Semantic Web-based Software Product Line for Building Intelligent Tutoring Systems

Alan Pedro da Silva[1], Evandro de Barros Costa[2], Ig Ibert Bittencourt[3], Patrick H. S. Brito[4], Olavo Holanda[5], and Diego Demerval[6]

[1, 2, 3, 4, 5, 6] GrOW - Group of Optimization of the Web
www.grow.ic.ufal.br
[1] Federal Institute of Education, Science and Technology
Palmeira dos ´Indios - Alagoas - Brazil
alan.silva@ifal.edu.br
[2, 3, 4, 5, 6] Federal University of Alagoas
{evandro, ig.ibert, patrick, olavoholanda, diego}@ic.ifal.br

Abstract. Intelligent Tutoring Systems (ITS) have been assumed as an important learning resource to be added as a module in e-learning systems. However, the construction of such systems is still a hard and complex task that involves, for instance, representation and manipulation of different knowledge source. To alleviate these issues, this paper proposes a new approach for building ITS by integrating Software Product Line and Semantic Web technologies focusing on two software engineering aspects: large-scale production and customization for different learners, and how to allow these knowledge to be automatically shared between software and authors in both reuse and knowledge evolution points of view. This paper shows a modeling for the proposed product line, as well as how the Semantic Web technologies was used to achieve the effective shared knowledge.

1 Introduction

Intelligent Tutoring Systems (ITS) are a kind of complex software system that can be used in an efficient way to improve the student learning process [1,11]. Such systems are provided with mechanisms that enable, in an automated fashion, a better understanding of the students' needs, responding them individually. Moreover, ITS allow student centered learning despite the teacher centered learning, thus allowing the student to control his/her own learning process. In addition, ITS have been assumed as an important learning resource to be added as a module in e-learning systems.

However, the process of constructing Intelligent Tutoring Systems (ITS) is still a time-consuming, hard and complex task, and often requires a high cost of production. For instance, in order to obtain one hour of tutoring, about 200-300 hours of development must be spent [1]. This happens, especially, because an ITS has embedded Artificial Intelligence techniques. First, the complexity appears through the fact of many of these techniques can be used to represent the knowledge to be taught (Domain Model), to represent the knowledge of a student (Student Model), to reproduce the strategies of conventional teaching from a teacher, to implement new pedagogical strategies (Teaching Model), to follow the learning and affective aspects (Diagnosis Process), and to motivate the student and enable an adequate interaction between the system and the student. Therefore, the project of an ITS demands a previous and very accurate study of how the techniques will be used, as the same Artificial Intelligence technique can be used for more than one purpose in an ITS as well as specific functionality can be implemented through integrating more than one AI technique. For instance, a mechanism for evaluating the knowledge of a student can be implemented through the integration of bayesian networks and rule-based reasoning, where each technique can offer different and complementary information about the student.

Actually, beyond the natural problematics of building a particular ITS, there are further limitations to increase even more the development process of such systems, especially if compared to traditional software [15,17,8,4]. The first problem is that every ITS is built independently, which makes any attempt of reusing any part of the system, especially the core components (Student Model, Domain Model, Diagnosis Process and Teaching Model) completely inviable. Another problem is the high maintenance cost due to the use of academics to develop such systems, by other means, using a development team which is not necessarily concerned about aspects of evolution and expansion.

To alleviate the mentioned issues, this paper proposes a new approach for building ITS by integrating Software Product Line and Semantic Web technologies focusing on two software engineering aspects: large-scale production and customization for different learners, and how to allow these knowledge to be automatically shared between software and authors in both reuse and knowledge evolution points of view.

According to [5], a Software Product Line (SPL) is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Based on this definition, SPL offers good conditions for its applicability on the particular context of Intelligent Tutoring Systems because various tutors, in different domains, have common and variable features (functionalities) [6], which allow the creation of an unified platform for tutoring systems that can be adapted according to the individual needs of each author. Moreover, when it comes to software reuse, Software Product Line is one of the most sophisticated concepts in software engineering, which is focused on small-grained, opportunistic and technology-driven software reuse [13]. The strategic reuse provided by SPL combines the business strategies of the market segment (ITS in this work) with the technical strategies of reuse.

