

Personalization of E-Learning Scenarios: The Case of Learning Scenarios for Programming Languages

Fathi Essalmi

Leila Jemni Ben Ayed

Research Laboratory of Technologies of Information and Communication UTIC
Ecole Supérieure des Sciences et Techniques de Tunis,
5, Av. Taha Hussein, B.P. 56, Bab Mnara 1008, Tunis, TUNISIA
Fathi.essalmi@utic.rnu.tn, Leila.jemni@fsegt.rnu.tn

ABSTRACT

The personalization of E-learning scenarios can be considered as the representation of knowledge to the learners according to their preferences and characteristics. This representation is assumed to be expressive as possible for describing and recommending appropriate knowledge to the learners. Furthermore, it is fundamental to communicate learning scenarios with clear and understandable representation. In this paper, we present an approach for automatic generation of adaptive hypermedia which facilitates the teaching of context-free grammar (CFG). This adaptive hypermedia is represented in the form of class diagrams and activity diagrams of UML. We use the class diagram to represent the terminals and the non-terminals and relations between them. In some case, the order between terminals and non-terminals of the context-free grammars is not defined in generated class diagrams. For that reason, we add the activity diagrams to precise this order. The teachers can use these diagrams to prepare course and to explain different concepts of programming languages.

Keywords: Adaptive hypermedia, E-Learning Scenarios, Programming languages

1. INTRODUCTION

When it is possible to represent the same concept with different ways, comprehensible and clear representation of concepts is fundamental to communicate any concepts to the learners. Consequently, the automatic generation of comprehensible and clear representation will facilitates the preparation of courses with high quality and allows learners to easily understand the concepts. In this paper, we define an approach for the automatic generation of class diagram and activity diagram of UML [12] from a Context-Free Grammars (CFG) [13] written with the Extended Backus-Naur Form (EBNF) [9]. In fact, UML is a standard graphical language which offers comprehensive and clear representation of concepts.

In [4], we have defined a set of generation rules that generate a class diagram from a CFG written with the EBNF. The resulting diagram allows the learner to visualise graphically the grammar. But in some case, the order between the classes and the attributes is not

defined with the class diagram. For that reason, we add the activity diagram to precise this order. In the literature there are several works that mention the advantages of the graphical notation. In [11], A. van Lamsweerde states the popularity of techniques based on tabular formats and diagrammatic notations. Similarly, in [18], Zimmerman and his co-authors have conducted an experimental evaluation that confirms the advantages of tabular and diagrammatic notations. This has been confirmed by several cognitive science studies [15]. In this context, Pinsky and Wipf have demonstrated the positive impact of visual representation on recall and retention of taught material [16]. Furthermore, Davis has showed that computer graphics fulfills some sort of relevance and intrinsic interest to students and provides an ideal problem-based learning platform upon which a variety of topics can be taught effectively [3]. At the same time, Saw and Majid have demonstrated that within the graphics display mode, the explanatory and cognitive types of communicative interaction were dominant [17]. In particular, schematic diagrams allow students to organize information in the problem to facilitate problem translation and solution [10]. In fact, schematic diagrams beneficiate from the advantage of graphics and they are powerful in the representation of relations between elements. This powerful can be explained by the fact that good fit to an object memory is possible only when the spatial relationships between the parts have been properly specified [14].

In order to beneficiate from the numerous advantages of describing knowledge with graphics and preserve a high degree of precision, we define a process which allows the generation of a graph composed of a class diagram and a set of activities diagrams. The concepts are defined in the class diagram, and the order between them is defined in the activities diagrams.

The paper is structured as follows: section 2 presents the general context of our work. Section 3 presents the concepts which we have used for the definition of our approach. Section 4 presents the proposed process for the generation of class and activity diagrams from the context-free grammar. Section 5 presents the elementary transformations in depth. Section 6 presents two algorithms for BNF-UML transformation and

EBNF-UML transformation. Section 7 concludes the paper.

