# PLITS: A PATTERN LANGUAGE FOR INTELLIGENT TUTORING SYSTEMS

Dina Salah, The American University in Cairo, Computer Science Department, dsalah@aucegypt.edu
Amir Zeid, The American University of Kuwait, Division of Sciences and Engineering, azeid@auk.edu.kw

## Abstract

The design and implementation of Intelligent Tutoring Systems (ITS) is a very complex task, as it involves a variety of organizational, administrative, instructional and technological components. In addition, there are no well established methodologies or development tools for ITS implementation. Therefore systematic, disciplined approaches must be devised in order to leverage the complexity of ITS implementation and achieve overall product quality within specific time and budget limits (Vladan Devedzic & Harrer, 2004).

The goal of patterns within software community is to provide software developers with solutions to recurring software problems. However, the concept of patterns has received surprisingly little attention so far from researchers in the field of ITS (Vladan Devedzic & Harrer, 2004).

In this research work, we tried to mine patterns by reverse-engineering some of the existing ITS systems. These identified patterns were semantically organized and categorized to form the basic core of a PLITS: A Pattern Language for Intelligent Tutoring Systems.

## 1. Introduction

A pattern describes a problem that occurs over and over again in our environment and then describes the core of the solution to that problem in such a way that you can use this solution a million times over without ever doing it the same way twice(Alexander, Ishikawa, & Silverstein, 1977).Moreover, a pattern language is a collection of such solutions which, at every level of scale, work together to resolve a complex problem into an orderly solution according to a pre-defined goal. Pattern Languages include rules and guidelines that suggest the order and granularity for applying each pattern in the language (Vladan Devedzic & Harrer, 2004).

ITSs incorporate built-in expert systems in order to monitor the performance of a learner and to personalize instruction on the basis of adaptation to the learner's learning style, current knowledge level, and appropriate teaching strategies(Liegle & Woo, 2000). The classical ITS architecture is composed of the following components illustrated in figure (1):

- **Expert Model**: This model contains the domain knowledge.
- **Pedagogical (Tutor) Model**: This model provides the knowledge infrastructure necessary to tailor the presentation of the teaching material according to the student model.
- **Domain Model**: This model contains the knowledge about the actual teaching material.
- **Student Model**: This model stores details about the student's current problem-solving state and long-term knowledge progress, essential for adapting the material to the student's characteristics
- **Communication (User Interface) Model**: This model is responsible of user interaction.

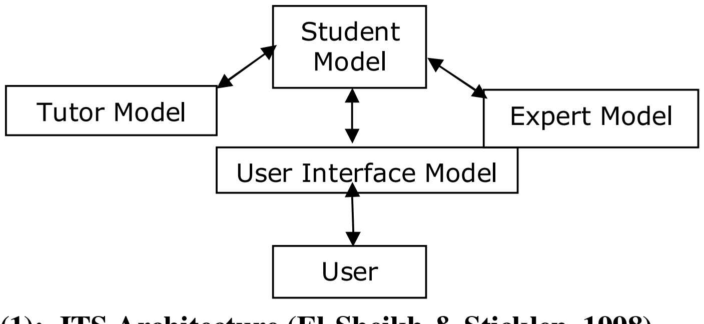


**Figure (1): ITS Architecture (El-Sheikh & Sticklen, 1998)**

A recent analysis of a number of existing ITS architectures has revealed that many ITS designers and developers use their own solutions when faced with design problems that are common to different systems, models, and paradigms. However, a closer look into such solutions and their comparison often shows that different solutions and the contexts in which they apply also have much in common, just like the corresponding problems do. In all such cases we can talk of the existence of patterns (Vladan Devedzic & Harrer, 2004).

## 2. The Proposed Pattern Language for Intelligent Tutoring Systems (PLITS)

This research, presents PLITS: A Pattern Language for Intelligent Tutoring Systems. PLITS is not a mere collection of patterns that can be used in the implementation of ITSs it includes rules and guidelines that explain how and when to apply its patterns to solve a problem which is larger than any individual pattern can solve.

In this research work, we first identified the functional requirements of ITS which will be summarized in table (1) and then tried to discover these features in a number of real ITSs that are broadly used and are listed in table (2). If these features were indeed found in existing ITSs, then these features were considered widely adopted and applicable and were therefore included in our pattern language for ITSs. In table (3) we map these Functional Requirements to their corresponding ITS patterns

**Table (1): ITS Functional Requirements and Related ITS Model**

| Functional Requirement | ITS Model |
|---|---|
| Curriculum Composition | Domain Model |
| Lesson Presentation Planning, Exam Generation | Tutor Model |
| Curriculum Instantiation | Domain Model |
| Collective Student Information Instantiation | Tutor Model |
| Students Registration and Access Control | Student Model |
| Student Model Standardization | Student Model |
| Determining and Achieving Student Goals | Student Model |
| Initializing Student Model | Student Model |
| Update and Maintenance of Student Model | Student Model |
| Representing Student Status Per Topic | Student Model |
| Representing Student Status Per Exercise | |
| Student Exam Grade Computation | Tutor Model |
| Monitoring Student Progress and Updating Students' Progress Reports | Tutor Model |
| Creating and Customizing Curriculum | GUI Model |
| Guiding Students Through a Complex Task | GUI Model |