Therefore, SPL is an adequate approach for providing software reuse. However, traditional product line approaches do not address questions regarding knowledge reuse and knowledge sharing or evolution. It means an important lack in terms of ITS development and lifecycle. In this sense, the role of the knowledge engineer becomes very important because the correct implementation of an ITS is linked to how well the knowledge is specified. In addition, an ITS deal with knowledge of various types, making necessary the existence of a broad knowledge about the relationship between them, so they can be joint with the purpose of providing learning. Thus, in the process of building an ITS, it is necessary to first express the knowledge, then perform the implementation. This approach becomes inefficient in systems such as Intelligent Tutoring Systems, since the number of knowledge domains an ITS can work is great.

This paper proposes a Product Line approach based on Semantic Web technologies. This is important for the ITS software layer to interoperate with the knowledge one. Thus, the software layer can be built and evolved independently on how the knowledge layer is constructed and evolved. So from this perspective, the Product Lines approach based on Semantic Services indeed produces some progress [16]. By adopting this approach one can obtain a high level of automation in the process of building applications in Product Lines. However, they do not address interoperability between knowledge and software. Thus the proposition of an approach based on Semantic Web technologies assumes the use of intelligent agents in the whole process of searching and sharing information. The agents are important in order to have better software adaptation and re-adaptation with knowledge being considered in various contexts [18][14].

Therefore, this approach enables an automated and dynamic evolution of the applications offered by a Software Product Line. Then, our proposal is to extend this approach to a Semantic Web technologies context in order to provide semantic annotation of all available resources in the context of an ITS, as well as the relationships between them, no matter how they were created.

Thus, in Section 1 a study of related work is described. In Section 2 the proposed architecture of the product line is presented. In Section 3 the artifacts to which the composition and evolution of ITS are performed in an automated way are semantically described.

2 Related Work

Nowadays, there are two approaches which are used for building Intelligent Tutoring Systems: Shell-Based Tolls and Frameworks. Both are not adequate to address the issues focused on this work.

Shell-based tools are developed to provide a mechanism for building ITS independently on the previous knowledge of advanced computing science concepts. This provides the authors with a building mechanism containing several built-in concepts and tolls. This way, the author is demanded to know how the pedagogical strategies work, as well as the representation of the main ITS components: domain model and student model. That means the author needs to understand how each component work, so that each component can be properly configured. This approach is inefficient from the reuse point of view: each ITS has its own configuration in such a fashion their educational components are also related in a specific way. This fact compromises reuse, maintenance and evolution. The direct components reuse is impracticable given that each essential component has a particular configuration for each application instance. The maintenance is also a complex task because a change in any component implies changes in all components, hence, there are configuration and reconfiguration needs in the whole system. The evolution is one aspect that was not correctly dealed [1][2][12][3].

Frameworks are tools that have better reuse and adaptation power. It is an intrinsic characteristic in such projects and this implies that the author can reuse components in several applications. However, its necessary that: (i) The tutor author knows how to use the framework components as well as its behaves, for that system schedule is attached in the frameworks and (ii) what are the consequences of the addition of a component in all the system. (iii) Moreover, its manipulations are restrict to Software Engineers. From the maintenance point of view, there is less effort: when a change occurs, work it is only needed in that particular system place. From the evolution point of view, the frameworks are very restrict for all the possible functionalities that were previously defined [10][7] [9]. However, every software layer is directly attached to the specific knowledge domain the system was designed. In this way, there are implementation needs when any knowledge change occurs.

3 Our Software Product Line Architecture

Figure 1 shows the proposed architecture, including all the necessary elements for the ITS production in large scale with simplified modeling, interoperability and scalability based on Semantic Web technologies. Every component follows the Massayo [4] semantic models, which are the learner, pedagogical and domain ones.