2. GENERAL CONTEXT

The personalization in the E-learning context has been upgraded from the personalization of learning contents to the personalization of learning scenarios. Learning contents are defined with the simple combination of learning objects, where learning scenarios are defined with the combination of learning objects by taking into account pedagogical approaches. Among the works which allow the personalization of learning contents, we distinguish: COLER (Collaborative Learning environment for Entity-Relationship modeling) [1], and PERSO [2] (an hypermedia adaptive system for supporting E-learning). In these works, the selection of learning contents for learners has been limited to the systematic association of learner characteristics to the learning contents and does not combine this association with pedagogical approach. In our previous work, [6] we have defined a multi-parameters personalization approach of learning scenarios and a way for the integration of the IMS LD specification [7]. We have also explained the feasibility and the advantages of using UML class diagram as a navigational support for E-Learning [5].

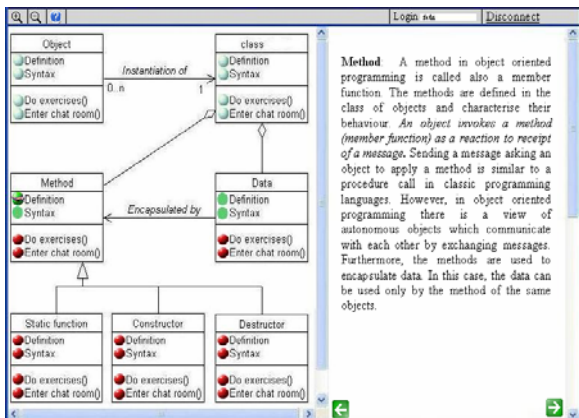


Figure 1. An example of UML adaptive hypermedia

As an example, Figure 1 [5] presents personalized learning scenarios for learning the object oriented programming paradigm. The proposed solution allows learners to benefit at the same time from the personalization of learning scenarios, and from the high expressivity of the standard language UML. Our aim is to generate automatically the class diagram, which is used as a navigational support.

3. DEFINITIONS

In this section, we present the concepts used in our approach. We start by defining the context-free grammar, the Extended backus-Naur form, and finally an overview of class and activity diagrams.

A context-free grammar [13] is a 4-tuple $G = (V, \Sigma, R, S)$, where:

- V : A finite set of non-terminals.

- Σ : A finite set of terminals.
- R : A finite set of production rules.
- S : A start symbol.

The Backus-Naur Form (BNF) is used for the definition of context-free grammars. The Extended Backus-Naur Form (EBNF) adds the regular expression syntax [9]. The BNF contains the following notation:

- ::= : Definition of a non-terminals.
- Space () : Concatenation of two grammatical units (terminals or non-terminal)
- | : Choice between two definition of a non-terminal

The EBNF adds the following notations:

- : n repetition (or concatenation) of grammatical unit, $n \in [0..+\infty]$
- + : n repetition (or concatenation) of grammatical unit, $n \in [1..+\infty]$
- [] : n repetition (or concatenation) of grammatical unit, $n \in [0..1]$

A class diagram [12] is a graph of classifiers elements connected by their various static relationships. The classes are the most common classifiers in the class diagram. A class is drawn as a rectangle with three compartments separated by horizontal lines. The top name compartment holds the class name and other general properties of the class; the middle list compartment holds a list of attributes; the bottom list compartment holds a list of operations. The classes are connected with relations such as associations, generalizations, compositions, and aggregations. An association can be a binary association or an n-ary association. A binary association is an association among exactly two classifiers (including the possibility of an association from a classifier to itself). An n-ary association is an association among three or more classifiers. A composition is a strong form of aggregation, which requires that a part instance be included in at most one composite at a time and that the composite object has sole responsibility for the disposition of its parts. A Generalization is the taxonomic relationship between a more general element (the parent) and a more specific element (the child)

An activity diagram [12] is a variation of a state machine in which the states represent the performance of actions or subactivities and the transitions are triggered by the completion of the actions or subactivities. A state is a condition during the life of an object or an interaction during which it satisfies some condition, performs some action, or waits for some event.

