A NEW APPROACH TO ITS-CURRICULUM AND COURSE AUTHORING: THE AUTHORING ENVIRONMENT

R. Nkambou\textsuperscript{1}, C. Frasson\textsuperscript{2}, G. Gauthier\textsuperscript{3}

\textsuperscript{1}University of Moncton, Department of computer science
Moncton (New-Brunswick), E1A 3E9, Canada
E-mail: nkambor@umoncton.ca
Phone: (1)(506)858-4939
Fax: (1)(506)858-4541

\textsuperscript{2}University of Montreal, Department of Computer Science and Operational research
PO.BOX. 6128, succ. Centre-ville, Montreal (Quebec), H3C 3J7, Canada
E-mail: frasson@iro.umontreal.ca

\textsuperscript{3}University of Quebec at Montreal, Department of Computer Science
PO.BOX. 8888, succ. Centre-ville, Montreal (Quebec), H3C 3P8, Canada
E-mail: gauthier.gilles@uqam.ca

Abstract. The aim of this paper is to present an authoring environment dedicated to the development of curriculum and course that can be handled by Intelligent Tutoring System (ITS). The adopted curriculum model is CREAM (Curriculum Representation and Acquisition Model) which represents subject-matter content from the domain, the pedagogical and the didactic points of view. The authoring environment, called CREAM-Tools, consists of a set of tools able to create and edit all of the elements (domain knowledge, instructional objective, didactic resources, pedagogical and didactic model) of a curriculum. This set of tools contains, among others, graphical editors, intelligent assistance and knowledge browsers. The environment also includes a course generation kit that allows a designer to generate a course by specifying a target public and a course description. This paper also presents curriculum and course development approaches (methodologies). The delivery of curriculums and courses produced from this environment is done by an object-oriented ITS. The teaching process is supported by new technologies such as Word-Wide Web browsers. We also show ways in which other modules in an intelligent tutoring system can exploit the resulting curriculums and courses in the teaching-learning process.
INTRODUCTION

THE INSTRUCTIONAL DESIGN PROBLEM

Many sources provide guidelines for instructional design [1, 2, 3]. Special expertise is involved in the instructional design process (design methodology, learning principles, ...). Including such expertise in an authoring system would be very useful for the users (useful recommendations and error detection), however, existing authoring systems fail to provide needed guidance such as:

- What is mandatory when defining a concept?
- What type of resource should be used. For instance: when should we use multiple choices, and when are short answers more appropriate?

Merrill [4], through ISD Expert, was one of the first to experiment the use of an expert system in the instructional design context. The purpose of ISD Expert is to guide instructional design decisions so that resulting products can more adequately implement the precepts of learning and instructional design. However, ISD Expert is not an authoring system. It does not include editors or other tools for developing computer-based learning materials.

Expert systems have proved helpful in advising and assisting users [5]. Where we lack instructional design experts, expert systems can be useful. In order to provide users with support that focuses on the expertise for building curriculums and courses, we propose to use an expert-based assistant integrated with the authoring environment. This expert-based assistant has a knowledge base that is editable and constraint-based. The goal of this system is to assist novice and intermediate instructional designers in the curriculum and course building process. For instance, verifying the suitability of the instructional objectives that the designer has created, advising him about what type of learning material might be effective in achieving a particular instructional objective (or to acquire a particular skill). This assistance is accomplished by integrating an expert system with our authoring environment. The expert system validates curriculums and courses that have been produced with the authoring tool. The validation technique uses strategies from learning and teaching theories. These strategies take advantage of knowledge models for curriculum and courses which have been used in both intelligent tutoring systems and authoring tools as reported in [6].

Our authoring environment includes an expert system for validating the curriculum. The goal of this validation system is to verify the consistency of the curriculum being built. The integration of the
validation system with the authoring tool can take two forms: on-the-fly validation, in which the validation system acts as a coach or an advisor that reacts according to the user’s actions, and post-construction validation in which the system acts as a critiquing system [7]. We also try to ensure that the validation system’s intervention does not disturb the instructional designer’s progress (that it will limit itself to what should be said and not more). To insure this kind of assistance, we introduce different intervention levels.

**INSTRUCTIONAL DESIGN THEORY AND AUTHORING SYSTEMS**

Industrial training is both time consuming and expensive. For instance, IBM US spends $2 billion a year on training including $1 billion for trainers’ salaries; Each year, the US government spends $20 billion on military training [15]. Part of this training cost is linked to instructional design and to the teaching process. Thus, a system that permits the user to build a course rapidly and according to specific needs could contribute to the reduction of these costs. Research has been made to ease this process. However, the course building process still remains an explicit task of the instructional designer. Our authoring environment uses design expertise to automatically generate a course from a set of parameters specified by the designer. The main ones are the expected knowledge to be acquired by the course and the target public which will be taking it. The course can be adapted by the designer who can also test it with a simulated student. The final course is an object that can be used directly by an intelligent tutoring system or by a human teacher.

