Building Intelligent Tutoring Systems: An Overview

Chapter · September 2010
DOI: 10.1007/978-3-642-14363-2_18

3 authors, including:

Roger Nkambou
Université du Québec à Montréal
214 PUBLICATIONS **1,559** CITATIONS
SEE PROFILE

Valéry Psyché
Télé-université
20 PUBLICATIONS **127** CITATIONS
SEE PROFILE

Chapter 18

Building Intelligent Tutoring Systems: An Overview

Roger Nkambou,[1] Jacqueline Bourdeau[2] and Valéry Psyché[2]

[1]Université du Quebec à Montréal (Canada)

[2]Centre de recherche LICEF de la Télé-Université

Nkambou.roger@uqam.ca, jacqueline.bourdeau@licef.ca, valery.psyche@licef.ca

Abstract. This chapter addresses the challenge of building or authoring an Intelligent Tutoring System (ITS), along with the problems that have arisen and been dealt with, and the solutions that have been tested. We begin by clarifying what building an ITS entails, and then position today's systems in the overall historical context of ITS research. The chapter concludes with a series of open questions and an introduction to the other chapters in this part of the book.

18.1 Introduction

Intelligent Tutoring Systems (ITSs) are complex computer programs that manage various heterogeneous types of knowledge, ranging from domain to pedagogical knowledge. Building such a system is thus not an easy task. ITS authors need to be well equipped to face multiple issues related to their building process. In fact, the resources needed to build an ITS come from multiple research fields, including artificial intelligence, the cognitive sciences, education, human-computer interaction and software engineering. This multidisciplinary foundation makes the process of building an ITS a thoroughly challenging task, given that authors may have very different views of the targeted system. Some promote pedagogical accuracy (ensuring that tutoring decision making is based on sound pedagogical principles), while others focus on effective diagnosis of learners' errors (using appropriate knowledge structure and algorithms to interpret learners' decisions correctly). Murray (1999) identified seven different classes of tutoring system, each corresponding to a different author view, conditioned by the author's needs. Murray's study clearly shown that most of the existing authoring systems were designed for building part or all of a specific class of ITSs. Furthermore, there is a lack of methods and standard tools which could ease the authoring process.

Users interested in building ITSs fall into two groups: those with programming skills and those without. While the former can use snippets of code and class libra-

ries (API) without requiring an intuitive user interface, the latter have a need for tools that are easy to use, and which reflect their mental model of the artifact they are building. For example, a non-programmer should be able to integrate a content-planning module without having to program it, or add domain knowledge without having to understand either the knowledge model, the language used for coding this model, or the underlying logic. Two types of systems to help build ITSs were therefore identified (Murray 1999, 2003): shells for programmers and authoring tools for non-programmers. Both provide suitable resources to facilitate the building of ITSs.

In this chapter, the challenge of building or authoring an ITS is addressed, along with the problems that have arisen and been dealt with, and the solutions that have been tested. The chapter begins with a presentation of historical and current work on building ITS shells. Then, an update of Murray's review of authoring systems is given, with an emphasis on other important factors that should be considered in the classification of authoring tools. The chapter ends with a series of open questions and an introduction to the other chapters in this part of the book.

18.2 The Shell-Based Approach

The shell-based approach is well known in artificial intelligence. Since the beginning of expert systems research, there have been many proposals for shells to facilitate their building. A shell is a software development environment containing the basic components for building expert systems. The first experiment on such an approach was done with E-Mycin (Crawford 1987), a general purpose Expert System shell derived from Mycin. E-Mycin was built by removing all domain-dependent knowledge from Mycin, leaving only the inference mechanisms in the system. This allowed the use of these mechanisms with other domain knowledge. Thus, the shell-based approaches focus mainly on the system components but little on the user interface, making shell-based systems very suitable for users with programming skills.

Viewed as a knowledge-base system, an ITS contains general knowledge that governs decision-making in the expert, tutor and student modules. An interesting parallel can thus be drawn with the approach of classical expert systems. This is the analogy underlying a number of shells proposed to facilitate the construction of ITSs. While some of them include a very limited user interface, they are built to be used by ITS developers with some programming skills. They provide code libraries or conceptual frameworks for building parts of an ITS. Some of them focus on user modeling, while others place the emphasis on curriculum planning or content acquisition. As examples, Kobsa & Pohl (1995) developed a user modeling shell named BGP-MS that offers host applications methods for communicating observations regarding the user, and for obtaining information such as the user's presumed knowledge, beliefs and goals. Along these lines, Kay (1995) developed

the UM toolkit, a shell for building student models which enable reflection. The student can access his model built with this tool and find answers to questions such as: What does the system know about me? How did it reach these conclusions about me? Paiva and Self (1995) developed TAGUS, a shell for student modeling. TAGUS uses logic to represent the knowledge, reasoning and cognitive strategies of the learner. It provides an interface with services for accessing and updating information about the learner's knowledge state. More recently, Zapata-Rivera and Greer (2004) proposed SModel, a Bayesian student modeling server which provides several services to a group of agents in a CORBA platform.