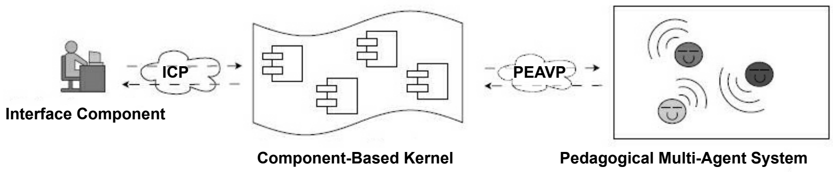

Fig. 1. Our Software Product Line Architecture.

Component-Based Kernel (CBK): For each ITS implementation there is model instances, these instances will be applied to specify exactly the ITS particularity. The instantiation of this model requires the specification of the domain in which the same is proposed to teach, what pedagogical strategies must be used, what must be represented on the student model, and the specification of the problem types that will be exposed to the learners.

Interface Component (IC): This component represents exactly how the learner will interact with the ITS. This interface has to be designed according to the domain specifications which it will be inserted. For example, the possible problem variations that may be represented by students have to be implemented, as well as all the possible behaviors the tutoring system may have when dealing with the student;

Interface Communication Protocol (ICP): This protocol defines the interaction rules between Manager and Interface. In fact, it is possible to have several Interface implementations, on the other hand, in order to each interface implementation communicate correctly with the Manager, it is necessarily for the interface to properly implement the protocol. This protocol, besides of defining the rules that establish when the interface may suffer changes from the system (originated from the Manager), and when interface may suffer change by the learner. It will also specify behaviors semantic and rules, in order that the own interface can be embedded with some intelligence level;

Pedagogical Multi-Agent System: The pedagogical agents are autonomous entities which have a collective purpose of teaching the best possible way the learners in the system. Another important point is that each pedagogical agent has as function a single activity well defined. Beyond this, a pedagogical agent has the power of adaptativity related to the ontologies specification which are stored on the repository. However, it is possible to exist differences between the agents to decide which approach might be best adopted in each moment of interaction with the student. As several different pedagogical strategies may be defined in a product line specification, there may exist a negotiation between these agents to define the correct action of the system at each moment.

Pedagogical Evolution, Adaptability and Variability Protocol (PEAVP): This protocol defines the interaction rules between Pedagogical Agents. With this protocol, an agent will know the right moment to realize the environment, to act, and to invoke an other agent to help in the accomplishment of its activity. Beyond this, that protocol will be capable to define the actuation rules of the agent when there is the possibility of competition between more than one agent in choosing what is the right action to be taken. As quoted before, there may exist disagreement on choosing the best action to be taken by the system.

4 Semantic Artifact Specication and Relationships

In this section, the artifacts related to the proposed Product Line will be discussed. These artifacts are semantically described in the Ontology Descriptions, presented in Figures 1 and 2. Through these codes, all the possible educational resources will be semantically annotated. This is important in order to exist an automated process of building and instantiation of the Intelligent Tutoring Systems. Those artifacts compose the *Component-Based Kernel* (CBK), and they are described below:

IntelligentEducationalEnvironment: This component represents the environment as a whole. Students, teachers and tutors see this environment as a single entity, and they interact directly through it transparently.

PedagogicalIntelligentAgent: This component represents a type of agent responsible for giving instruction in a particular knowledge domain to a certain student. This agent has its own teaching strategy, it performs the planning of the learner study and monitor his activities during the process of teaching him.

SupportAgent: This component represents the agents which will support the effective work of the Pedagogical Agents. Actually, this agent will be responsible for maintaining the environment with external information to the interaction between the participants and the environment.

EducacionalResource: This component represents all the educational resources that could be used with the purpose of teaching. All resources should be linked to a specification so that they can be found and used properly and orderly.

StudentModel: This component represents the student model, it is used to identify the characteristics of each student. And through this model, the system resources will personalize the student results.

KnowledgeAssessmentMethod: This component represents the methods that assess at which cognitive level, each student who is interacting with the environment, is. Therefore, for each round of interaction between the student and the environment must exist a method that assesses the knowledge evolution of the students. These methods may follow different theories, such as constructivism, evolutionary and subjectivism.