Table (2) provides a list with the ITSs that were searched for patterns

**Table (2): ITSs that were searched for Patterns**

| Name | Domain |
|---|---|
| SQLT-Web (Mitrovic, 2003) | Database Learning(SQL) |
| Cyberphysique (Nkambou & IsaBelle, 1998) | Physics |
| Web Passive Voice Tutor-Web PVT(Virvou & Tsiriga, 2001) | Language Learning |
| Verb Expert (Fum, Giangrandi, & Tasso, 1989) | Language Learning |
| CAPIT: An ITS for Capitalization and Punctuation(Mayo, Mitrovic, & McKenzie, 2000) | Language Learning |
| Intelligent Language Tutoring System for Grammar Practice(Heift, 2001) | Language Learning |
| An Interactive Course Support System for Greek (Heift, Toole, McFetridge, Popwich, & Tsiplakou, 2000) | Language Learning |

## 2.1 PLITS Pattern Categories

While formulating PLITS we aimed to follow a comprehensive approach to cover all aspects related to ITS implementation in order to provide a road map for any ITS developer or designer. To provide this comprehensive approach a number of pattern categories were used in PLITS including access, instructional, design, adaptive and interaction patterns. Although some of those patterns were not intended for ITSs, yet we found that they can be useful in ITS implementation.

1. **Access Patterns:** Access Patterns are concerned with the ways that users may access the various resources (Paris Avgeriou, Retalis, & Papasalouros, 2003).
2. **Instructional Patterns:** Instructional Patterns are concerned with the various tasks that tutors perform in order to create and edit courses and learning resources (Paris Avgeriou, Retalis, & Papasalouros, 2003).

Both Instructional Patterns and Access Patterns are used in the Learning Management Systems domain. However, we found out that they can be useful in ITS implementation if they were modified to suit the ITS developers and designer needs.

3. **Design Patterns**: Design Patterns can be divided into the following subcategories:
   - **Creational Patterns:** Creational Patterns abstract the instantiation process. They help make a system independent of how its objects are created, composed, and represented (Gamma, Helm, Johnson, & Vlissides, 1995).
   - **Structural Patterns:** Structural Patterns are concerned with how classes and objects are composed to form larger structures. Structural class patterns use inheritance to compose interfaces or implementations (Gamma, Helm, Johnson, & Vlissides, 1995).
   - **Behavioral Patterns:** These patterns are concerned with algorithms and the assignment of responsibilities between objects. These patterns describe patterns of communication between objects or classes (Gamma, Helm, Johnson, & Vlissides, 1995).
4. **Adaptive Patterns:** Adaptive instruction can be defined as real-time modification of the instructional curriculum, learning environment to suit different student characteristics ("ELEN Project").

   Adaptive Patterns are patterns that are used in Adaptive Learning Systems. However, we found that there is a number of adaptive learning patterns that can be of great use to the designers and developers of ITSs if they were modified to suit the ITS developers and designer needs.

   More details on adaptive patterns for ITS implementation can be found in (Salah & Zeid, 2009).
5. **Interaction Patterns:** Interaction Patterns are focused on solutions to problems that end-users have when interacting with systems. The patterns take an end-user perspective which leads to a format where usability is the essential design quality (Trætteberg & Welie, 2000).

It is important to note that some of the discovered patterns were based on existing patterns; however, we modified some of these patterns in order to suit the needs of ITS implementation. All modified patterns have new names that begin with the word **"New"**.

## 2.2 PLITS Pattern Template

Documenting a pattern is the first step to build a pattern language. The GOF pattern template was used since it is very complete and provides straightforward guidelines for implementing the patterns into software by including more implementation details rather than generic solutions.

## 3 Building an Intelligent Tutoring System

The process of building a typical ITS is composed of four phases; building the Domain Model, Student Model, Tutor Model and the Graphical User Interface Model. However, this is the typical sequence followed, nevertheless, it is not mandatory to strictly follow this order and some ITS developers use a different order for building ITSs.

In section 3 we will highlight the patterns that can be useful in every phase, and then in section 4 we will present PLITS our proposed pattern language for ITSs.

### 3.1 Phase One: Building the Domain Model

The domain model contains the knowledge about the actual teaching material, since the teaching material is usually composed of a number of topics. Each topic is composed of a number of learn items, each topic has a number of exercises related to it and each learn item has a number of examples related to it, furthermore, since the curriculum is shared by all students but yet is tailored to every student current knowledge level thus a single curriculum instance and global student access is needed.
This led us to identify two patterns that are needed in building the domain model; the Whole Part Pattern and the Singleton Pattern.