4. THE GENERATION PROCESS

The generation process which combines the rules that generate the class diagram with the rules generating the activity diagrams is presented in the following activity diagram (Figure 2). For the specification of this process, we use the activity diagram of UML for the reason that

UML activity diagram allows the specification of processes in the form of activity. Furthermore, the use of activity diagram allows to benefit from the numerous advantages of graphical languages.

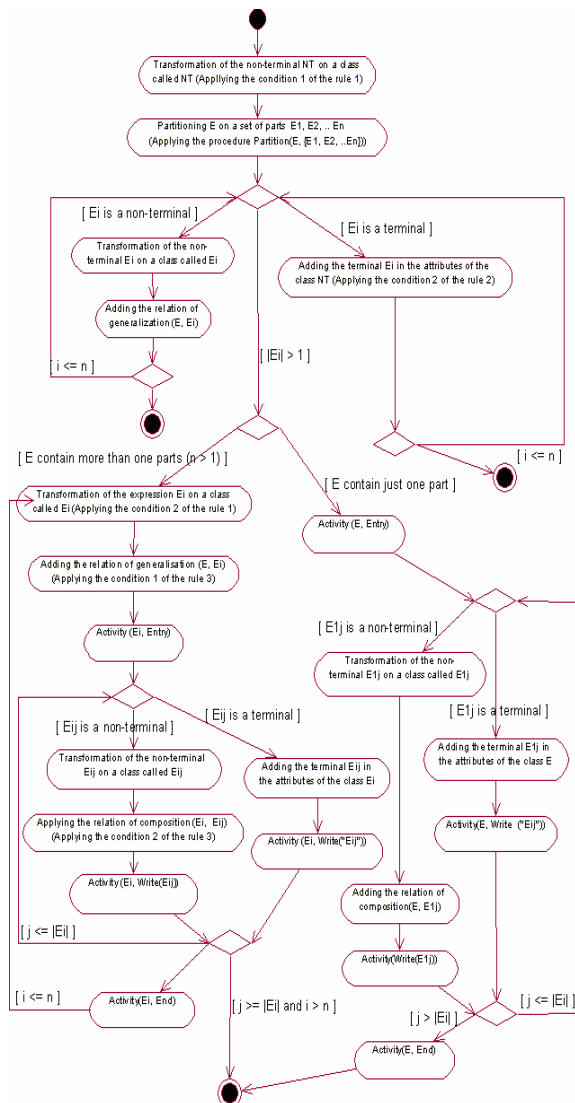


Figure2: The Generation Process.

The generation process generates a class diagram for one production rule, and evolves the first diagram when it is applied on other production rules. The production rules entered in the generation process are of the form: $NT ::= E$, where $NT \in \Sigma$ and $E \in (V \cup \Sigma)^*$, this process uses a procedure partition defined as follows: $Partition(E, \{E1, E2, \dots, En\})$: This operation parts E in a set $\{E1, E2, \dots, En\}$ such as $E = E1 | E2 | \dots, En$ and $E1, E2, \dots, En$ do not contain the symbol '|'

As it is illustrated in the activity diagram presented in Figure 2, the generation process generates the classes from some features of a grammar by applying some rules (each non terminal which appears in left part of a production rule is translated into a class). Then, the defined process uses this diagram in order to evolve it and treats each production rule to generate sub classes, attributes, associations, and other relations (composition, dependence,...).

Notes

The activities of an activity diagram are ordered according to the apparition of the grammatical units. The generation process adds an activity diagram for each class which needs a high level of precision. We use the symbol E_{ij} in the generation process (Figure 2) to specify the grammatical unit number j in E_i .

5. ELEMENTARY TRANSFORMATIONS

In this section, we present the elementary transformations which transform a set of production rules written in BNF or EBNF to a class diagram. We assume that A, B and C are three non-terminals. We also assume that a, b and c are three terminals.