The primary purpose of the instructional design process is to structure the learning environment so as to provide the learner with conditions that will support the learning process. However, this process is not simple. For instance, courseware design and development are time-consuming, tedious and somewhat repetitive. There is material to create, there are lesson plans to develop, and so on. Thus, the automation of the instructional design process [3] might make courseware production more cost-effective, and simpler. There exist authoring systems that serve that purpose [8]. These systems offer a set of tools that are used in the instructional design process. We have developed such an environment [9] for curriculum and course building. This environment includes design tools such as knowledge browsers, graphical editors and visualization and simulation tools. These tools are seamlessly integrated in the authoring environment.

Although such an environment exists, there remain several key difficulties:
• the design process is difficult for humans,
• it requires difficult-to-acquire expertise,
• human actions introduce unacceptable errors,
• there is a lack of human experts.

It is to solve these problems that traditional authoring systems (that were nothing more than tool sets) have evolved towards more intelligent systems which try to incorporate instructional design expertise. Systems such as ID Expert [4, 10, 11], IDE [12], ISD Expert [13], the GAIDA project [14] and AGD [8] have been specified in this manner. The need is to incorporate instructional design expertise with regards to the selection, sequencing and presentation of materials in support of various lesson objectives and subject-matter domains. Our system covers the selection and sequencing aspect, that is, the curriculum and course design.

Several approaches to this integration of the expertise in an authoring system were proposed. For example, Gagné [14] argues for automating only the guidance framework. The advantage of this sort of automation is that it is relatively inexpensive. Merrill’s proposal incorporates instructional transactions, that are intelligent object-oriented frameworks for creating specific kinds of instructional interactions, in a computer-based environment. His system contains rules for prescribing instructional strategies, which in turn will prescribe transactions. Tennyson [13] proposed the incorporation of a user model.

Our approach is based on encapsulating the curriculum and course building expertise in an expert system that is integrated with the authoring environment. This expert system reasons on a constraint base that contains constraints on curriculum and course design. These constraints come from different instructional design theories [1, 2, 4]. In order to build such a system, we use a subject-matter (curriculum) model called CREAM\(^8\) [6]. In this model, a curriculum is represented and organized according to three points of view: the domain knowledge (through the capability model (CREAM-C)), the pedagogical aspect (through the objective model (CREAM-O)) and the didactic point of view (through the resource model (CREAM-R)). These three models combine to form a transition network

\(^8\) Curriculum REpresentation and Acquisition Model
structure called CKTN (Curriculum Knowledge Transition Network) that represent the pedagogical and didactic model of the curriculum.

After a brief presentation of CREAM approach, the curriculum and course conception environments are presented. For each of them, the associated approaches are defined. We finally show how other ITS modules can exploit the resulting curriculum and courses in a teaching/learning process.

**CREAM: A CURRICULUM REPRESENTATION MODEL**

We consider the curriculum in ITS as a structured representation of the subject matter in terms of capabilities [16], instructional objectives and pedagogical resources (learning materials). Achievement of instructional objectives contributes to the acquisition of capabilities. This achievement is supported by pedagogical resources through learning activities (exercises, demonstrations, problems, simulations...). CREAM implements domain, pedagogical and didactic aspects of a subject matter through a network organization of capabilities, of instructional objectives defined on these capabilities and of pedagogical resources supporting the accomplishment of instructional objectives. Using these three knowledge structures, we construct a curriculum knowledge transition model which contains particulars links between their elements.

**THE CAPABILITY MODEL**

A capability is a knowledge (or cognitive) unit stored in a person’s long term memory that allows him to succeed in the realization of physical, intellectual or professional activity [16]. When we want to represent this kind of curricular knowledge, we ask ourselves what must be taught, that is the content we want the student to acquire. Three categories of capabilities are set up in CREAM: verbal information, intellectual skills (discriminations, concepts and rules) and cognitive strategies. The capability model (CREAM-C) is a multi-graph where nodes are capabilities (each type of node denotes a capability category) and links among them can be of several types: analogy, generalization-specialization, abstraction, aggregation, deviation.

**INSTRUCTIONAL-OBJECTIVE MODEL**

An instructional objective is the description of the behavior (or performance) that the student must demonstrate following a learning process. Several studies have shown the necessity of specifying
objectives in teaching systems [17, 18]. We have taken this into account in our representation approach by introducing an instructional objective model (CREAM-O) in which instructional objectives are represented and organized together by links of several types: prerequisite, complementary or pretext. These links can wither be defined by the instructional designer or be generated by the curricular system by reasoning on its knowledge structures [4].