**Pattern 1: The Whole Part Pattern**

**Classification:** Design Patterns
**Intent:** Helping with the aggregation of components that together form a semantic unit.
In ITSs teaching material is composed of a number of topics. Each topic is composed of a number of learn items, each topic has a number of exercises related to it and each learn item has a number of examples related to it.
**ITS Applicability:** The Whole Part Pattern can be applied in the domain model implementation in aspects in ITS design related to composing topics, composing lessons and composing curricula.
**Known Uses**

1. **GET-BITS (Vladan Devedzic, Jerinic, & Radovic, 2000)** : GET-BITS is an object-oriented model of ITSs .GET-BITS Uses a variant of the Whole Part Pattern in the framework's lesson presentation planner and remedial actions planner which is a part of the student's knowledge examination and assessment.
2. **SimQuest (Joolingen, King, & Jong, 1997)** : SimQuest is an intelligent biomedical simulator for medical training utilizing the expertise of professionals in the fields of 3D graphics programming, clinical training, game and simulation design, medical illustration, human factors and mechanical engineering. SimQuest focuses on the use of advanced simulation and gaming technologies for medical training. SimQuest uses the Whole Part Pattern in its lesson presentation planner to represent the teaching material and its constituent parts.

**Related Patterns:** Course Creation and Customization, Singleton, Adapter, New Student Model Initialization.
**References:** The Whole Part Pattern is one of the Pattern Oriented Software Architecture patterns (Buschmann et al., 2000).

**Pattern 2: Singleton**

Since this pattern is one of the GOF patterns we will only list the pattern template items that shows its usefulness in the ITS field.

**Classification:** Design Patterns (Creational Pattern).
**Intent:** Ensuring that a class has only one instance, and provide a global point of access to it.
**ITS Applicability:** The Singleton Pattern can be applied in the Domain Model implementation in ITS design in Curriculum Instantiation since we need a single curriculum instance and global student access to it.
**Related Patterns:** Whole Part, Observer.
**References:** The Singleton Pattern is one of the Gang of Four patterns (Gamma, Helm, Johnson, & Vlissides, 1995).

## 3.2 Phase Two: Building the Student Model

The following patterns are needed in building the student model.

**Pattern 3: Registration-Authentication-Access Control**

**Classification:** Access Patterns
**Intent:** Providing a standard registration mechanism for every user of the system. This can be achieved through building a web interface related to a database with user data and providing a mechanism for user authentication.
**ITS Applicability:** ITSs are large, multi-user systems**.** Due to security, privacy and institutional policy reasons, students' access to the resources of the ITS must be restricted to authorized students only. How can all the different students' access rights and privileges be effectively managed?
**Known uses**
1. **SQLT-Web (Mitrovic, 2003):** An ITS for teaching SQL (Structured Query Language).
2. **Cyberphysique (Nkambou & IsaBelle, 1998):** An ITS based on the World Wide Web that teaches physics**.**

Both (Mitrovic, 2003) and (Nkambou & IsaBelle, 1998) acquire information about a student through a login screen. Individual student models are stored permanently on the server, and retrieved for each student's session.
**Related Patterns:** User Model Definition.
**References:** The Registration-Authentication-Access Control Pattern is used in learning management systems; however, we discovered that this pattern can be useful to ITS developers and designers.

**Pattern 4: User Model Definition**

The User Model Definition Pattern is one of the patterns that were used in adaptive systems ("ELEN Project"), however, we discovered that this pattern can be useful to ITS developers and designers in creating and maintaining a student model and providing mechanisms to modify application features based on that in order to offer the student the best possible learning experience.

More details on the User Model Definition Pattern and its usage for ITS implementation can be found in (Salah & Zeid, 2009).

**Pattern 5: User Goals**

The User Goals Pattern is one of the patterns that were used in Adaptive systems ("ELEN Project"), however, we discovered that this pattern can be useful to ITS developers and designers in determining both short term and long term educational goals for each student which is an important component of the student model description.

More details on the User Goals Pattern and its usage for ITS implementation can be found in (Salah & Zeid, 2009).

**Pattern 6: New Student Model Initialization**

This New Student Model Initialization Pattern was based on the User Model Initialization Pattern that is used in Adaptive systems ("ELEN Project"), however, we discovered that this pattern can be modified to suit ITS developers and designers needs.

The original Student Model Initialization Pattern ("ELEN Project") deals with determining and providing the information needed to initialize the student model before all interactions in adaptive systems. It proposes that this can be done by one of the following methods:

- **User driven.** The user specifies explicitly what stereotype he belongs to.
- **Inferred by rules.** These rules indicate which user model elements and values can activate a stereotype.
- **Speculated by rules .**If the user does not specify his knowledge level then it is assumed to be average.

However, we modified the Student Model Initialization Pattern and introduced the New Student Model Initialization Pattern by initializing the student model with two types of knowledge:

- **Knowledge that is acquired from the students:** This knowledge should be determined with the guidance of both the User Model Definition Pattern and the User Goals Pattern.
- **Knowledge that can be acquired automatically from the system:** This knowledge can be acquired by implementing an entry exam that uses an Exam Generator Component. This component can use the pool of questions that was filled earlier by the teachers through Course Creation and Customization Pattern that will be discussed in section 3.4, and randomly choose a set of questions that represent different difficulty levels thus can accurately measure current student knowledge level per topic (stereotype).