4.1. ELEMENTARY BNF-UML TRANSFORMATIONS

Given a set of production rules written in BNF, the application of BNF-UML transformation results a class diagram and a set of activities diagrams.

In order to obtain an operational transformation, we distinguish the three combinations of grammatical units:

- Two non-terminals
- Two terminals
- Non-terminal and terminal

We distinguish also the two conventions used between grammatical units:

- Space ()
- |

In Table 1, we present six rules of transformation. These rules can be generalized by introducing more than two grammatical units in the right-hand side of a production rule. If the symbol $|$ is used between two non-terminals, we transform it to generalization. If the symbol $|$ is used between two terminals or between one terminal and one non-terminal, we transform it to enumeration. If the symbol space is used between two non-terminals, we transform it to composition. If the symbol space is used between two terminals we transform the terminals into attributes. If the symbol space is used between one terminal and one non-terminal, we transform it into composition and the terminal into attribute.

Table 1. Mapping BNF to UML

BNF	UML
$A ::= BC$	
$A ::= b c$	
$A ::= B C$	
$A ::= b c$	
$A ::= B c$	
$A ::= B c$	

4.2 ELEMENTARY EBNF-UML TRANSFORMATIONS

Given a set of production rules written in EBNF, the application of EBNF-UML transformation results a class diagram and a set of activities diagrams.

In Table 2, we propose simple rules for the transformation of the followings grammatical units:

- Non-terminal
- Terminal

We distinguish also three conventions related to grammatical units:

- *
- +
- []

The symbol * belong to EBNF and belong also to UML with the same meaning. We transform the symbol * into *. We transform the symbol + into 1..* and the symbol [] into 0..1.

Table 2. Mapping EBNF to UML

EBNF	UML
$A ::= B^*$	
$A ::= B^+$	
$A ::= [E]$	
$A ::= b^*$	
$A ::= b^+$	
$A ::= [b]$	

5. TRANSFORMATION ALGORITHMS

In this section, we present two algorithms. The first one allows the transformation of a BNF grammar to an UML model (BNF-UML algorithm). The second extend the first one and allows in addition the transformation of an EBNF grammar to an UML model (EBNF-UML algorithm).

Given a context free grammar $G = (V, \Sigma, R, S)$, in the BNF-UML algorithm, we use three programming classes. The first one is responsible for the analysis of EBNF expression (it is named Ex). The second is responsible for the construction of class diagrams (it is named CD). The third is responsible for the construction of activity diagrams (it is named AD). The following procedures are specified in the three programming classe Ex, CD, and AD:

- Ex.Partition (E, {E1, E2, ...En}): partitioning E in a set {E1, E2, ...En} such as E1, E2...En do not contains the symbol '|'.
- CD.Class (A): adds a class titled A if the class A do not exist in the class diagram.
- CD.Enumeration (A, {E1, E2, ...En}): adds the stereotype <<enumeration>> in the class A and adds the attributes E1, E2, ...En in the class A.
- CD.Attribute (A, E): adds the attribute E in the class A.
- CD.Composition (A, E): adds the relation A composed of E.
- CD.Generalization (A, Ai): adds the relation A can be Ai.

- AD.Activity(A, Start): adds the activity start in the activity diagram that describe A.
- AD.Activity(A, Write(E)) adds the activity Write(E) in the activity diagram that describe A.
- AD.Activity(A, End): adds the activity End in the activity diagram that describe A.