LEARNING RESOURCE MODELLING
Pedagogical resources are the means used by the teaching system to support the teaching/learning process. Several categories of resources are considered in CREAM: Those that support the teaching/learning process for the acquisition of knowledge (problems, demonstrations, exercises, hypermedia document...), the expert type resources that act as experts that can intervene in the teaching/learning process to help or critic the student in a specific activity (advisor, critiquing systems, coach) and physical and media resources that represent basic teaching material (simulators, video, sound, pictures...). We have identified several types of links between resources: equivalence, abstraction, case of, use of, and auxiliary. The network defined by these relations represents the resource model (CREAM-R).

CKTN: COUPLING THE PRECEDING MODELS
The coupling of the preceding models creates special links between their elements. In the space resulting from this coupling, we group each objective together with its associated resources in an element we call a transition. The input capabilities of a transition are those that are prerequisites to the achievement of the objective involved in this transition. The output capabilities of a transition are those that are produced by the achievement of the objective involved in the transition. Thus, they produce two types of links between capabilities and objectives: prerequisites and contribution links. The resulting network is called CKTN (Curriculum Knowledge Transition Network). Figure 1 shows part of CKTN in the Baxter pump manipulation domain. Note that, for the generation process, we will only consider the objectives included in the transition network. The transitions themselves are taken into account during the actual teaching process.
A prerequisite link from a capability C to an objective O expresses the fact that C is a precondition to the realization of O. A prerequisite link is characterized by its nature (mandatory, optional) and by the minimum mastery level required on the source capability to be able to consider it sufficient for overstepping the link (entry level) and thus be able to eventually achieve the objective. This entry level is specified by using a qualifying value taken from an evaluation (or acquisition) vocabulary. For example, in Klausmeier’s vocabulary [19], a capability of type “concept” can be recognized, identified, classified or generalized (4 acquisition levels) whereas a capability “rule” can be applied or transferred (only 2). Evaluation vocabularies are denoted in our system by an ordered set of integers representing different levels of acquisition. Since several vocabularies exist for describing the same type of capability, the designer must choose a vocabulary before proceeding to the construction.

A contribution link qualifies the way in which the realization of an objective contributes to the acquisition of a capability: It can be a strong, moderate or weak contribution. Therefore, several objectives can contribute to the acquisition of one capability.

In the following sections, we will show how the CREAM approach supports course generation.
METHODOLOGY OF CURRICULUM CONSTRUCTION

In order to provide the designer with the capability to define, create and organize the curriculum using a methodology, we have retained three approaches for the construction of a curriculum: **content-driven, course-driven** and **material-driven**.

**CONTENT-DRIVEN APPROACH**

Also called approach by competence, it allows the user to build a curriculum by first specifying the content (of the domain). This phase is followed by the definition of teaching objectives and the specification of resources to achieve these objectives (figure 2). The main advantage of this approach is that it leads to a curriculum with a “strong content” permitting the construction of several courses and pedagogical views. However, this approach is costly in development time for a large domain as we are sometimes constructing needless details for a given course.

---

**Stage** | **Output**
---|---
Content determination and specification | Capability model
Determination and specification of instructional objectives | Objective model
Determination and specification of didactic resources | Resource model
Building of the CKTN (Pedagogical model) | Pedagogical model

Fig. 2: content-driven specification stages
COURSE-DRIVEN APPROACH

In this approach, the curriculum construction begins with a course already in the designer’s mind. This means that the general instructional objectives are known. Their refinement constitutes the main part of curriculum objectives specification stage. They will in turn determine the content (knowledge) to include in the subject matter. These specifications lead to low-level objectives permitting the user to figure out knowledge (capabilities) involved in the subject matter. We then reach the stage of resource specifications aimed at defining the material to be used.

![Diagram of the course-driven approach]

Fig. 3: The course-driven approach

The advantage here is that this approach leads to a good organization of instructional objectives, but could lead to “poor content”.

MATERIAL-DRIVEN APPROACH

The third approach we consider (but that has not been experimented) is the material-driven approach. It consists in identifying and collecting pedagogical materials and defining instructional objectives that
could be covered by a set of materials. However this approach does not seem very efficient because we could obtain an amount of resources not really pertinent to a specific course. Therefore, curriculum development remains an iterative process. For instance, when the subject-matter-content phase precedes the instructional-objective-specification step, new capabilities could be identified at this latter stage and integrated to the subject-matter content. From our own curriculum development experience, we found that the first approach is very adapted to small domains but that the second approach is more efficient for large domains.