More details on the New Student Model Initialization**,** its usage for ITS implementation and modifications made to the original pattern can be found in (Salah & Zeid, 2009).

**Pattern 7: New Student Model Maintenance**

The New Student Model Maintenance Pattern was based on the User Model Maintenance Pattern that is used in Adaptive systems ("ELEN Project"), however, we discovered that this pattern can be modified to suit ITS developers and designers needs.

The original User Model Maintenance Pattern ("ELEN Project") deals with the methods for capturing and maintaining changes that occur to student model elements as a results of interacting with the system. It proposes that this can be done by one of the following methods:

- Using a questionnaire form that indicates the amount of benefit the user gained from using the system.
- Interactive update of the user model by showing a pop-up form requesting the user to answer a question.

However, we modified the Student Model Maintenance Pattern and introduced the New Student Model Maintenance Pattern by capturing and maintaining changes that occur to student elements by using an exam generator component to conduct a per topic exam. This can occur after the student is presented by the topic learning material. The exam generator component can use the pool of questions supplied by the subject instructor to randomly select a number of questions that can test the current student knowledge level and according to the results of this exam the user model maintenance module can update the student model to reflect his current status and the topics covered and his knowledge level per topic.

More details on the New Student Model Maintenance and its usage for ITS implementation and modifications made to the original pattern can be found in (Salah & Zeid, 2009).

**Pattern 8: Adapter**

**Classification:** Structural Patterns
**Intent:** Convert the interface of a class into another interface that clients expect. Adapter lets classes work together that could not otherwise because of incompatible interfaces (Gamma, Helm, Johnson, & Vlissides, 1995).
**Participants**
In the context of ITS we have the following participants:

- Client: Collaborates with objects conforming to the Target interface.
- Adaptee: Defines an existing interface that needs adapting.

In ITSs the Adaptee participant can be represented by two classes; Topic class and Exercise class.

- Adapter: Adapts the interface of Adaptee to the Target interface. It also acts as the Target.

In ITSs the Adapter participant can be represented by two classes; StudentTopic class and the StudentExercise class.
**ITS Applicability**: The Adapter Pattern can solve two problems for ITSs:

1. **Representing each student's specific status per topic?**

Any ITS deals with subject topics, however, there is a problem with the Topic class. This problem occurs because this class doesn't match the domain specific interface that the application requires because it doesn't reflect the student specific status per topic.
**This class doesn't represent the following information:**

- The student stereotype per topic; whether he is a Beginner, Intermediate, Advanced or Expert student.
- What learning item is currently being learned inside the topic by the student?
- How many times the student viewed a certain topic?
- What are the exams that he took on a certain topic?
- Student grades on every exam concerning this topic.

That is where the Adapter Pattern fits in ITSs Design. An Adapter class is needed that is capable of converting the interface of a class (Topic) into another interface that the client expects (StudentTopic). And to provide us with the functionality that the adapted class (Topic) doesn't provide.

2. **Representing each student's specific status per exercise?**

Any ITS deals with exercises, however, there is a problem with the Exercise class. This problem occurs because this class doesn't match the domain specific interface that the application requires because it doesn't reflect the student specific status per exercise.
**This class doesn't represent the following information:**

- Did the student view this exercise or not?
- The student answer on this exercise and whether he answered correctly or not, this will help to understand any misconception that the student might have.
- The student score per exercise.
- The number of student tries for solving this exercise.
- Did the student use out all his tries in this exercise or not yet?
- The examples that is related to this exercise, this means that if the student answers incorrectly he is shown some teaching material to revise the learning items that is covered by this exercise, this teaching material includes some examples so a record should be kept of the examples that is related to this exercise.

That is where the Adapter Pattern fits in ITSs. A StudentExcercise class is needed to act as an Adapter that is capable of converting the interface of the Exercise class into another interface that the client expects. And to provide us with the functionality that the adapted class (Exercise) doesn't provide.

**Modifications Made to the Adapter Pattern Implementation**
The main difference between our implementation and the classic Adapter Pattern implementation is that the Adapter Pattern has two extra participants Target; which defines the domain specific interface that Client uses and Client; which collaborates with objects conforming to the Target interface.
**Related Patterns:** New Student Model Initialization, Whole Part, Master Slave, and New Student Model Maintenance.
**References:** This pattern is one of the GOF Patterns (Gamma, Helm, Johnson, & Vlissides, 1995).

## 3.3 Phase Three: Building the Tutor Model

The following patterns are needed in building the tutor model.

### Whole Part Pattern

The Whole Part Pattern can be used in the Tutor Model in the Lesson Presentation Planner and the Exam Generator Component.