InferenceMechanism: This component represents the inference mechanisms that are used in order to measure what are the possible approaches that students should receive prior to the execution of them. Aiming to optimize the education of the students, causing him to reach your goals as quickly as possible.

DomainModel: This component represents the syntax and semantics which a particular domain is represented. It represents all the aspects that the student needs to learn until the end of all their interactions.

TeachingPlan: This component represents a resource that describes at which sequence the students must interact with each educational resource. Therefore, depending on the situation in which the student is, the teaching plan should be adjusted to give continuity in the teaching process properly.

PedagogicalStrategy: This component represents the pedagogical strategies in which a particular pedagogical agent will compromise to address a certain student throughout the teaching process.

01 : $SupportAgent \sqsubseteq IntelligentAgent$
02 : $PedagogicalIntelligentAgent \sqsubseteq IntelligentAgent$
03 : $PedagogicalIntelligentAgent \sqsubseteq EducacionalResource$
04 : $AudioClass \sqsubseteq EducacionalResource$
05 : $ModalLogicBased \sqsubseteq StudentModel$
06 : $PropositionalLogicBased \sqsubseteq StudentModel$
07 : $OntologyBased \sqsubseteq StudentModel$
08 : $Constructivist \sqsubseteq KnowledgeAssessmentMethod$
09 : $AnalogyStrategyBased \sqsubseteq PedagogicalStrategy$
10 : $SolvingProblemStrategyBased \sqsubseteq PedagogicalStrategy$
11 : $KnowledgeAssessmentMethod \sqsubseteq \top$
12 : $StochasticMethod \sqsubseteq InferenceMechanism$
13 : $LogicalModelBased \sqsubseteq DomainModel$
14 : $OntologyModelBased \sqsubseteq DomainModel$
15 : $NeuralNetworkModelBased \sqsubseteq DomainModel$
16 : $BayesianNetworkModelBased \sqsubseteq DomainModel$
17 : $Objetivistic \sqsubseteq KnowledgeAssessmentMethod$
18 : $Rule\text{-}BasedReasoning \sqsubseteq InferenceMechanism$
19 : $Case\text{-}BasedReasoning \sqsubseteq InferenceMechanism$
20 : $Subjetivista \sqsubseteq KnowledgeAssessmentMethod$
21 : $VideoClass \sqsubseteq EducacionalResource$
22 : $SupportWebService \sqsubseteq WebService$
23 : $PedagogicalWebService \sqsubseteq EducacionalResource$
24 : $PedagogicalWebService \sqsubseteq WebService$

Ontology Description 1: Inheritance relationship among all the entities.

Furthermore, in order to representthe way these elements relate itself, a hierarchy of entities using the concept of inheritance was created. For example, in Ontology Description 1, the relations 02 and 03 express that an Intelligent Pedagogical Agent (IntelligentPedagogicalAgent) can be seen as an intelligent agent or as an educational resource (EducationalResource). This definition is important because in the process of the construction of the Intelligent Pedagogical Agent for the environment, it should behave both as an intelligent agent (respecting interaction protocols) as educational resource (Subject to a certain meta-definition on how to specify this type of element). The other relations of the Ontology Description 1 below comply with the same line of reasoning.

However, it is also necessary to relate the elements Computational involved in the construction of educational environments adaptive and semantic level of composition, because they define where they should be embedded elements ranging from adapting to the personalized education plan. The Ontology Description 2 represents the relationships between the entities at that level. The Following is the meaning of the described compositions relations:

environmentHasSupportAgent: This relationship indicates that a Intelligent Educational Environment (*IntelligentEducationalEnvironment*) has Agents Support (*SupportAgent*), according to the relations 33 and 34. In relationship 33 is expressed as the element *environmentHasASupportAgents* is a subset of the entity *IntelligentEducationalEnvironment*, and the same time, in relation 34, it is stated that the same element *environmentHasSupprtAgent* has linked to it from the elements type *SupportAgent*. Therefore, the environments must have agents support in order to assist teaching activities promoted the pedagogical agents, such as seeking information external environment, educational resources index, updated domain model, update the model student and so on.