Algorithm BNF-UML

Begin

For each production rule $A ::= E$ **Do**

 CD.Class (A)

 Ex.Partition (E, $\{E_1, E_2, \dots, E_n\}$)

If $E_1, E_2, \dots, E_n \in \Sigma$ **Then**

 CD.Enumeration (A, $\{E_1, E_2, \dots, E_n\}$)

Else If $n = 1$ **Then**

 AD.Activity(A, Start)

For $j = 0$ to $|E_1|$ **Do**

If $E_{1j} \in \Sigma$ **Then**

 CD.Attribute (A, E_{1j})

 AD.Activity(A, Write("E₁"))

Else

 CD.Class (E_{1j})

 CD.Composition (A, E_{1j})

 AD.Activity(A, Write(E_{1j}))

End If

End For

 AD.Activity(A, End)

Else

For $i = 1$ to n **Do**

If $E_i \in V$ **Then**

 CD.Class (E_i)

 CD.Generalization (A, E_i)

Else

 CD.Class (A_i)

 CD.Generalization (A, A_i)

 AD.Activity (A_i , Start)

For $j = 1$ to $|E_i|$ **Do**

If $E_{ij} \in \Sigma$ **Then**

 CD.Attribute (A_i , E_{ij})

 AD.Activity(A_i , write("E_{ij}"))

Else

 CD.Class (E_{ij})

 CD.Composition (A_i , E_{ij})

 AD.Activity (A_i , Write(E_{ij}))

End If

End For

 AD.Activity(A_i , End)

End If

End For

End If

End For

End BNF-UML

For the definition of the Algorithm EBNF-UML, we add the function "cardinality" which determine and return the repetition number of an EBNF expression to be used as the cardinality of the attributes and the relation in the class diagrams according to the mapping Table 2 . Subsequently, the procedures "CD.Attribute"

and "CD.Composition" are extended for considering the determined cardinality as following:

- Integer CD.cardinality (E): return the cardinality of the expression E.
- CD.Attribute (A, E, CD.cardinality (E)): adds the attribute E in the class A associated with the cardinality CD.cardinality (E).
- CD.Composition (A, E, CD.cardinality (E)): adds the relation A composed of E associated with the cardinality CD.cardinality (E).

Algorithm EBNF-UML

Begin

For each production rule $A ::= E$ **Do**

 CD.Class (A)

 Ex.Partition (E, $\{E_1, E_2, \dots, E_n\}$)

If $E_1, E_2, \dots, E_n \in \Sigma$ **Then**

 CD.Enumeration (A, $\{E_1, E_2, \dots, E_n\}$)

Else If $n = 1$ **Then**

 AD.Activity(A, Start)

For $j = 0$ to $|E_1|$ **Do**

If $E_{1j} \in \Sigma$ **Then**

 CD.Attribute (A, E_{1j} , CD.cardinality (E_{1j}))

 AD.Activity(A, Write("E₁"))

Else

 CD.Class (E_{1j})

 CD.Composition (A, E_{1j} , CD.cardinality (E_{1j}))

 AD.Activity(A, Write(E_{1j}))

End If

End For

 AD.Activity(A, End)

Else

For $i = 1$ to n **Do**

If $E_i \in V$ **Then**

 CD.Class (E_i)

 CD.Generalization (A, E_i)

Else

 CD.Class (A_i)

 CD.Generalization (A, A_i)

 AD.Activity (A_i , Start)

For $j = 1$ to $|E_i|$ **Do**

If $E_{ij} \in \Sigma$ **Then**

 CD.Attribute (A_i , E_{ij} , CD.cardinality (E_{ij}))

 AD.Activity(A_i , write("E_{ij}"))

Else

 CD.Class (E_{ij})

 CD.Composition (A_i , E_{ij} , CD.cardinality (E_{ij}))

 AD.Activity (A_i , Write(E_{ij}))

End If

End For

 AD.Activity(A_i , End)

End If

End For

End If

End For

End EBNF-UML

7. CONCLUSION

In this paper, we define a process for the generation of a graph composed of a class diagram and a set of activities diagrams from a CFG. The generated graph allows to benefit from the numerous advantages of describing knowledge with graphics and preserves a high degree of precision. The class diagram represents the static view of the grammar, and the activity diagram represents the dynamic view. Representing the grammars of programming languages with UML facilitates the preparation of courses with high quality and allows learners to easily understand the concepts. We have developed a prototype for the generation of the class diagram and the related activities diagrams. This prototype allows also to interact with the generated diagram in two ways. First, it allows teacher to update the diagrams, add specific links to the pedagogical contents, and personalize the diagrams. Second this tool allows the learner to navigate in the UML adaptive hypermedia.