### Pattern 9: Master Slave

**Classification:** Design Patterns
**Intent:** The Master Slave Pattern supports fault tolerance and computational accuracy. A master component distributes work to identical slave components and computes a final result from the results that these slaves return.
**ITS Applicability:** In ITSs the Master Slave Pattern can be used to compute the exam results of each student according to their grade in each particular exercise within the exam.
**Related Patterns:** Adapter, Observer, New Student Model Maintenance, and New Student Model Initialization.
**References**: The Master Slave Pattern is one of the POSA patterns (Buschmann, et al., 2000)

### Pattern 10: Singleton Pattern

ITS designers and developers can benefit from the Singleton Pattern in the Tutor Model in the instantiation of collective student information.

### Pattern 11: Observer Also Known as Dependents, Publish Subscribe

**Classification:** Behavioral Patterns
**Intent:** The Observer Pattern defines a one to many dependencies between objects so that when one object changes state, all its dependents are notified and updated automatically (Gamma, Helm, Johnson, & Vlissides, 1995).
**ITS Applicability:** In ITSs an Observer class is needed to observe the status of all students and update the reports with the new student status as he navigates through the learning path.
**Related Patterns:** Master Slave, Singleton.
**References:** The Observer Pattern is one of the Gang of Four Patterns (Gamma, Helm, Johnson, & Vlissides, 1995).

## 3.4 Phase Four: Building the Graphical User Interface Model

The following patterns are needed in building the graphical user interface model.

**Pattern 12: Course Creation and Customization**

**Classification:** Instructional Patterns
**Intent:** How can the instructors be assisted in building on-line courses in ITS so that some of the tasks they need to perform can be automated in order to decrease the time and effort of performing those tasks?
**ITS Applicability:** ITSs must provide instructors with appropriate tools for creating and customizing a course. Course creation can be based on design templates with pre-set interfaces, content structure and features.
**Known Uses**
1. **Cyberphysique (Nkambou & IsaBelle, 1998):** An ITS based on the World Wide Web that teaches physics**.** This ITS provides an authoring environment dedicated to teachers and pedagogical designers.
2. **Eon Tools(Murray, 1998)**: "Eon" is the name for a suite of authoring tools for building ITSs. Eon includes tools for authoring all aspects of intelligent tutors, including the learning environment, the domain knowledge, the teaching strategies, and the student model. Curriculum can be easily extended or modified to update information and theories, and new curriculum can be uploaded over the World Wide Web.

**Related Patterns:** Whole Part.
**References:** The Course Creation and Customization Pattern is one of the patterns that were used in learning management systems (Paris Avgeriou, Papasalouros, Retalis, & Skordalakis, 2003), however, we discovered that this pattern can be modified to suit ITS developers and designers needs.

The following patterns represent user interface patterns that focus on solutions to problems that end users have when interacting with systems. These patterns focus on usability as an essential design quality. Usability can be measured through one of the following usage indicators: Learn-ability, user guidance, memor-ability, speed of performance, error rate, satisfaction, task completion. Each pattern should state the impact on these usage indicators. If a pattern does not improve at least one usage indicator, it is *not* a user interface design pattern (Trætteberg & Welie, 2000).

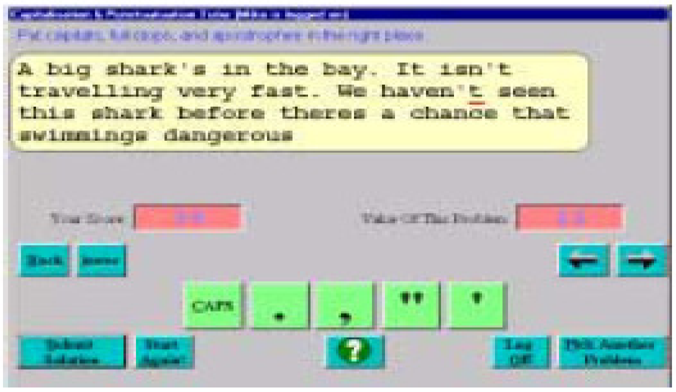
**Pattern 13: Wizard**

**Classification:** User Interface Patterns (Interaction Patterns).
**Intent:** Students sometimes need to perform an infrequent complex task consisting of several subtasks which ranges between 3 to 10 tasks where decisions that need to be made in each subtask may not be known to the user.
**Usability Principle:** User Guidance (Visibility) (Trætteberg & Welie, 2000).
**ITS Applicability:** ITSs should be designed in order to take the user through the entire task one step at the time. The user steps through the tasks and is shown which steps exist and which have been completed (Trætteberg & Welie, 2000). When the complex task is started, the user is informed about the goal that will be achieved and the fact that several decisions are needed. The user can go to the next task by using a navigation widget such as a button. If the user cannot start the next task before completing the current one, feedback is provided indicating the user cannot proceed before completion for example by disabling a navigation widget. The users are given feedback about the purpose of each task and the users can see at all times where they are in the sequence and which steps are parts of the sequence. When the complex task is completed, feedback is provided to show the user that the tasks have been completed and optionally results have been processed.