environmentHasIntelligentPedagogicalAgents: This relationship indicates that an Intelligent Educational Environment (*IntelligentPedagogicalEnvironment*) has Agents Teaching (*IntelligentPedagogicalAgents*) as the relations 35 and 36. 35 is expressed in relation to the element *environmentHasIntelligent-PedagogicalAgents* is a subset Element Intelligent Educational Environment (*IntelligentPedagogicalEnvironment*), and that even in about 36, may have elements of type *IntelligentPedagogicalAgents*. The idea is that the environment has bound to it agents capable of Education of policy knowledge that he intends to teach. In reality, agents Pedagogical *IntelligentPedagogicalAgents* built in a flexible manner in order to adjust to a given field of knowledge and a model student.

environmentHasDomainModel: This relationship indicates that an Intelligent Educational Environment (*IntelligentEducationalEnvironment*) has only one domain model (*DomainModel*) according to the relationships 37, 38 and 39. In the relationship 37 is described that there is only one element of type *environmentHasDomainModel*, and this element is a subset of the entity Intelligent Educational Environment (*IntelligentEducationalEnvironment*) according to relation 38. And in relationship 39, is described that an element of type *environmentHasDomainModel* should have linked to it an element of the Domain Model type *DomainModel*. This relationship is important to restrict an educational environment to just one knowledge domain. Indeed, this restriction will be chained to the process of building educational environments, because as the environment itself is restricted to a single domain knowledge, all other elements will follow this principle.

environmentHasStudentModel: This relationship indicates that an Intelligent Educational Environment (*IntelligentEducationalEnvironment*) has a single student model *StudentModel*, according to the relationships 40, 41 and 42. In the relationship 40 is expressed that there is a single element of the type *environmentHasStudentModel*, and in its turn, in the relationship 41, it represents that the same element is a subset of the element *IntelligentEducationalEnvironment*. And in the relationship 42, it is expressed that this element has a Student Model *StudentModel* linked to it. In the same way that the domain model, the specified student model for the domain will chain in a sequence of adjusts in all the elements that use them as reference to perform their tasks.

pedagogicalAgentsHasPedagogicalStrategies: This is the ratio representing the Pedagogical Agents (*IntelligentPedagogicalAgents*) must have strategies teaching (*PedagogicalStrategies*). This entity is defined in relations 25 and 26. 25 is represented in relation to there is a link called (*pedagogicalAgentsHasPedagogicalStrategies*) that is built within the Intelligent Agent for Pedagogical (*IntelligentPedagogicalAgents*). In turn, the relation 26, that every relationship is described has *pedagogicalAgentsHasPedagogicalStrategies* pedagogical strategy. This educational strategy must be implements the following model specification of the user model user, and educational resource, so that it can be adapted to the knowledge being entered by their Agent.

pedagogicalAgentsHasInferenceMechanism: This is the relationship which represents that the Pedagogical Agents (*IntelligentPedagogicalAgents*) must have nnference mechanisms (*InferenceMechanism*). This entity is defined in the relationships 27 and 28. Following the same reasoning to other compositions, these relationships are represented that an entity (*pedagogicalAgentsHasInferenceMechanism*) is a subset of Intelligent Pedagogical Agent, and that it has a mechanism inference associated with the purpose of making assumptions about the best next approach that must be accomplished to the student for each interaction. In the same way as the other entities, the actions taken by an element of this type must be customized with respect to the domain model and the student model.

pedagogicalAgentsHasAssesmentMethod: This is the relationship which represents that the Pedagogical Agents (*IntelligentPedagogicalAgents*) must have assessment methods (*AssesmentMethod*). This entity is defined in relationships 29 and 30. Also following the same reasoning to the other compositions, in these relationships are represented that an entity (*pedagogicalAgentsHasAssesmentMethod*) is a subset of Intelligent Pedagogical Agent, and that it has a assessment method (*AssesmentMethod*). Still, this assessment method will have as responsibility to evaluate the students' development in relation to the interactions that are occurring between the student and the environment.

pedagogicalAgentsHasTeachingPlan: This is the relationship which represents that the Pedagogical Agents (*IntelligentPedagogicalAgents*) must have teaching plans (*TeachingPlan*). This entity is defined in relationships 31 and 32, which means (by analogy to other relationships) that an element *pedagogicalAgentsHasLearningPlan* is embedded within the element *IntelligentPedagogicalAgents*, and that it has the element of the type *TeachingPlan*. The Plan of teaching represents how the agent will lead the teaching to the student who the same will be dealing with. In it, all the resources that will be used to teach the student are related, and in which sequence of these resources will be available.