Multiple other shell projects have been developed, focusing on particular ITS components. For instance, KEPLER (Vivet 1988), an expert system shell for building the domain and tutor modules of an intelligent tutoring system, was used to develop AMALIA, a tutoring system for teaching algebraic manipulations. The PIXIE shell (Sleeman 1987) was proposed to develop the diagnosis and remediation processes within an ITS. SCENT-3 (McCalla and Greer 1988) helped for fine-grain task sequencing. PEPE, a competence-based computational framework, was used for content planning (Wasson 1992).

ITS shells sometimes target the whole system (all components included). FITS (Ikeda and Mizoguchi 1994) is a good example of such a shell. FITS is a domain-independent framework that provides building blocks for student, tutor and domain modeling.

Recently, Stankov et al. (2008) developed a system called the Tutor–Expert System (Tex-Sys). Tex-Sys is an ITS shell that provides a generic module, implementing ITS components that can be used for the deployment of any given content. The content is specified in terms of user and domain knowledge databases. Two versions of Tex-Sys are provided, each dealing with an implementation approach taken by the ITS. DTex-Sys provides the client-server implementation, where the generic components (pedagogical controls) are implemented in a web server. The problem with this shell is that generic components deal only with pedagogical control of the ITS, not learner or domain control. The other version, xTex-Sys, provides another implementation based on a service-oriented architecture, where generic components (including learner and expert modules) are implemented as web services. The main drawback with the Tex-Sys approaches is that there is very little information about the shell's content. What knowledge is stored in it? How is it used? There is no answer to these questions.

A similar approach that targets all ITS components is the shell developed by Goodkovsky (1997). It provides simple component implementation (domain model, expert model, student model) as well as procedural models of the tutor's activity and the tutoring criteria and constraints. The usefulness of this shell can be questioned, however. For example, the domain model it provides is as simple as a set of domain concepts, which is a very limiting knowledge structure for a good tutoring system (see the chapter on domain modeling).

In summary, while the idea of providing ITS developers with a shell that targets the whole system is a very nice one, existing shells tend to focus on the big

picture, neglecting the detailed rationale for each component of the system. We believe that approaches that target particular ITS components are more profound. We also feel, given the complexity of the functions of an ITS, that further refinements must be made by considering the development of very specific shells that meet the special needs associated with certain complex and essential ITS functions, as has been done with PIXIE (Sleeman 1987). Thus, complex mechanisms such as cognitive diagnosis (Pelleu et al. 2007) may benefit from special attention that may result in generic implementations to be adapted in different problem-solving contexts. Such a shell would certainly be a welcome addition to the ITS-building toolbox, and would encourage the reuse of predefined small building blocks when developing new ITSs.

18.3 The Authoring Tools Approach: An Update on Murray's Review of ITS Authoring Tools

Authoring tools go beyond the simple shell by providing an additional user interface that allows non-programmers to formalize and visualize their knowledge. The goal is to increase both the accessibility and the affordability of authoring ITSs (Heffernan et al. 2006). After developing and demonstrating powerful systems, ITS research teams are prepared to simplify the ITS building process by developing higher-level tools which do not require programming skills and are therefore accessible to instructional designers and teachers. Decreasing the implementation costs by reducing the time/product ratio is another important target of authoring tools**.**

Murray, Blessing and Ainsworth (2003) edited a landmark book on the topic of ITS authoring tools. Murray (1999, 2003) has classified existing authoring tools under two categories: pedagogy-oriented and performance-oriented.

18.3.1 Pedagogy-oriented authoring tools

Pedagogy-oriented tools are those that focus on how to sequence and teach relatively canned content. REDEEM (Ainsworth et al., 2003) is an example of the tools in this category. It does not explicitly generate an instructional plan, but allows the production of a representation of instructional expertise, enabling the author to categorize the didactic material, or tutorial page, according to its level of difficulty, its generality and the prerequisites that connect it to other materials. This represents an implicit sequencing of content and learning activities, based on underlying tutoring strategies.

CREAM-Tools (Nkambou et al. 2003) is another example in this category. It provides operations for organizing content in terms of interconnected structures,

giving it the characteristics of a pedagogy-oriented authoring tool. Moreover, its organizing capabilities go beyond didactic material (which is what REDEEM is equipped to handle) allowing it to deal with both cognitive (organization of domain knowledge) and pedagogical aspects (organization of learning objectives).

Hypermedia tools such as Interbook (Brusilovsky et al. 1998) and MetaLinks (Murray 2003) also fall into this category. These systems manage the hyperlinks between units of content (both the form and the sequencing of the content). Hyperlinks provided to the learner can be intelligently filtered, sorted and annotated with respect to a model or a learner profile, sometimes based on an ad hoc ontology. The filtering of links can be based on prerequisites, cognitive load, appropriateness of the topic, difficulty, etc.