Users that know the default options can immediately use a shortcut that allows all the steps to be done in one action. At any point in the sequence it is possible to abort the task by choosing the visible exit (Trætteberg & Welie, 2000).

**Known Uses**

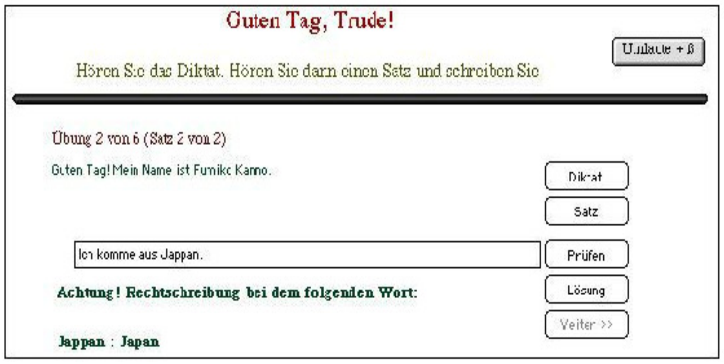
1. **CAPIT (Mayo, Mitrovic, & McKenzie, 2000):** An ITS that teaches the rules of English capitalization and punctuation. As shown in figure (6) instructions relevant to the current problem are clearly displayed at the top of the screen. Immediately below the instructions, and clearly highlighted, is the current problem. There are also navigation buttons to move back or next to assist student in navigating without overwhelming him with plenty of lists or options.



A big shark's in the bay. It isn't travelling very fast. We haven't seen this shark before theres a chance that swimmings dangerous

**Figure (6): CAPIT Main User Interface (Mayo, Mitrovic, & McKenzie, 2000)**

2. **The German Tutor (Heift, 2001):** An ITS for teaching German. After the student finishes answering the current question he can go on to the next exercise with the "Weiter" (next) button. The German Tutor utilizes the Wizard Pattern by using navigation buttons to move to next exercise to assist students in navigating without overwhelming them with plenty of lists or options. Figure (7) illustrates the Wizard Pattern in the context of the dictation exercise in the German Tutor.



Guten Tag, Trude!

Hören Sie das Diktat. Hören Sie dann einen Satz und schreiben Sie

Übung 2 von 6 (Satz 2 von 2)

Guten Tag! Mein Name ist Fumiko Kamo.

Ich komme aus Jappan.

Achtung! Rechtschreibung bei dem folgenden Wort:

Jappan : Japan

**Figure (7): Dictation Exercise in the German Tutor (Heift, 2001)**

**Related Patterns:** User Goals.

**References:** The Wizard Pattern is one of the Interaction Patterns (Trætteberg & Welie, 2000).

Table (3) maps our proposed ITS functional requirements to the corresponding ITS patterns. This summarized description can be used as a roadmap for ITS developers and designers.

## 4. PLITS –A Pattern Language for Intelligent Tutoring Systems

**Table (3): Mapping Between the Functional Requirements and the Corresponding ITS Patterns**

| # | Category | Pattern Name | Functional Requirement | ITS Model |
|---|---|---|---|---|
| 1 | Design Pattern | Whole Part | Curriculum Composition | Domain Model |
| | | | Lesson Presentation Planning, Exam Generation | Tutor Model |
| 2 | Creational Pattern | Singleton | Curriculum Instantiation | Domain Model |
| | | | Collective Student Information Instantiation | Tutor Model |
| 3 | Access Patterns | Registration-Authentication-Access-Control | Students Registration and Access Control | Student Model |
| 4 | Adaptive Patterns | User Model Definition | Student Model Standardization | Student Model |
| 5 | Adaptive Patterns | User Goals | Determining and Achieving Student Goals | Student Model |
| 6 | Adaptive Patterns | New Student Model Initialization | Initializing Student Model | Student Model |
| 7 | Adaptive Patterns | New Student Model Maintenance | Update and Maintenance of Student Model | Student Model |
| 8 | Structural Patterns | Adapter | Representing Student Status Per Topic | Student Model |
| | | | Representing Student Status Per Exercise | |
| 9 | Design Pattern | Master Slave | Student Exam Grade Computation | Tutor Model |
| 10 | Behavioral Patterns | Observer | Monitoring Student Progress and Updating Students' Progress Reports | Tutor Model |
| 11 | Instructional Patterns | Course Creation and Customization | Creating and Customizing Curriculum | GUI Model |
| 12 | Interaction Patterns | Wizard | Guiding Students Through a Complex Task | GUI Model |

