Novel Algorithm Explanation Techniques for Improving Algorithm Teaching

Andreas Kerren*
Computer Science Department
University of Kaiserslautern
Kaiserslautern, Germany

Tomasz Müldner†
Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia, Canada

Elhadi Shakshuki‡
Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia, Canada

1 Introduction

The analysis and the understanding of algorithms is a very important task for teaching and learning algorithms. We advocate a strategy, according to which one first tries to understand the fundamental nature of an algorithm, and then—after reaching a higher level of awareness—chooses the most appropriate programming language to implement it. To facilitate the process of understanding of algorithms, their visualization, in particular animation, is considered to be the best approach. Traditional Algorithm Animation (AA) systems usually aim for teaching algorithms in higher education, see for example the chapter introduction of Kerren and Stasko [2002] or the earlier anthology on software visualization [Stasko et al. 1998].

Evaluations of systems designed to achieve this aim using various visualization and animation techniques have shown that such systems have not achieved many expectations of their developers [Hundhausen et al. 2002]. One reason for this failure is the lack of stimulating learning environments which support the learning process by providing features such as multiple levels of abstraction, support for hypermedia, and learner-adapted visualizations. Furthermore, runtime interpretation requires specific input data and cannot consider all possible inputs and often suffers from the lack of focus on relevant data, see Braune and Wilhelm [2000]. Most existing algorithm animation systems do not address the issue of representing algorithm invariants, implementing algorithms in specific programming languages, paying attention to their structure, or finding their time complexity. Adapting facilities for the learner behaviour are not supported, nor is the additional use of media beyond graphics and animation.

2 The SHALEX System

In this poster paper, we describe a system that includes a hypermedia environment providing links between various kinds of multimedia. Our system, called Structured Hypermedia Algorithm Explanation [SHALEX 2006], aims to address most of the aforementioned problems. It provides several novel features, such as reflection of the high-level structure of an algorithm and support for programming the algorithm in any procedural programming language. By defining the structure of an algorithm as a digraph of abstractions, algorithms may be studied top-down, bottom-up, or using a mix of the two. It is also possible to support several levels of abstractions which help the learner to understand basic properties of the algorithms as well as to recognize good implementation strategies.

*e-mail: kerren@acm.org
†e-mail: Tomasz.Muldner@acadiau.ca
‡e-mail: Elhadi.Shakshuki@acadiau.ca

SOFTVIS 2006, Brighton, United Kingdom, September 04–05, 2006.
© 2006 ACM 1-59593-464-2/06/0009 $5.00

A major weakness of many existing systems is that they do not adapt to the learner's behaviour. Therefore, a good student may be bored while a novice student may be overwhelmed. SHALEX includes a learner model to provide *spatial* and *temporal* links, and to support evaluations and adaptations. In this context, the system's users can play one of the following four roles: *learners*, who study algorithms; *authors*, who are responsible for tasks such as creating algorithm explanations, or assigning evaluations; *administrators*, who are responsible for tasks such as maintaining user accounts; *algorithm administrators*, who are responsible for tasks such as group management of users assigned to study specific algorithms or management of algorithm explanations. SHALEX supports many algorithms; explanations of which are created by various authors. To support this, we designed a taxonomy of explanations which has a tree-like structure. Non-leaf nodes of the taxonomy represent concepts, such as "Iterative Algorithms" (the root represents all algorithms). Leaves represent explanations of specific algorithms, created by specific authors.

Structured Hypermedia and Abstraction Levels In our approach, operations are provided in a textual form, but there is also a hyperlinked visual description used to help the learner understand basic properties of an algorithm, for example *algorithm invariants*. Each operation is either implemented in an abstraction at the lower level, or it is a primitive operation. This is a generalization of micro/macro-level animations used in HalVis [Hansen et al. 2002] which will allow the novel mode of studying unavailable in any other visualization system: an algorithm may be studied top-down, bottom-up, or using a mix of the two.

We define the *algorithm structure* as a hierarchical Abstract Algorithm Model (AAM) which is an acyclic digraph with nodes representing *abstractions* and edges representing operation dependencies. Each abstraction is designed to focus on a single operation used directly or indirectly in the algorithm, i.e., it explains a single operation op and consists of a *textual* and a *visual* representation. The textual representation includes, among other things, an Abstract Data Type (ADT) that gives a high-level view of generic data structures and operations. As an example, let's assume that f is an operation. The abstraction that explains f, abst(f) is a pair (ADT, repr(f)), where ADT consists of data types and primitive operations. repr(f) contains visual representations, additional text, and concrete implementations (see below). There is a directed edge from abst(f) to abst(g) if g is one of the primitive operations from the ADT abst(f). Thus, a successor abstraction provides a partial implementation of the operation from the predecessor abstraction. An AAM of an algorithm f is a graph rooted at abst(f).