Other authoring tools which use pedagogy-oriented domain modeling are Eon (Murray 1998), IDE (Russell et al. 1988) and GTE (Van Marcke 1992). Specifically, GTE is a rule-based tool that performs actions according to a given pedagogical goal. Eon, a "one-size-fits-all" authoring tool which provides a full-fledged set of ITS tools, was initially implemented to perform activity streams based on a given instructional goal, in order to provide the author with multiple tutoring strategies. Finally, Eon uses an approach similar to REDEEM's parameterized one that allows the author to generate tutoring strategies from scratch. Several tutors have been implemented using Eon.

It should be noted that these tools are often based on a behavioristic-empiricist approach and tend to produce 'instructivist' tutors (Jonassen and Reeves 1996) with the possible exception of Eon (which is theory-independent). Also, their instructional strategies are fixed (predefined) and they usually do not have ontology-oriented representations.

18.3.2 Performance-oriented authoring tools

Performance-oriented tools are those that focus on providing a rich learning environment in which students can learn skills by practicing them and receiving feedback. RIDES (Munro et al. 1997) is an example of the authoring tools in this category. It is used for the construction of tutors that teach students how to operate devices through simulations. RIDES (for Rapid ITS Development Environment) generates instruction by providing tools for building graphical representations of a device and defining the device's behavior. In the past years, many RIDES tutors have been implemented. A system that adds capabilities to those of RIDES is DIAG (Towne 1997), a tool that simulates equipment faults and guides students through the process of diagnosing and repairing them. DIAG is concerned with the creation of domain knowledge and performs student error diagnosis by providing a mechanism that is applicable to many domains related to diagnosing equipment failure.

CREAM-Tools also belongs to this category, since it allows a connection between skills and the way they are acquired. For example, specific learning materials are linked to specific skills to support their learning. In this way, CREAM-Tools allows automatic generation of instruction and especially of complex learning materials that provide the student with a rich learning environment. When problems or exercises are created using CREAM-Tools, a knowledge structure for student tracking and error diagnosis during the problem-solving phase is also generated.

Other well-known systems in this category include SIMQUEST (Van Joolingen et al. 1997), Demonstr8 (Blessing 1997) and XAIDA (Hsieh et al. 1999). SIMQUEST provides tools for designing and creating dynamic and interactive simulation-based learning environments. Demonstr8 supports the development of model-tracing tutors by inducing production rules from examples. It addresses the ability of non-cognitive scientists to program a model-tracing tutor with limited training. With Demonstr8, an author has available three tools: a palette to create the student interface, a method for creating higher-order declarative representations of these student interface elements; a programming by demonstration method for creation of productions.

XAIDA (for Experimental Advanced Instructional Design Advisor) was originally designed to allow expert maintenance technicians to develop ITSs for maintenance topics. XAIDA relies on an instructional device known as a transaction shell, an instructional procedure applicable to particular instructional objectives of a specific type. XAIDA consists of four sub-tools (transaction shells), each of which uses a different scheme for representing and teaching a specific aspect of maintenance knowledge.

Another interesting system that may also be classified here is the Knowledge Construction Dialog (KCD) tool suite (Jordan et al. 2001), a set of tools that ease the implementation of natural language dialog capabilities within an ITS. The KCD suite was used in building important inputs for ATLAS's components: a plan operator library that is used by ATLAS's dialog manager and planner component (APE) and a semantic grammar for ATLAS's natural language understanding component (CARMEL). ATLAS is the natural language processing component that was used in several intelligent tutoring systems such as ATLAS-Why2 (VanLehn et al. 2002).

18.3.3 Instructional-Design-Oriented Authoring Tools

Some authoring tools have a specific focus on the instructional design (ID) and provide authors with relevant assistance in that process. Even though some of these were included in Murray’s classification, we believe they can be considered separately as ID-oriented systems.

Authoring systems in this category include Merrill's ISD-Expert (Merrill 1993), a system that provide rules to guide the ID process. The system suggests the best content structure (an organization of subject matter content) that is consistent with the instructional goals, subject matter knowledge and student profile. Expert CML (Jones and Wipond 1991) and IDE (Russell et al. 1988) are other ID-oriented systems. Using IDE, instructional designers can enter, edit and manipulate their instructional analyses and specifications in the form of complex networks of interrelated notecards (Pirolli and Russell 1991). Smarties (Hayashi et al. 2009) is one of the recent authoring systems within this category. It provides the user with a tool for building learning scenarios by utilizing explicit knowledge related to instructional design theories, which serves as the rational basis for decision making in this context. In the same family, a lighter tool is CIAO, a hypermedia ontology-based authoring assistant that was developed to assist authors of IMS-LD scenarios by providing them with theory-aware services (Bourdeau et al. 2007; Psyché et al. 2005). Another tool in this category is aLFanet (Santos et al. 2003) an authoring system which interprets an IMS-LD schema to develop a pedagogical scenario. It provides scenarios tailored to the particular interests, level of knowledge and experience of the learners.