**GENERATING A COURSE FROM THE CURRICULUM**

The CREAM model can serve many educational purposes and can be exploited using the set of tools we have developed. In this section, we describe the course development approach which allows the designer to create a course given some parameters. The course generation process starts with a specification of training requirements in terms of goals or knowledge to be acquired. The system performs a reasoning on the corresponding curriculum-knowledge structures to obtain a graph, in terms of objectives to be realized, of the course that will satisfy the specified training requirements.

**THE COURSE CONCEPT**

In the educational technology field, a course is a sequence of instruction periods dealing with a particular subject and aimed at the evolution of a student's knowledge. In our model, we defined a course as a structured set containing three categories of objectives: global, specific and terminal objectives. A *global objective* being a statement expressed by the teacher to globally describe all of the lasting changes (cognitive, affective and psychomotor) that he wishes to induce in his students’ behavior during a course; A *specific objective* describes a set of behaviors that the learner should be able to demonstrate (specified in terms of the capabilities the learner should acquire); A *terminal* (or operational) *objective* is, in our context, an objective that describes a precise performance the student should achieve. In general, a specific objective is composed of several terminal objectives. This set of objectives is centered on a well-defined educational purpose and can tackle various themes around it. By grouping different parts that became apparent after our study of the different representations of the course concept [20, 18, 29], we find that a course is composed of three main parts:
In our system, a course does not consist of only one type of activity to accomplish but of a variety of them (going from the teaching of verbal information, through the presentation of a content, to complete problem solving activities). Each course comprises several kinds of pedagogical resources, connected to the course objectives and essential for their achievement. Their aim is to permit the learner to eventually acquire the capabilities attached to these objectives.

COURSE GENERATION PROBLEM

Up until now, course creation, even on a model such as CREAM, has always been done manually. The instructional designer had to determine and choose the course objectives himself, according to what he wanted to instill in his students. He then had to structure them, decide how he could make his students realize them, etc... All this could be very long and tedious and could result in a badly designed course if carried out too fast or carelessly.

The generative aspect of our method transforms completely the role of the designer. He now only has to specify the knowledge he wants to teach, supervise the course generation process and approve or not the resulting course. If one aspect of the course does not satisfy him, our toolkit offers him all the means to help him modify the original curriculum, change the initial knowledge specification for his course and even edit the course. This last thing is done by changing either the course structure or some elements of it (for example, adding, removing or modifying objectives or resources). Therefore, he is free to accept or not the generated course: he is the final judge in the process. Automatic course generation reduces the volume of his task and might then create more accurate and refined instructional material in addition to being less time-consuming and more cost-effective.

To generate a course is thus to allow an instructional designer to orient his efforts towards a different aspect of course construction than the one concerned with the objectives to be chosen for the course. He can now concentrate on several aspects such as: the course structure, the pedagogical resources which will help realize the objectives, the ways to teach the different kinds of knowledge involved
(tutorial strategies), the definition of the various course themes and sub-themes and the objectives to be associated with them, the relevance of the course or some of its objectives, etc. In fact, the instructional designer can concentrate on the more declarative aspects of the course. In addition, he can, with our system, evaluate his courses by using the student interface at his disposal to simulate the evolution of a course.

In this paper, we want to bring forth a solution to precisely the problem of reducing the amount of time devoted to the conception of a course, by introducing an automatic course generation process and by proposing a system that performs this generation. This reduction will thus lead to lower training costs, an appreciable factor in industry.

Our approach is thus a key point in educational and industrial matter and is actually part of a large system which permits the specification, generation, edition, development and supervision of an entire course.

**Automatic generation of a course graph**

We have studied the possibility of generating a course from one or several curricula built according to the CREAM approach. Several parameters have been considered:

- a final state corresponding to the goals of the course to be generated. This state can be expressed either as a set of objectives that the course should reach, or as set of capabilities to be acquired by the student. We use the term Knowledge To be Acquired (KT) to designate this set.
- a starting point corresponding in reality to the knowledge state of the targeted public. We represent this state as a vector whose elements are couples \((\text{capacity name}, \text{level of mastery})\). For example, \((\text{front panel}[\text{concept}], \text{identifies})\) is an element of the vector representing the knowledge level of intermediate-level nurses on the Baxter pump. We think that a good cognitive analysis of a domain can allow the identification and the classification of students categories with a general idea on their knowledge. The resulting course will be adjusted and adapted by the planner to a particular student by taking into account the interaction between the student and the ITS.