TeachingPlanHasEducationalResource: This relationship represents that the Teaching Plans (*TeachingPlan*) should have Educational Resources (*EducationalResource*). This entity is defined in relationships 43 and 44, which means (similarly to the other relationships) that an element *TeachingPlanHasEducationalResource* is embedded within the element *TeachingPlan*, and that the same element have the type *EducationalResource*. As previously mentioned,the teaching plan is who references the educational resources, and all educational resources are accessed through what is described in the teaching plan.

25 : $\exists$ *pedagogicalAgentsHasPedagogicalStrategies* $\sqsubseteq$ *IntelligentPedagogicalAgent*
26 : $\top \sqsubseteq \forall$ *pedagogicalAgentsHasPedagogicalStrategies:PedagogicalStrategy*
27 : $\exists$ *pedagogicalAgentsHasInferenceMechanism* $\sqsubseteq$ *IntelligentPedagogicalAgent*
28 : $\top \sqsubseteq \forall$ *pedagogicalAgentHasInferenceMechanism:InferenceMechanism*
29 : $\exists$ *pedagogicalAgentHasAssesmentMethods* $\sqsubseteq$ *IntelligentPedagogicalAgent*
30 : $\top \sqsubseteq \forall$ *pedagogicalAgentHasAssesmentMethods:KnowledgeAssessmentMethod*
31 : $\exists$ *pedagogicalAgentHasTeachingPlan* $\sqsubseteq$ *IntelligentPedagogicalAgent*
32 : $\top \sqsubseteq \forall$ *pedagogicalAgentHasTeachingPlan:TeachingPlan*
33 : $\exists$ *environmentHasSupportAgent* $\sqsubseteq$ *IntelligentEducationalEnvironment*
34 : $\top \sqsubseteq \forall$ *environmentHasSupportAgent:SupportAgent*
35 : $\exists$ *environmentHasPedagogicalAgents* $\sqsubseteq$ *IntelligentEducationalEnvironment*
36 : $\top \sqsubseteq \forall$ *environmentHasPedagogicalAgents:IntelligentPedagogicalAgent*
37 : $\top \sqsubseteq \leq 1$ *environmentHasDomainModel*
38 : $\exists$ *environmentHasDomainModel* $\sqsubseteq$ *IntelligentEducationalEnvironment*
39 : $\top \sqsubseteq \forall$ *environmentHasDomainModel:DomainModel*
40 : $\top \sqsubseteq \leq 1$ *environmentHasStudentModel*
41 : $\exists$ *environmentHasStudentModel* $\sqsubseteq$ *IntelligentEducationalEnvironment*
42 : $\top \sqsubseteq \forall$ *environmentHasStudentModel:StudentModel*
43 : $\exists$ *TeachingPlanHasEducationalResource* $\sqsubseteq$ *TeachingPlan*
44 : $\top \sqsubseteq \forall$ *TeachingPlanHasEducationalResource:EducationalResource*

Ontology Description 2: Composition Relationship among all the entities.

5 Conclusions and Future Works

This work proposes a new approach to building Intelligent Tutoring Systems in large scale concerning the knowledge as the system's main artifact, thus, making the building process dependent on how this knowledge was conceived and distributed. After a deep study about the whole building process of the current approaches and all the resources which can be used on its building, a semantic description to enable the automatic building and evolution of ITS was created, as described by section 4. This description is part of a wider architecture which comprises the software product line architecture proposed on section 1. This architecture manifests the idea of protocols to enable the evolution and interaction through agents.