In fact, an interesting particularity of these recent ID-oriented systems is that they tend more and more to incorporate emerging ID and eLearning standards such as IEEE-LOM (Learning Objects Metadata), SCORM (Sharable Content Object Reference Model), EML (Educational Modelling Language) and IMS-LD (IMS-Learning Design).

ID-oriented systems have a special focus on educational principles, giving first priority to instructional design in the ITS design project. In this perspective, ITSs are perceived as artifacts for the purpose of instruction (Pirolli and Russell, 1991). Thus, their pedagogical value, meaning how well they teach, is very important. Instructional design is a process that can guarantee this pedagogical value. The main question here is how this process can be supported in an ITS development project. We believe that an explicit, formal specification of a shared conceptualization of ID expertise and related theories could be a solution to that issue. Hence, ontology engineering can play a role in this context.

What would be efficient directions to take for the future in the ITS authoring process? It is worth repeating that ITS tutors are usually complex systems involving different dimensions. As in the shell-based approach, some ITS authoring tools focus on only a part of the system, while others consider all components of the system. Also, some are dedicated for tutors in specific learning domains, while others can be used for any domain. To produce generic (covering several areas) or complete tutors (implementing all ITS components) is a challenge. Therefore, when building an ITS, the following principles apply:

- First, approaches involving small building blocks should be preferred in order to reduce the time/product ratio;

- Secondly, increased assistance to authors should be a requirement, because the tools are still complex, although they are becoming more accessible as the teams develop higher-level tools;
- Finally, ontologies should play a role in formalizing the different types of expertise involved in ITS building.

In the ITS community, the trend currently observed is a segmentation: there are more and more specific foci of research on particular aspects of an ITS, such as open learner modeling or educational data mining. This trend increases the chance of seeing local standards emerging. However, this may not occur if knowledge and results are not shared across teams. In other words, those who share the same vision should be given opportunities for dialogue. Again, ontology engineering should help here by providing a framework for engineering ITSs, to facilitate interoperability and shareability between components.

18.4 Recent Approaches in Research and Development

This section highlights and discusses recent approaches in the building of ITSs. First, recent authoring tools are characterized; then, other types of software tools; finally, we discuss Woolf's framework for positioning building tools as they correspond to components and functions of an ITS.

18.4.1 Authoring Tools

Several ITS teams have been attempting to build authoring tools that would allow for sharing of components across ITSs and reduce development costs (Heffernan et al. 2006). The current R-D practice is to develop building tools that are paradigm-specific (Kodaganallur et al. 2005), such as CTAT for model-tracing systems (Aleven et al. 2006) and example-based systems (Aleven et al. 2009); ASPIRE for constraint-based systems (Mitrovic et al. 2009); TuTalk Tool Suite (Jordan et al. 2007) for dialogue-based learning agents; and authoring tools for inquiry-based systems (Murray et al. 2004; Gijlers et al. 2009) and for virtual reality and game-based ITS (Johnson and Valente 2008). Using a "backbone", teams built tools such as the Cognitive Model SDK (Blessing and Gilbert, 2008; Blessing et al. 2009), or developed tools that allow for derivation and variabilization, such as the ASSISTment Builder (Turner et al. 2005; Razzaq et al. 2009; – also see Chapter 18). In other words, recent developments in ITS show a "specialization" by paradigm, discipline of reference and privileged application domains, resulting in a similar specialization in the authoring tools that are derived from them. The paradigm for the CTAT tools and ASSISTment Builder is the ACT*

cognitive architecture; their domain of reference is cognitive psychology; the privileged component is the student model; and most of their applications are in math and science. ASPIRE authoring tool has constraint-based modeling as a paradigm, and computer science as a domain of reference. Discovery Learning Environments' paradigm is Discovery Learning; the domain of reference is the sciences, and the applications are mainly in science learning, etc.

Some new web paradigms are becoming good metaphors for collaborative authoring. For example, the open authoring model inspired by Wikipedia seems to be quite appropriate in the ITS context (Aleahmad et al. 2008).

18.4.2 General Software Engineering Tools

Besides the development of authoring tools, another way to facilitate the ITS building process is to view ITSs as software. As such, software engineering (SE) methods and tools can help. Tools provided in this context will be said to be SE-oriented. Various research proposals have been put forward with the aim of integrating different software engineering approaches. One group, the pattern-based approaches, are aimed at providing ITS developers with interesting patterns they can use to build the ITS. A pattern is a generalized solution of a typical problem within a typical context. A thorough analysis of existing ITS development solutions is an important stage in determining such patterns. Devedzic and Harrer (2005) discussed possible architectural patterns that can be found in existing ITSs. Harrer and Martens (2006) described the basis for a pattern language and catalogue for building ITS. Along the same lines, Salah and Zeid (2009) developed PLITS, a pattern language for ITS. PLITS was built from pattern mining by reverse-engineering many existing ITSs. As a proof of the concept, the authors used PLITS to build the Arabic Tutor, a web-based Intelligent Language Tutoring System for teaching a subset of the Arabic language.