The process of course generation [21] takes into account these two parameters and uses the structures of the curriculums to lead to a reasoning that will allow to define a relevant course that would allow the public targets to achieve the specified learning objectives (figure 4).
**Student target group**: We define a target group as a group of students' state of knowledge of various capabilities which may be part of several subject matters (curriculums). For instance, the knowledge of a novice nurse on the handling of the Baxter pump will not be the same as that of an advanced nurse. Therefore, a course on this topic should not include the same objectives for the former as for the latter. The advanced nurse would waste her time learning things that she already knows. Thus, these two groups of nurses constitute two different student target groups and the generator should build a course well-suited to each one.

![Fig. 4: Process of course generation](image)

To determine these target publics and which capabilities they must include, the instructional designer must perform a good cognitive task analysis in order to differentiate and classify the students and their expected knowledge. If the designer has no idea about the state of knowledge of his students, he can generate a course with no target group. He will then be free to readjust the resulting course for his needs and purposes at a later stage.

**DynCKTN**: Since the generation is performed by going through the CKTN graph defined on the chosen curriculum, we thus have to assign the target group state of knowledge to the capabilities and the links which constitute the CKTN to make it **dynamic**. The resulting graph is called a DynCKTN.
and we name this operation the **marking** of the CKTN. More precisely, it consists in attributing to each prerequisite link a value in \{acquired, partially acquired, not acquired\} indicating whether the minimum acquisition level on this link has been reached according to the target group, and to each capability a value in \{possessed, partially possessed, not possessed\} representing the acquisition standard of our target public on this knowledge and calculated from the levels assigned to the links.

Figure 5 shows some of the rules used to calculate this marking.

```
Marking of a prerequisite link
For a capability C prerequisite to an objective O,
  IF TargetGroupeLevel(C) >= EntryLevel(C, O)
    THEN link (C,O) is acquired
  IF TargetGroupeLevel(C) = NIL
    THEN link (C,O) is not acquired
  ELSE link (C,O) is partially acquired

Marking of a capability
For a capability C prerequisite to objectives O1, O2, ..., On,
  IF links(C,O1),(C, O2),...,,(C,On) are acquired
    THEN C is possessed
  IF links(C,O1),(C, O2),...,,(C,On) are not acquired
    THEN C is not possessed
  ELSE C is partially possessed
```

Consequently, a generation without target public leads automatically to the marking of all the CKTN-prerequisites links as *not acquired* and therefore of all capabilities involved in the current CKTN as *not possessed*.

Example: Considering the CKTN in figure 1, we want to generate a course on the manipulation of the Baxter pump for intermediate nurses. Thus, a part of the target group can be described as: [(k1,2), (k2,1), (k3,2), (k4-k9,0), (k10,1), (k11,0), (k12,1), (k13,3), (k14-k17,0)]. The marking process produces the DynCKTN of figure 6.

**Heuristics**: We now have a dynamic CKTN based on which we have to reason in order to generate a course. We traverse it to determine which objectives have to be included in the course in order to permit the acquisition of the knowledge specified in the KT. To do this, we perform from each capability to acquire, a backward chaining traversal of the sub-graph rooted at the capability in order to choose the objectives we judge necessary for the acquisition of that capability. We first evaluate the immediate prerequisite objectives of the capability and then, since some objectives possess mandatory
prerequisite knowledge which in turn has contributing objectives, we have to trace the subgraph back until we reach an objective without any prerequisite or a capability already mastered by the student (as specified in the target group or seen by the marking).

![DynCKTN of the Baxter pump manipulation curriculum](image)

Fig. 6: Part of the DynCKTN of the Baxter pump manipulation curriculum

The choice of objectives is carried out by applying heuristic rules introduced into the system and which consider several parameters: the links between capabilities and objectives (prerequisite or contribution), the knowledge in the KT and also the DynCKTN. We defined (and actually implemented) three possible heuristics:

- **General inclusion heuristic** which consists in including all the sub-graph objectives in the course. This simple heuristic can lead to unnecessarily overloaded courses while it is possible that a course with fewer objectives could bring a learner to the acquisition of the same knowledge. On the other hand, it could leave more freedom to the designer for his personal objectives’ choices among those selected by the generator and thus lead to a course with several possible plans. For instance, he could find some objectives unnecessary and take them out of the generated course (while ensuring that the entire KT could still be acquired).

- **Random choice heuristic** which consists in choosing a certain number of objectives according to their contribution value to the capability we want to obtain. Thus, if an objective O contributes strongly to a capability C, it is sufficient to its eventual acquisition so we only select it and continue the generation in the sub-graph rooted at O. If there is no objective of strong contribution, we have to pick, for instance, two medium-valued contributing objectives or three weak ones. In addition, we must also consider the marking of the capability because a partially possessed capability requires a lower objectives’ contribution for its acquisition than a non possessed one. So we take
this last parameter into account when setting the *contribution quota* needed for the acquisition of a capability.