This new approach to construct ITS environments benefits to the author, in the sense that programming background is not necessary, could be verified in this proposal. From the implementation point of view, one could notice the software product line approach is, nowadays, the largest considering the software artifacts reusability, which is promising to the intelligent tutoring systems context. As future work, a refinement of the specification through the addition of new kinds of resources along with their corresponding implementations, such as pedagogical strategies and new visualization forms.

References

1. V. Aleven, B. M. McLaren, J. Sewall, and K. R. Koedinger. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley, and T. W. Chan, editors, International Conference on Intelligent Tutoring Systems, pages 61–70. Springer Verlag, 2006

2. V. Aleven, B. M. McLaren, J. Sewall, and K. R. Koedinger. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In Intelligent Tutoring Systems, pages 61–70, 2006
3. L. Aroyo, A. Inaba, L. Soldatova, and R. Mizoguchi. Ease: Evolutional authoring support environment. In Intelligent Tutoring Systems, pages 140–149, 2004
4. I. I. Bittencourt, E. Costa, M. Silva, and E. Soares. A computational model for developing semantic web-based educational systems. Know.-Based Syst., 22(4):302–315, 2009
5. P. C. Clements and L. Northrop. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley, August 2001
6. P. Dillenbourg, P. Mendelsohn, and D. Schneider. The distribution of pedagogical roles in a multiagent learning environment. In Proceedings of the IFIP TC3/WG3.3 Working Conference on Lessons from Learning, pages 199–216, Amsterdam, The Netherlands, The Netherlands, 1994. North-Holland Publishing Co.
7. M. Dragomiroiu, M. Ventuneac, I. Salomie, and T. Coffey. Application framework development for virtual learning environments. In 25th Int. Conf. lnformation Technology lnterfaces IT, Cavtat, Croatia, 2003
8. E. El-Sheikh and J. Sticklen. A framework for developing intelligent tutoring systems incorporating reusability. In IEA/AIE '98: Proceedings of the 11th international conference on Industrial and engineering applications of artificial intelligence and expert systems, pages 558–567, London, UK, 1998. Springer-Verlag
9. L. Esmahi and F. Lin. Designing Distributed Learning Environment with Intelligent Software Agents, chapter A Multiagent Framework for an Adaptive E-Learning System, pages 218–241. Idea Group, 2005
10. B. Goodman, M. Geier, L. Haverty, F. Linton, , and R. McCready. A framework for asynchronous collaborative learning and problem solving. In In the Proceedings of the 10th International Conference on Artificial Intelligence in Education, AIED, November 2001
11. K. R. Koedinger, J. R. Anderson, W. H. Hadley, and M. A. Mark. Intelligent tutoring goes to school in the big city. International Journal of Artificial Intelligence in Education, 8:30–43, 1997
12. T. Murray. Authoring Tools for Advanced Technologies Learning Environments: Toward cost-effective adaptive, interactive and intelligent educational software, chapter An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art, pages 493–546. Number 17. Kluwer Academic Publishers, Netherlands, 2003
13. L. M. Northrop. Sei's software product line tenets. IEEE Softw., 19(4):32–40, 2002
14. G. Picard, J. F. H¨ubner, O. Boissier, and M.-P. Gleizes. Reorganisation and self-organisation in multiagent systems. In J. Sabater-Mir, editor, 7th European Workshop on Multi-Agent Systems (EUMAS 2009), pages 66–78, 2009
15. M. Rodrigues, P. Novais, and M. F. Santos. Future challenges in intelligent tutoring systems : a framework. In International Conference on Multimedia and Information and Communication Technologies in Education, 2005
16. J. J. Rusk and D. Gasevic. Semantic web services-based reasoning in the design of software product lines. In SPLC (2), pages 123–130, 2008
17. J. Self. The defining characteristics of intelligent tutoring systems research: Itss care , precisely. International Journal of Artificial Intelligence in Education, 1999
18. I. D. V. Tamma, S. Phelps and M. Wooldridge. Ontologies for supporting negotiation in e-commerce. Engineering Applications of Artificial Intelligence, 18(2):223–236, 2005