Another alternative for building ITSs is to use a multiagent systems (MAS) approach for the basic building infrastructures. In fact, ITSs fulfill all of the conditions to be viewed as multiagent systems: 1) they are made of different interconnected, complex components; 2) they provide multiple, different and complementary services; 3) each of their components is functionally autonomous and equipped with specific knowledge structure and reasoning mechanisms. In this light, many ITSs have been built using agent and multiagent technologies. In particular, Vicari and Gluz (2007) have developed several ITSs to exemplify a set of Agent-Oriented Software Engineering (AOSE) methods derived from ITS research, which defines applicability criteria, design principles and implementation guidelines to be applied in the software analysis, design and development process . The agentification sometimes targets a specific ITS component: the tutor (Mengelle et al. 1998) or the learner (Vassileva et al. 2003). It can also target a specific ITS service (e.g., planning, dialogue management, collaboration) or the whole sys-

tem (Capuano et al. 2000; Hospers et al. 2003; Nkambou and Kabanza 2001). Even though there are many ITSs that were built using the agent and MAS approach, there is no agent-based framework specially dedicated to facilitating the ITS building process; rather, classic models and tools developed in the MAS community are used. FIPA specifications are well-known basic packages that can be used to support the building of agent-based ITSs. However, programming skills are required in order to use this building approach. Many agent- and MAS-oriented platforms (agent builders) such as JADE (Bellifemine et al. 2008) can be used to ease the development process.

18.4.3 A Framework for ITS Building Tools

Recently, a framework for organizing the necessary building blocks found in authoring systems for building ITS was proposed (Woolf 2008). Four layers were identified, each including specific classes of building blocks (Figure 18.1). The knowledge representation level includes tools for easily representing knowledge. At this level, the user should adopt the right formalism and select the right language or tool to ease the representation process. Level 2 is about the type of domain and student models, level 3 contains tools for implementing teaching knowledge while level 4 comprises those for communication knowledge.

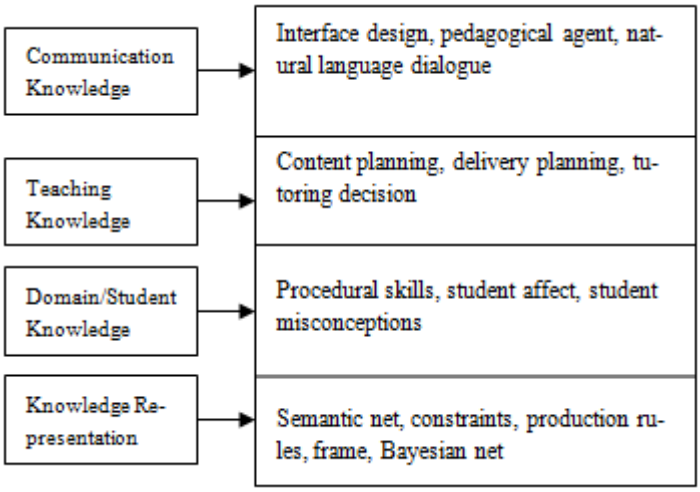

Figure 18.1 A framework of intelligent tutor building blocks (adapted from Woolf (2008))

Providing such a framework can be seen as a starting point for developing a real methodology for ITS engineering, where each step may provide guidelines that help the author make the best decision and select the relevant tools to produce

the output artifacts of that step. For instance, at level 1, based on some evaluation criteria, such as those proposed in Chapter 1 of this book for the evaluation of knowledge representation languages (expressiveness, inference power, etc), such guidelines may help the user select the right knowledge representation formalism in a given context.

The proposed framework sounds conceptually simple but may be very difficult to implement, due to the heterogeneous nature of the methods and tools that may be found at each level as well as the issue of tool integration at the same level or across levels. For example, how can a tool used for curriculum planning (at level 3) be plugged into a domain model built using CBM (at level 2)? So, even if a decision at the lower level becomes constraints for the upper level, a decision at the latter may still be incompatible with the one at the former. In short, such a framework cannot work without standards established in the community, and we are very far from that in the AIED community.

A simpler approach that should be investigated in the future is to ease the possibility of providing shareable, albeit non-standard, conceptualizations of ITS rationales. In other words, there should be a way for researchers who have the same conceptualization of some fundamental ITS concepts to share it. Fortunately, ontology provides a solution for this. It may allow small teams to clearly and formally define ITS artifacts as they conceive them, and then build their system on that formal conceptualization. As an example, a formal representation of an explicit conceptualization of instructional design and learning theories called OMNIBUS was developed by Bourdeau et al. (2007). OMNIBUS can be used by any ITS developer who shares that conceptualization to build his or her own system. The benefit of initiatives such as OMNIBUS is that they prepare a solid semantic ground on which different tools for building ITSs can be implemented. This shared semantic ground is a guarantee that can ease communication between the different tools. This approach will make it easier to move from the current proprietary, non-shareable solutions for building ITSs to others that are interoperable, reusable and easy to integrate.