REFERENCES

- [1] Constantino-González, M., Suthers, D., and Santos, J. Coaching web-based collaborative learning based on problem solution differences and participation. *International Journal of Artificial Intelligence in Education*, 13, 261-297, 2003.
- [2] Chorfi, H., Jemni M. PERSO: Towards an adaptive e-learning system, *Journal of Interactive Learning Research*, 15, 433-447, 2004.
- [3] Davis T. A., "Graphics-Based Learning in First-Year Computer Science", *Computer Graphics Forum*, OnlineEarly Articles, Jun 2007.
- [4] Essalmi F, and Jemni Ben Ayed, L., Graphical UML View from Extended Backus-Naur Form Grammars, *The 6th IEEE ICALT*, 2006
- [5] Essalmi F, Jemni Ben Ayed L, Jemni M, and Kinshuk., UML Class Diagram as a Navigational Support for E-Learning, *The 8th IEEE ICALT*, 2008.
- [6] Essalmi, F., Jemni Ben Ayed, L., Jemni, M. A Multi-Parameters Personalization Approach of Learning Scenarios, *The 7th IEEE International Conference on Advanced Learning Technologies*, Niigata, Japan, 90-91, 2007.
- [7] Essalmi, F., Jemni Ben Ayed L., Jemni, M. Personalization of Learning Scenarios for E-Learning, *2nd International Conference on Interactive Mobile and Computer aided Learning*, Princess Sumaya University for Technology, Amman, Jordan, 2007.
- [8] Essalmi, F., Jemni Ben Ayed, L. A Process for the Generation of Personalized Learning Scenarios based on Ontologies, *The first International Conference on ICT & Accessibility*, Hammamet, Tunisia, 173- 175, 2007.
- [9] ISO/IEC, EXTENDED BNF 1996 www.dataip.co.uk/Reference/EBNF.php
- [10] Jitendra A, "Teaching Students Math Problem-Solving Through Graphic Representations", *TEACHING Exceptional Children*, vol. 34, No. 4, 2002, pp. 34-38.
- [11] Lamsweerde, A.V, "Formal Specification: a Roadmap", *ICSE'00: The Future of Software Engineering Track*, ACM Press, pp. 147-159, 2000.
- [12] OMG, UML Notation Guide, *OMG-Unified Modeling Language*, v1.5, 2003 www.infor.uva.es/~mlaguna/is1/materiales/UML1.5.pdf
- [13] Ong, C.-H. L., Context-free grammars. <http://users.comlab.ox.ac.uk/luke.ong/teaching/moc/cfg2up.pdf>
- [14] Peterson M A, Kim J H, "On what is bound in figures and grounds", *VISUAL COGNITION*, 2001, volume 8, pp. 329-348
- [15] Petre, M., "Why looking isn't always seeing: Readership skills and graphical programming", *Communications of the ACM*, Vol 38 N° 6, pp. 33-44, 1995.
- [16] Pinsky L E and Wipf J E, "A Picture is Worth a Thousand Words: Practical Use of Videotape in Teaching", *Journal of General Internal Medicine*, volume 15, Issue 11, November 2000, pp. 805-810,
- [17] Saw K. G, Majid O, N. Abdul Ghani, H. Atan, R. M. Idrus, Z. A. Rahman and K. E. Tan, "The videoconferencing learning environment: Technology, interaction and learning intersect", *British Journal of Educational Technology*, Published article online: Jun 2007.
- [18] Zimmerman M. K., Lundqvist K, and Leveson N G., "Investigating the readability of state-based formal requirements specification languages", *ICSE'02: 22rd International Conference on Software Engineering*, pp 33- 43. ACM Press, 2002.