- **Evaluation of complexity heuristic** consists in evaluating the complexity of the sub-graphs rooted at each of the objectives we are considering. It chooses the objective with the highest probability of acquiring the considered capability. The complexity is expressed in terms of weights assigned to each capability of the sub-graph. To calculate this weighted sub-graph, we take into account the marking of capabilities, the value and marking of the prerequisite links and the contribution links value. For instance, a *possessed mandatory* capability will more likely permit the realization of an objective than a *non possessed optional* one, and thus will weigh more. By adding up all the weights in each sub-graph, we can evaluate each of their “global acquisition level“ and make a more “intelligent“ objective choice.

We have implemented each of these heuristics and generated courses by using each of them. It is worth noting that the system permits an easy implementation of new heuristics since the heuristic is part of the input given to the generation subsystem. The designer can also describe his own heuristic and ask it to be implemented, so as to generate his courses in a more personal way; The constraints on an heuristic description being only that it must receive a capability as input, reason on a CKTN structure and output a set of objectives. The obtained set of relevant objectives will be put in form of a course graph by considering links defined in the objective models of current curriculums.

**STRUCTURING COURSE**

Once the graph of the course is generated, we continue with a phase of structuring. The structuring process determines categories of the different objectives of the course. Global objectives, terminals, and intermediate are then determined. A visualization of the structure can be made. This visualization shows a tree-like structure composed by learning hierarchies.

Now that once the graph of course and the structure of the courses are built, the last step is when designer defines the different themes and sub-themes of the course and their associated objectives. The resulting course is directly usable in a learning or teaching in the context of an ITS [22].

**BUILDING ENVIRONMENT**

We have developed a toolkit (CREAM-Tools) for supporting designers in the curriculum and course building processes. CREAM-Tools possesses a variety of tools for curriculum and course design
including a course generating system, graphical editors, knowledge browsers, visualization and simulation tools, intelligent supporting tools and multimedia authoring tools. It has been developed under VisualWorks 2.0. (Smalltalk) and constitutes a complete authoring system

**Curriculum Building toolkit**

The goal of this workshop (figure 1) is to enable the designer to define, create and organize curriculum component objects. Therefore, the result is a curriculum object offering an interface accessible to other ITS components that exploits it for several purposes: instructional planning, student modeling, learning process, ... The curriculum environment included two types of tools: Building tools and intelligent support tools.

![Curriculum Building Environment](image)

**Fig. 7:** An overview of the curriculum building environment

**Building toolkit**

They are essentially *browsers*, graphical editors and multimedia edition tools. They allow the designer to make up curriculum objects (building of capability-model objects, objective-model objects
and learning-material objects; definition of objectives and capabilities, creation of learning material). Learning materials are created by using special tools intended for this purpose (demonstration, problem, and multimedia editors [23]) (figure 8a).

Learning materials are stored in a database reachable during the authoring process. The designer can test a learning material to verify if it corresponds to his need. CREAM-Tools also permits the creation of resources that can be used through a WWW client. For example, to create a multiple-choice question, an interface allows the designer to specify attributes (question and possible answers) (figure 8b). Our
system generates a test object from this specification and stores in the curriculum model. At the time of learning process, the tutor can decide to activate a test object just by sending the displaying message to it. The displaying message of a WWW resource converts it as a HTML object and displays it in the current WWW client (figure 8c) [31].

**Intelligent assistant tools**

Two intelligent tools are available in the curriculum building environment. One concerns the automatic generation in the curriculum (cf. 3.1.3), and the other one is an expert system for curriculum validation [24]. The second one (figure 9) contains a knowledge base with about sixty rules extracted from instructional design theory [1, 2, 14, 11, 13]. These rules support advising during the design process, or critiquing of the designer’s work. Therefore, the system is able to give gradual hints (on request) at any moment of the building process. For example, a hint could be an error signal or a suggestion. In critical mode, there is an intervention of the system only after a user’s request. This intervention is formulated as general comments.

![Fig. 9: Initialization of curriculum building coach and critiquing systems](a) ![Fig. 9: Initialization of curriculum building coach and critiquing systems](b)

**SOME GENERATIVE ISSUES**

We have examined some generative aspects of the design of a curriculum, using the CREAM approach:
• **Automatic generation of an objective model isomorphic to the capability model.** Several classifications of domain knowledge exist [1, 11, 25, 26, 27]. In CREAM we use Gagné’s classification and we establish a mapping between capabilities and abilities involved in instructional objectives. Theses abilities are defined using Bloom’s taxonomy [28]. When the capabilities model is defined, we can generate an objective associated with each capability in this model. For example, we derive the objective `define(infusion-concept)` from the capability named `infusion(defined-concept)`. Links between objectives are derived from links between capabilities. Thus, aggregation links in the capability model will be interpreted as sub-objectives in the objective model.