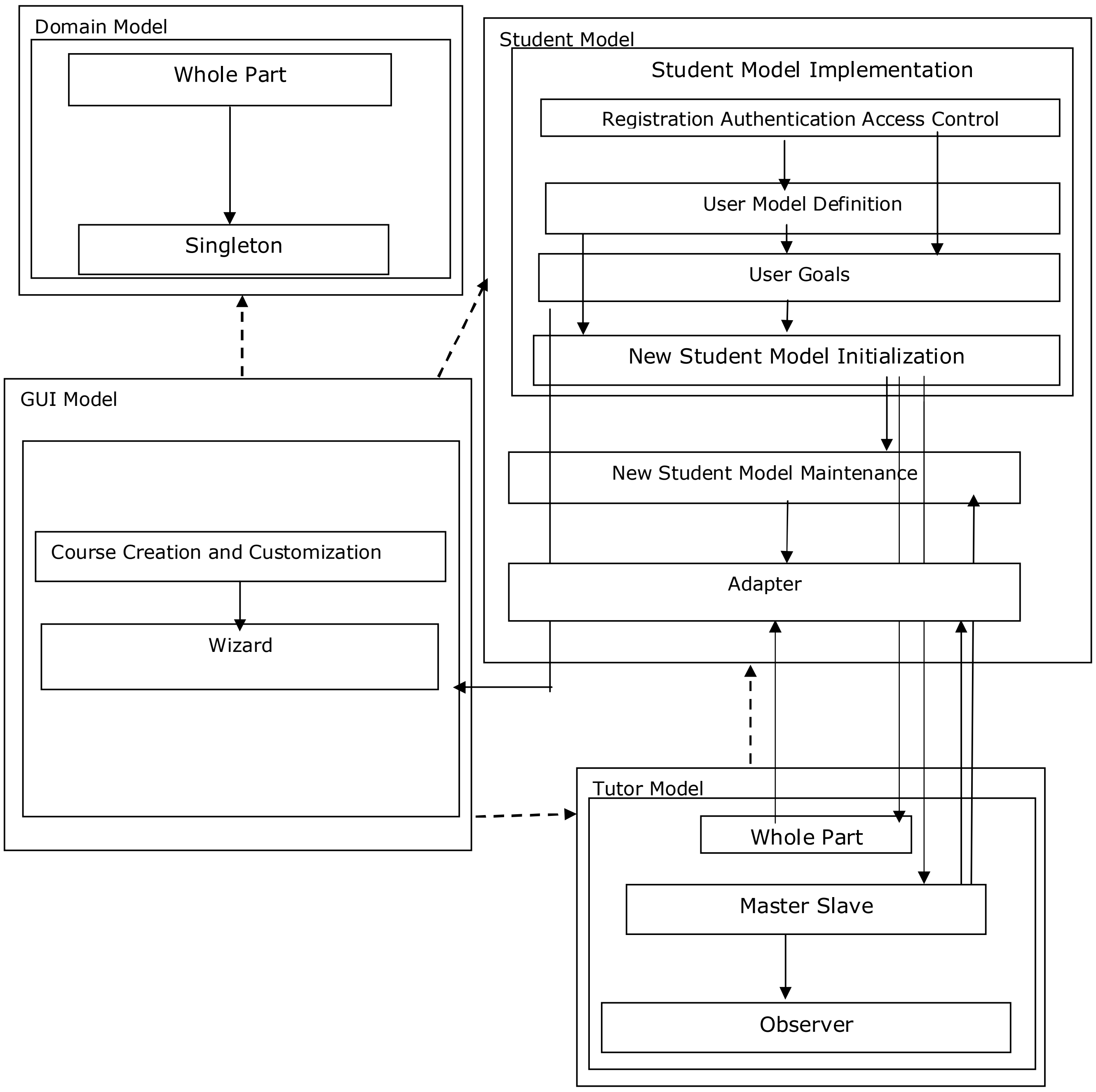
Figure(8) represents the Pattern Language Approach. In figure (8) we used two types of arrows:

$\dashrightarrow$ The dotted arrow is used to indicate the occurrence of a relationship between the ITS Models.

The solid arrow is used to indicate the occurrence of a relationship between patterns. This relationship is either a precedence or dependency.

$\longrightarrow$

**Figure (8): PLITS –A Pattern Language for ITS**


Domain Model
Whole Part
Singleton
Student Model
Student Model Implementation
Registration Authentication Access Control
User Model Definition
User Goals
New Student Model Initialization
New Student Model Maintenance
Adapter
GUI Model
Course Creation and Customization
Wizard
Tutor Model
Whole Part
Master Slave
Observer

## 5. Conclusion

ITS complexity can be overcome by creating a pattern language for ITSs that can help software developers resolve recurring problems encountered throughout all of software development process of any ITS. In this way, designers of new or existing ITSs, especially inexperienced designers, can take advantage of previous design expertise and save precious time and resources.

In this research work, we showed that one cannot talk of patterns in the ITS domain only in the context of ITS architectures. On the contrary, there are many kinds of other patterns that can be helpful in ITS implementation. This research started out by surveying the existing ITSs and their components in search for some possible common design decisions, common interactions among components, and common generalized principles underlying superficially different designs. We extracted patterns from numerous known examples, systems, architectures, designs, etc.

We investigated the use of design patterns, access patterns, instructional patterns, adaptive patterns and interaction patterns in ITS implementation. As a result of this research we formulated PLITS a pattern language for intelligent tutoring systems implementation that includes rules and guidelines which explain how and when to apply its patterns to solve a problem.

