework for building systems of the iLM family.

IV. **AN APPLICATION FRAMEWORK AND A PROOF OF CONCEPT IMPLEMENTING THE PROPOSED MODEL**

After the creation of the model, we developed an application framework on the domain of iLM using it. After, we created the geometry system, this time instantiating the framework. This section presents the domain of iLM, the application framework and then the geometry system, in order to analyze the influence of the model on these developments and to highlight some lessons learned.

A. **Interactive Learning Modules – iLM**

Interactive Learning Modules (iLM) are a family of educational software intended to improve teachers’ and students’ tasks [21]. These systems are Java Applets1 with

1 http://java.sun.com/applets/
some common features. For teachers they provide interactive assignment authoring tools and communication with LMS. Authoring tools let teachers adapt their method to the use of the system and communication with LMS helps them organizing the assignments and grades taken within iLM. For students they provide interactivity and automatic assessment. Interactivity, as described in section II is important for learning and automatic assessment is an essential aspect of feedback.

Currently, there are four iLM [20]: (i) iGeom, an interactive geometry system; (ii) iGraf, a system for teaching functions and graphs; (iii) iComb, a counting system; and (iv) iWProg, a visual programming system. These systems inspired the abstraction that originated the domain and the assignment components of the proposed model. In spite of that, it is generic to any domain, not only iLM.

In the case of instantiating the model to the domain of iLM, interactivity and authoring components can be directly derived. The structural component is adapted to the application architecture and to their communication with LMS. The tutoring component is an addition to the current definition of iLM.

B. The Application Framework

Using the proposed model, an application framework was developed. We describe the framework’s architecture, how to use it to implement iLM, and how these aspects relate to the model.

The component architecture of the framework is presented on Fig. 2 using UML 2.0 notation. There are eight components, divided in three groups: (i) core components in bold font; (ii) domain-specific components in italic and underlined; and (iii) tutoring components in comic sans font.

![Component Architecture](image)

Figure 2. The component architecture of the iLM Framework.

The core components provide the iLM’ common features, such as communication with LMS and assignment authoring and it represents the model’s structural and assignment components. The domain components, the model and the user interface, provide domain-specific features and represent the model’s domain component. Two tutoring components, one for authoring and one for executing tutoring actions, provide tutoring features to iLM and represent the model’s tutoring component.

To use this framework just domain features must be created, since tutoring, assignment and other features are common to all systems in the iLM family. Therefore, one must follow these steps to use it: (i) define the domain and design all specific operations and objects; (ii) implement the model’s domain component in order — objects, actions, operations, converter and user interface; (iii) define configurable behaviors and additional features using plug-in based extensions.

The addition provided by this framework in relation to iLM’ features is the tutoring component. This feature is based on the plug-in architecture for extensions. The example-tracing tutor paradigm is used [18]. Using the model’s specification and adapting this paradigm, it is domain-independent, so it can be used for any iLM. One component is responsible for authoring tutoring actions for assignments and the other calculates the best tutoring action for a given assignment state. The code and manuals for using this framework are available at our institution’s website.

Analyzing this development, the model proposed on section III influenced every framework’s component. This influence and the instantiation from the model to the framework was probably easier than it will be for other domains and ITS because of the common features of existing iLM: authoring tools for teachers and interactivity. However, a good amount of work was dedicated to the development of the tutoring features. In order to adapt an ITS to the proposed model, most effort will be taken to create the missing features, whenever it is the authoring tool, the interactivity or both.

C. A Proof of Concept on the Domain of Geometry

In order to make a first evaluation of the proposed model, and the application framework created with it, regarding the development of systems Interactivity-Intense Intelligent Tutor Authoring Tools, we developed a proof of concept on the domain of geometry. The work executed was the adaptation of an existing system, iGeom. It was restructured in order to be compatible with the framework’s architecture and code requirements, representing most of the model’s specifications.

Currently, iGeom has a legacy code which is only responsible for domain features, and uses the framework’s code for other ones. When a teacher authors assignments, she uses geometry-specific user interfaces from domain components and the assignment organization from assignment component. When a student solves an assignment, she uses specific actions to manipulate geometric objects from domain component and can access tutoring actions from the framework’s example-tracing tutor. Fig. 3 shows screenshots of iGeom, the assignment solving and tutor authoring, with the states to which tutoring actions are assigned.

This proof of concept is sufficient to present the possibility of creating Interactivity-Intense Intelligent Tutor Authoring Tools with the use of the proposed model. Also, in comparison with developing the system from scratch, having a generic architecture that defines the key features of the system reduces development time and effort.

---

V. CONCLUSIONS

We found a lack of ITS with both authoring tools and interactivity-intense assignment features on the literature. To facilitate the integration of these features in a compatible way to the tutoring features on architectural and design level, we presented a generic model for a family of systems called Interactivity-Intense Intelligent Tutor Authoring Tools. This model was tested by developing an application framework and an interactive geometry system.

The main contribution of this paper is considering the flexibility of use and interactivity features in designing educational systems in general and ITS in particular. Also, by providing abstract reusable concepts and architectures such as the proposed model, the development effort and time of these systems are reduced. The model presented centralizes many concepts and design decisions that serve as basis for developing educational systems in a way that the design effort can be focused on the instructional dimension.

The results presented are the first of a group. As the application framework and the first Interactivity-Intense Intelligent Tutor Authoring Tool are done, we are developing new ones. Currently, all iLM are being restructured to this new architecture. Moreover, with the results from these other developments, it will be possible to make a more robust evaluation of the proposed model and in particular of the framework.

ACKNOWLEDGMENTS

Danilo L. Dalmon is supported by FAPESP under grant 2010/06805-2. This work was partially supported by FAPESP (2011/10926-2) and CNPq (550449/2011-6).

REFERENCES