18.5 Conclusion: Biodiversity or Tower of Babel? Future or Pipedream?

Multiple solutions have been provided for building ITSs, ranging from programmer-oriented tools to software for non-programmers. However, it is worth noting that there is no standard in the AIED research community to guide this process. In other words, ITS building cannot yet be considered an engineering process, as there are no methods and standard tools available to support it. As a result, after thirty years, existing solutions are still not widely shared in the field, making it difficult to find adequate building blocks and guidance to build an ITS. By comparison, the more recent research field of multi-agent systems is developing in a

community where many standard development principles, methods and tools for building MASs emerge. The lack of standards in the ITS community is probably due to the multidisciplinary nature of the field. There are multiple views of the target artifacts and services that an ITS should provide. In this light, an overall reflection on the problem of building ITSs leads us to raise the following questions:

1. Is the authoring bottleneck a 'natural' border to preserve the biodiversity of ITS species? Or is the adoption of standards a necessity for the survival of the species?
2. Is the idea of one-size-fits-all a pipedream, or is it truly the future of ITS research?

Our conception of the human brain may provide answers: in contrast to our former view of the brain as a set of regions with specific functions, our understanding now is that the brain works as a whole, and several regions can perform several functions, together or as substitutes for each other. Reconsidering the architecture of ITSs in that light may provide fruitful insights into how to build them.

The chapters in this part of the book provide the reader with two examples of authoring systems and an example of an ITS. Chapter 19 presents a thorough comparative analysis between CTAT, a well-known authoring tool for cognitive tutors, and ASTUS, a new cognitive tutor authoring tool. Through examples, the chapter addresses many limitations of CTAT and shows how ASTUS copes with these limitations. Chapter 20 is about ASSISTMENT, a suite of web-based tools that help researchers to easily design, build and then compare different ways of teaching students in order to improve their achievement. A randomized controlled experiment conducted using these tools is described. Chapter 21, the last in this part, presents ANDES, one of the most popular ITSs. ANDES is an intelligent homework helper for physics. That is, it replaces students' pencil and paper as they do problem-solving homework. The author presents ANDES' behavior, the development experience, evaluations of its pedagogical effectiveness and recent progress on dissemination/scale-up.

References

Ainsworth S, Major N, Grimshaw SK, Hayes M, Underwood JD, Williams B, Wood DJ (2003) REDEEM: Simple Intelligent Tutoring Systems From Usable Tools. In: Murray T, Blessing S, Ainsworth S (ed) Tools for Advanced Technology Learning Environments. Kluwer Academic Publishers, Amsterdam

Aleahmad T., Aleven V., Kraut R. (2008) Open Community Authoring of Targeted Worked Example Problems. Proc. of ITS 2008, 216-227.

Aleven V, McLaren BM, Sewall J, Koedinger KR (2009) A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. *International Journal of Artificial Intelligence in Education* 19:105-154

Aleven V, McLaren BM, Sewall J, Koedinger KR (2006) The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In: Ikeda M, Ashley KD, Chan TW (ed) Proceedings of the 8th International Conference on Intelligent Tutoring Systems. Springer Verlag, Berlin

Bellifemine F, Caire G, Poggi A, Rimassa G (2008) JADE: A software framework for developing multi-agent applications. Lessons learned Information & Software Technology 50(1-2): 10-21.

Blessing SB (1997) A Programming by demonstration authoring tool for model-tracing tutors. The International Journal for Artificial Intelligence in Education 8:233-261

Blessing S, Gilbert S (2008) Evaluating an Authoring Tool for Model-Tracing Intelligent Tutoring Systems. Proc. Of ITS 2008 204-215

Blessing SB, Gilbert SB, Ourada S, Ritter S (2009) Authoring Model-Tracing Cognitive Tutors. International Journal of Artificial Intelligence in Education 19:189-210

Bourdeau J., Mizoguchi R., Hayashi Y., Psyche V., & Nkambou R. (2007) When the Domain of the Ontology is Education. *Proc. of the 4th Conf. on Intelligent, Interactive Learning Objects Repository Networks (I2LOR'07)*

Brusilovsky P, Eklund J, Schwarz E (1998) Web-based education for all: A tool for developing adaptive courseware. Computer Networks and ISDN Systems 30(1-7):291-300

Capuano N, Marsella M, Salerno S (2000) ABITS: An Agent Based Intelligent Tutoring System for Distance Learning. Proceedings of the International Conference on Intelligent Tutoring Systems, Springer, Berlin