• **Automatic generation of prerequisites links between objectives.** An interesting feature of our system is its ability to reason on a given CKTN. By looking at prerequisite and contribution links it automatically derives new prerequisite links between objectives. For instance if transition T1 produces a capability required by a transition T2, then a prerequisite link will be automatically created between objectives involved in T1 and T2.

**Course generation toolkit**

Fundamental objects for our course generation and for the whole system have been implemented: a curriculum object, a CKTN object, a course object... They will be extensively used in the generation process. The algorithm developed for the generation requires: a CKTN, a knowledge specification (KT), an optional target group and the description of the heuristic to be used.

We created a target public editor (see figure 4a) to let the designer specify his target groups and use them afterwards to generate his courses. Our editor presents him a set of knowledge related to the desired subject matter and simply he has to select the level of acquisition his public possesses on the given capabilities. The editor takes into account the different acquisition levels as well as the acquisition level vocabulary chosen by the designer (kept in a global variable of the system). So, the system adapts the terms presented to the user to qualify the different acquisition levels to the right vocabulary. With this editor, the instructional designer can mark the different states of knowledge of his target group. He can also modify one of the existing target groups.
A CKTN object is automatically defined when a new curriculum object is created (with our system’s curriculum editor). Its prerequisite links and capabilities are marked according to the target group knowledge, to produce a DynCKTN on which the generator will now reason on. We can also already remove from the KT the knowledge that the student possesses according to the marking, since it's not useful to include in the course the objectives for its acquisition.

Each capability of the KT receives a message of objectives’ generation and the chosen heuristic determines which objectives will be used for its acquisition during a learning session. The three heuristics that have been implemented have led to different courses as expected. During the progress of the algorithm, we keep in a list the capabilities which could not be acquired with the CKTN actual objectives, according to the employed heuristic. These capabilities will bring about the introduction of a course prerequisite i.e. some additional knowledge the student should master prior to the beginning of the course.

The generated set of objectives is used to construct the course graph (figure 10c) which is then passed to the structuring process (figure 10d). Then, the system opens a course editor which will allow the designer to modify it if he wishes. A window of statistics also appears to give some information about the generation (figure 10b): the KT specified, the number of objectives chosen and which ones, the KT capabilities already possessed by the student (according to the target group) and the capabilities which require the course to have a prerequisite.
USING COURSE AND CURRICULUM OBJECTS IN AN ITS

An object oriented ITS that exploits curriculums and courses object produced by CREAM-Tool has been built. This ITS comprised four main components: The Curriculum component, the Planner and the Tutor subsystems, the Student model and the Student interface.

The Curriculum component is composed of two parts: a first part is made of one or many objects that are instances of the class Curriculum and the second part is an object of the class Course. A choice of several courses can be offered to the learner, nevertheless we place ourselves in the situation where the learner has already chosen a course.

The student model is an overlay on the curriculum [14]. It includes an inference system that reasons on curriculum structures in order to infer the acquisition of new knowledge in the cognitive part of the student model.

The Planner subsystem [32] uses the course specification and takes into account the learner, as perceived through the portion of the student model that concerns the subject matter (objectives and pedagogical resources available), this subsystem elaborates dynamically the plan for the course. It is in part in this dynamic elaboration of the course that the ITS shows pedagogical competence. There will be re-planning whenever the tutor subsystem considers that the learning is less than adequate. The planner proposes teaching tasks (to motivate the student, to recall previous knowledge, to formulate an
objective, to present content, to assess the student...) for the tutor to realize. The planner is responsible of the global aspects of individualized teaching.

The Tutor subsystem manages the interactions with the learner, realizes the teaching tasks by choosing and using pedagogical resources (teaching material), controls the teaching process and the dialogue with the student, according to the selected tutoring mode (coaching, critiquing, ...) replies to remediation needs and to help requests from the student, updates the student model... The tutor is responsible for the interactions with the learner and the local control on the teaching process.

A student interface has been design to support student activities during the learning process. Figure 11 shows the student interface in a traffic regulation course learning process. This course was generated from a curriculum we built in this domain.