## 6. Directions for Future Research

**Suggested directions for future research include the following:**

1. Establishing an initiative for constructing a repository of patterns for ITSs in order to attract more researchers to deposit their own patterns. That would strengthen the pattern language and offer a wealthy pool of patterns for inexperienced designers of an ITS.
2. **Introducing Web Services into ITS implementation**: Web Services are self-contained, modular applications that provide a set of functionalities (for instance; ITS expert Model) to anyone that requests them. The main characteristic of Web Services is that they interact with the applications that invoke them, using web standards such as WSDL (Web Service Definition Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description, Discovery and Integration). Basing learner modeling on web standards has the advantage of enabling the dynamic integration of applications distributed over the Internet, independently of their underlying platforms (Kabassi & Virvou, 2003).

## References

1. Alexander, C., et al. (1977). *A Pattern Language*: Oxford University Press.
2. Avgeriou, P., et al. (2003). Towards a Pattern Language for Learning Management Systems. *Educational Technology & Society, 6*(2), 11-24.
3. Avgeriou, P., et al. (2003). *Patterns For Designing Learning Management Systems.* In Proceedings of the European Pattern Languages of Programming (EuroPLOP), Irsee, Germany
4. Buschmann, F., et al. (2000). *Pattern-Oriented Software Architecture:A System of Patterns*: John Wiley & Sons.
5. Devedzic, V., & Harrer, A. (2004). Common Patterns in ITS Architectures. *Künstliche Intelligenz, 18*(3), 17-21.
6. Devedzic, V., et al. (2000). The GET-BITS Model of Intelligent Tutoring Systems. *Journal of Interactive Learning. Research, 11*(3/4), 411-434.
7. El-Sheikh, E., & Sticklen, J. (1998). *A Framework for Developing Intelligent Tutoring Systems Incorporating Reusability*. In Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Methodology and Tools in Knowledge-Based Systems. Springer-Verlag. Retrieved.
8. ELEN Project. Retrieved May, 2004, from http://www2.tisip.no/E-LEN/outcomes.php
9. Fum, D., et al. (1989). *Tense generation in an Intelligent Tutor for Foreign Language Teaching: Some Issues in the Design of the Verb Expert*. In Proceedings of the Fourth

Conference on European Chapter of the Association for Computational Linguistics. Manchester, England. Association for Computational Linguistics. Retrieved.

10. Gamma, E., et al. (1995). *Design Patterns, Elements of Reusable Object-Oriented Software.* (First ed.): Addison Wesley Professional.
11. Heift, T. (1998). *An Interactive Intelligent Tutor over the Internet,* In Proceedings of the Proceedings of ED-MEDIA 1998, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Charlottesville, VA. Association for the Advancement of Computing in Education (AACE)
12. Heift, T. (2001). Intelligent Language Tutoring System for Grammer Practice. Retrieved May, 2004, from http://zif.spz.tu-darmstadt.de/jg-06-2/beitrag/heift2.htm
13. Heift, T., et al. (2000). *An Interactive Course Support System for Greek.Bourdeau,J.&Heller,R.(eds).* In Proceedings of the ED-MEDIA 00,World Conference on Educational Multimedia,Hypermedia &Telecommunications, Montreal Canada. Association for the Advancement of Computing in Education (AACE)
14. Joolingen, W., et al. (1997). *The SimQuest Authoring System for Simulation-Based Discovery Learning.* In Proceedings of the Artificial Intelligence in Education, Tokyo. IOS Press
15. Kabassi, K., & Virvou, M. (2003). Using Web Services for Personalized Web-based Learning *Educational Technology & Society, 6*(3), 61-71.
16. Liegle, J., & Woo, H.-G. (2000). *Developing Adaptive Intelligent Tutoring Systems:A General Framework and Its Implementations.* In Proceedings of the ISECON Philadelphia, PA, USA
17. Mayo, M., et al. (2000). *CAPIT: An Intelligent Tutoring System for Capitalisation and Punctuation.* In Proceedings of the International Workshop on Advanced Learning Technologies, Palmerston North, New Zealand
18. Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. *International journal of Artificial Intelligence in Education, 13*(2-4), 173-197.
19. Murray, T. (1998). Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design, *Journal of the Learning Sciences.*
20. Nkambou , R., & IsaBelle, C. (1998). Cyberphysique: A Web Based Distance Learning Environment. *Global Education on the Net, 1*, 252-256.
21. Salah,D., Zeid,A. (2009). Aaptive Patterns for Intelligent Tutoring Systems. In Proceedings of the EuroPLOP 2009.Germany.
22. Trætteberg, H., & Welie, M. (2000). *Interaction Patterns in User Interfaces.* In Proceedings of the 7th. Pattern Languages of Programs Conference Illinois, USA
23. Virvou, M., & Tsiriga, V. (2001). *Web Passive Voice Tutor: An Intelligent Computer Assisted Language Learning System over the WWW.* In Proceedings of the Proceedings of the IEEE International Conference on Advanced Learning Technologies. IEEE Computer Society. Retrieved.