Crawford J (1987) EMYCIN : an expert system shell. Series Technical report,University of Sydney Basser Dept. of Computer Science

Devedzic V, Harrer A (2005) Software Patterns in ITS Architectures. International Journal of Artificial Intelligence in Education 15(2):63-94

Gijlers H, Saab N, Van Joolingen WR., De Jong T, Van Hout-Wolters B (2009) Interaction between tool and talk: How instruction and tools support consensus building in collaborative inquiry-learning environments. Journal of Computer Assisted Learning 25:252-267

Goodkovsky, VA (1997) Pop Class Intelligent Tutoring Systems: Shell, Toolkit & Design Technology. New Media and Telematic Technologies for Education in Eastern European Countries, Twente University Press, Enschede

Harrer A, Martens A (2006) Towards a Pattern Language for Intelligent Teaching and Training Systems. ITS 2008 - LNCS 4053, Springer, Berlin

Hayashi Y, Bourdeau J, Mizoguchi R (2009) Using Ontological Engineering to Organize Learning/Instructional Theories and Build a Theory-Aware Authoring System. International Journal of Artificial Intelligence in Education 19:211-252

Heffernan N, Turner T, Lourenco A, Macasek G, Nuzzo-Jones G, Koedinger K (2006) The ASSISTment Builder: Towards an Analysis of Cost-Effectiveness of ITS Creation. Proc. Of FLAIRS'06

Hsieh P, Halff H, Redfield C (1999) Four Easy Pieces: Development Systems for Knowledge-Based Generative Instruction. *International Journal of Artificial Intelligence in Education (IJAIED) 10:*1-45.

Hospers M, Kroezen A, Nijholt R, Heylen D (2003) Developing a generic agentbased intelligent tutoring system and applying it to nurse education. Proceedings of the IEEE International Conference on Advanced Language Technologies, Athens

Ikeda M, Mizoguchi R (1994) FITS: A Framework for ITS -- A Computational Model of Tutoring. J. of AI in Education 5(3):319-348

Johnson L, Valente A (2008) Collaborative Authoring of Serious Games for Language and Culture. Proc. Of SimTecT 2008

Jonassen DH, Reeves, TC (1996) Learning with technology: Using computers as cognitive tools. In: Jonassen DH (ed) Handbook of research for educational communications and technology, Macmillan, New York

Jones M, Wipond K (1991) Intelligent Environments for Curriculum and Course Development. In: Goodyear (ed) *Teaching Knowledge and Intelligent Tutoring, Ablex,* Norwood

Jordan P, Rose CP, Vanlehn K (2001) Tools for Authoring Tutorial Dialogue Knowledge. Proceedings of AI in Education

Jordan PW, Hall B, Ringenberg M, Cue Y, Rosé C (2007) Tools for Authoring a Dialogue Agent that Participates in Learning Studies. In: In Looi CK, McCalla G, Bredeweg, Breuker J (ed) Proceedings of the 12th Artificial Intelligence In Education, ISO Press, Amsterdam

Kay J (1995) The um toolkit for cooperative user modelling. User Modeling and User-Adapted Interaction 4(3):149-196

Kobsa A, Pohl W (1995) The user modelling shell system BGP-MS. User Modelling and User Adapted Interaction 4(2):59-106

Kodaganallur V, Rosenthal D, Weitz R (2005) A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms, IJAIED (15)

Koedinger K, Corbett A (2006) Cognitive Tutors. In: Sawyer K (ed) The Cambridge Handbook of the Learning Sciences, Cambridge University Press, Cambridge

McCalla G, Greer J (1988) Intelligent advising in problem solving domains: the SCENT-3 architecture. Proc. International Conference on Intelligent Tutoring Systems, Montreal, Canada

Mengelle T, de Léan C, Frasson C (1998) Teaching and Learning with Intelligent Agents: Actors. Proceedings of the International Conference on Intelligent Tutoring Systems (ITS'1998), Springer, Berlin

Merrill MD (1993) An Integrated Model for Automating Instructional Design and Delivery. In: Spector JM, Polson MC, Muraida DJ (eds) *Automating Instructional Design: Concepts and Issues*; ETD, Englewood Cliffs

Mitrovic A, Martin B, Suraweera P, Konstantin Z, Milik N, Holland J (2009) ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors. International Journal of Artificial Intelligence in Education 19:155-188

Munro A, Johnson MC, Pizzini QA, Surmon DS, Towne DM, Wogulis JL (1997) Authoring simulation-centered tutors with RIDES. *International Journal of Artificial Intelligence in Education* 8(3-4):284-316.