![Figure 11. The Student interface](image)

The traffic regulations curriculum contents a capability model with about 350 knowledge items creates by using CREAM-Tool and content about 150 links between knowledge. Several learning materials have been created by using CREAM-Tools: demonstrations on some driving activities, hypermedia document for the verbal information and a large library of road sign images related to concepts and denoting examples, non-examples and potential deviations of the concept. These image resources are exploited by a test generation kit [31] in other to create interactive hypermedia test associated with concepts. In the same way, True-False tests and multiple choice questions are generated to support the assessment of verbal information knowledge. In the figure 11, the current activity planned by the Tutor is a test that will permit to assess the student knowledge about some road sign concept (indication
The student is asked to point out indication signs. The tutor captures the student performance and gives a feedback based on that. We can see that the student interface integrates tools that support learning activities. In the case present in the figure 11, the presentation of the test is supported by a Netscape browser that have been encapsulated in our ITS. In addition the student interface offers some metacognitive features based on the curriculum model [10]; It allows the student to ask questions that can help him to situate himself in the learning context. The system generates answer to such question by reasoning on the curriculum organization.

**DISCUSSION**

CREAM-Tools takes advantage of existing authoring environments and implements new features. It can be considered as a fourth generation authoring system for several reasons:

1. The subject-matter is well defined; In fact, the CREAM approach of the subject-matter modeling allows the designer to specify fine grains domain-knowledge and to organize them together.

2. Instructional design theories are integrated into the system in order to support the designers through intelligent assistance. These theories could be easily edited to add new constraints and rules in the curriculum development process. Actually, about sixty rules have been introduce in CREAM-Tools and most of them are based on the Gagné’s theory.

3. The authoring environment is adaptable to the designer's needs and he or she can incorporated his own vocabulary as vocabularies varies from one theory to another. In this way, CREAM-Tools offers a global environment specification tool to allow the designer to incorporate his own vocabulary or to choose one of the default vocabularies that are already implemented in the system.

4. Automatic generation of course and of didactic resource is a facility offers by CREAM-Tools that could be very useful in the instructional design context. This facility could allows course designers to quickly prototype and test courses in order to validate them before their real delivery.

5. Although CREAM-Tools serves to build ITS curriculum and course, we found that it could also be used as a very useful tool in formal teaching field. Teachers in a given field can use CREAM-Tools to create the contain to be taught and to create a wide set of varied teaching activities. The system help the teacher to create these activities according to the students profiles.
Even if CREAM-Tools provides all the possibilities above, some works remain to do. For instance, the hints given by the system during the authoring process is only based on instructional theories incorporated in the system. When the designer defines a concept, he or she could be asked to specify examples, non-examples and near miss, if this is not done. We think that, as Tennyson states, it could be very useful to take into account the designers model. This could lead to a more individualized authoring. Our future works will include this problem.

Another thing we will try to look at is the way to enable cooperating curriculum and course authoring. That is how our system could allow several instructional designers to collaborate in curriculum and course building process. We are now work on a meta-authoring language that will permit a distributed way to specify a curriculum.

**CONCLUSION**

We have proposed a curriculum and course model for use in an intelligent tutoring system (ITS). These models have been experimented through the construction of a complete course and its delivery. This paper presented an authoring system that permits to build curriculums and courses based on these models. The outputs of the authoring system are courses and curriculums objects that are immediately usable by others ITS components. They can also be used as curriculums and courses specification for a human teacher. Our models equally support decision making in the teaching-learning process (by the planner and the tutor) and student modeling. To point out how our models allows a high scale ITS development, we have set up some applications based on them (Baxter pump manipulation curriculum, clinic examination in an intensive care unit curriculum and Quebec traffic regulation curriculum). For instance, the Baxter pump manipulation curriculum supports a course including the following learning activities:

- Activities related to theoretic aspects of the infusion pump (component explanations, infusion concept, concepts illustration by images...). Pedagogical resources supporting this part of the course are given by HTML documents.
• Activities concerning the free manipulation of Baxter pump by a nurse (through the execution of tasks).
• Activities related to motivation and help, notably activities where the nurse can be shown by the system how to execute a specific task.
• Activities regarding problem solving in which the nurse is facing a situation where she is required to program an infusion with the help of a Baxter pump.

A prototype of the system has been developed with VisualWorks 2.0 (Smalltalk 80). Actually, a team composed of Industry-University computer specialists has been created and the aim of this team is to transform CREAM-Tools in a commercial product; a beta-version is now available.

Acknowledgment: We thank the MICST (Ministère de l’industrie, du commerce, de la science et de la technologie du gouvernement du Quebec) for providing financial funds for this research. We also thank Frasson M.C., Madja E., Boulhama H., Lupascu D., Leibu, D. and Zalida N. for their collaboration.

REFERENCES