Murray T (1998) Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. Journal of the Learning Sciences 7(1):5-64

Murray T (1999) Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. Int'l J. Artificial Intelligence Education 10:98-129

Murray T (2003) An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. In: Murray T, Blessing S, Ainsworth SE (ed) Tools for Advanced Technology Learning Environments, Kluwer Academic Publishers, Amsterdam

Murray T, Blessing S, Ainsworth S (2003) Authoring Tools for Advanced Technology Learning Environment, Kluwer Academic Publishers, Amsterdam

Murray T, Woolf B, Marshall D (2004) Lessons Learned from Authoring for Inquiry Learning: A Tale of Authoring Tool Evolution. Proceedings of the International Conference on Intelligent Tutoring Systems (ITS'2004), Springer, Berlin

Nkambou R, Frasson C, Gauthier G (2003) CREAM-Tools: An Authoring Environment for Knowledge Engineering in Intelligent Tutoring Systems. In: Murray T, Blessing S, Ainsworth S (ed) Tools for Advanced Technology Learning Environments, Kluwer Academic Publishers, Amsterdam

Nkambou R, Kabanza F (2001) Designing Intelligent Tutoring Systems: A multiagent Planning Approach. ACM SIGCUE Outlook 27(2):46-60

Paiva A, Self J (1995) TAGUS - A User and Learner Modeling Workbench. User Modeling and User-Adapted Interaction 4(3):197-226

Pelleu J, Nkambou R, Bourdeau J (2007) Explicit Reflexion in Prolog-Tutor. International Journal of Artificial Intelligence in Education 17(2):169-215

Pirolli P, Russell DM (1991) Instructional design environment: Technology to support design problem solving. *Instructional Science* 19(2) :121-144

Psyché V, Bourdeau J, Nkambou R, Mizoguchi R (2005) Making Learning Design Standards Work with an Ontology of Educational Theories. Frontiers in Artificial Intelligence and Applications 125: 539-546

Razzaq L, Patvarczki J, Almeida S, Vartak M, Feng M, Heffernan N, Koedinger K (2009) The ASSISTment Builder: Supporting the Life Cycle of Tutoring System Creation. IEEE Transaction on Learning Technologies 2(2):157-166

Russell D, Moran T, Jordan D (1988). The Instructional Design Environment. In: Psotka J, Massey LD, Mutter SA (ed) *IntelligentTutoring Systems, Lessons Learned*, Hillsdale, Lawrence Erlbaum

Salah D, Zeid A (2009) PLITS:A Pattern Language for Intelligent Tutoring Systems. Proceedings of the 15th European Conference on Pattern Languauges of Programs

Santos OC, Boticario JG, Koper EJR (2003). *aLFanet.* Paper presented at the m-ICTE 2003, Badajoz (Spain)

Sleeman D (1987) Pixie: a shell for developing intelligent tutoring systems. Artificial Intelligence and Education 1:239–263

Stankov S, Rosic M, Zitko B, Grubisic A (2008) TEx-Sys model for building intelligent tutoring systems. Computers & Education 51 (3):1017-1036

Towne DM (1997) Approximate reasoning techniques for intelligent diagnostic instruction. *International Journal of Artificial Intelligence in Education* 8(3-4):262-283

Turner TE, Macasek MA, Nuzzo-Jones G, Heffernan NT, Koedinger K (2005) The Assistment Builder: A Rapid Development Tool for ITS. In: Looi CK, McCalla G, Bredeweg, Breuker J (eds) Proceedings of the 12th Artificial Intelligence In Education, ISO Press, Amsterdam

Van Joolingen WR, King S, de Jong T (1997) The SimQuest authoring system for simulation-based discovery learning. In: du Boulay B, Mizoguchi R (ed) Artificial intelligence and education: Knowledge and media in learning systems, Amsterdam, IOS Press

Van Marcke K (1992) Instructional Expertise. In: Frasson C, Gauthier G, McCalla GI (ed) Procs. of Intelligent Tutoring Systems 1992, Springer-Verlag, New York

VanLehn K, Jordan P, Rosé CP, et al. (2002) The architecture of Why2-Atlas: A coach for qualitative physics essay writing. In: Cerri SA, Gouarderes G, Paraguacu F (ed) Intelligent Tutoring Systems: 6th International Conference, Springer, Berlin

Vassileva J, McCalla GI, Greer JE (2003) Multi-Agent Multi-User Modeling in I-Help. User Model. User-Adapt. Interact 13(1-2):179-210

Vicari RM, Gluz JG (2007) An Intelligent Tutoring System (ITS) view on AOSE. *International Journal of Agent-Oriented Software Engineering, 1*(3-4):295-333

Vivet M (1988) Knowledge based tutors: towards the design of a shell. International Journal of Educational Research 12(8):839-850

Wasson B (1992) PEPE: A computational framework for a content planner. In: Dijkstra SA, Krammer HPM, van Merrienboer JJG (ed) Instructional Models in Computer-Based Learning Environments. NATO ASI Series F 104:(153–170)

Woolf B (2008) *Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, New York

Zapata-Rivera JD, Greer J (2004) Inspectable Bayesian student modelling servers in multi-agent tutoring systems. *International Journal of Human-Computer Studies archive* 61(4):535–563