To build an algorithm explanation, we construct an AAM with a sufficient number of levels so that the learner is able to understand how and why the algorithm works. Let's consider the Insertion Sort algorithm as an example. This algorithm can be implemented using operations from two ADTs: the Insertion ADT provides generic operations, such as insert and the primitive operation swap; the Insert ADT provides only primitive operations. The AAM for this algorithm is shown in Figure 1. Further examples are provided in [Müldner and Shakshuki 2004; Müldner et al. 2005].

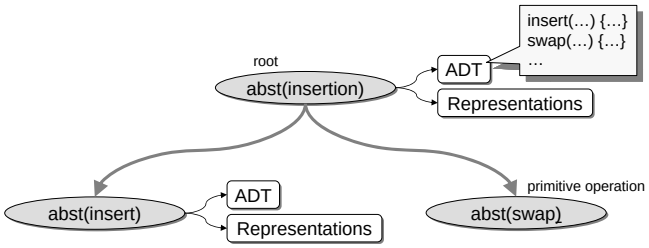

Figure 1: An AAM for Insertion Sort

Important Features Associated *visualizations* may be used by the learner to help him or her understand the basic properties of this abstraction. It is possible to embed any web-viewable animations built by AA systems, such as Ganimal [Diehl and Kerren 2002] or JSamba [2006], as well as other formats, for example Macromedia Flash visualizations, animated GIFs, sound files, etc. SHALEX provides *easy language transfer* by an intermediate representation of all AAM's primitive operations, called an Abstract Implementation Model (AIM). To implement the algorithm in a specific programming language, the learner has to map to the selected language all primitive operations that do not have implementations in the AAM. The representations in AIM are generic: they use high-level concepts that can be mapped to many procedural programming languages. Explanation of *algorithm complexity* is one of the most difficult goals of algorithm visualization, because it requires mathematical proofs that are hard to visualize. The current version of SHALEX includes three kinds of tools designed to help the learner to derive the complexity of the algorithm being studied.

Learner and Author Models SHALEX is an interactive system that allows the learner to select one of the available algorithms to study. It uses a *learner model* to record learner activities. These interactions are vital to support *active learning* [Hundhausen et al. 2002]. SHALEX helps the learner not only to understand *what* the algorithm is doing but also *how* the algorithm works; as well *why* the algorithm works (algorithm correctness).

In addition, it uses an *author model* to record decisions made by an author. For example, the author may decide to prepare, for a single algorithm, various lessons with different evaluations, and various AAM trees providing more or fewer abstractions. Authors' responsibilities include selecting tools to keep track of the learner performance. SHALEX provides several tools, such as measuring the time spent on studying specific issues and comparing this time with author-specifies soft and hard deadlines, or keeping track of user activities, such as selecting menu items etc. The author then selects a specific tracking tool and then decides on the adaptivity of the system. Additionally, our system has built-in features that help to evaluate the effectiveness of studying algorithms using this system. To compare the effectiveness of two different lessons for the same algorithm, the administrator may create two disjoint groups of students, and assign a different lesson to each group.

Authoring The process of creating an algorithm explanation is supported by various tools, such as a library of existing lessons, and descriptions of ADTs. The author may fetch an existing item and adjust to her or his needs. A novel and essential feature of SHALEX is that it allows the author or the algorithm administrator to assign different modes of learning an algorithm: *top-down*, *bottom-up*, and *learner-selected*. In top-down learning, the learner studies the textual and optionally visual representation of the source node of the AAM at first. Then, the learner studies all successor nodes and so on. The bottom-up learning approach is performed in an opposite direction, i.e., starting from leaves of the AAM. The learner-selected mode needs a more careful description. For any operation op that appears in the operation currently focused on, the learner may select op and request one of the following: help, taking a test, or explanation of this operation. In the first case, SHALEX provides a context-sensitive help. In the second case, the learner may be given a test, and if the test is passed, the learner model will be updated. The author may specify that in order to complete studying the algorithm, the learner has to complete all tests, using evaluations available in the author model.

3 Conclusion and Future Work

This poster paper briefly presented our proposed system for explaining algorithms, which is based on structured hypermedia approach. It has been shown that the system has some fundamental advantages, including availability of studying an algorithm top-down, bottom-up, or using a mix of the two; support for understanding invariants; building a learner model to provide spatial and temporal links. The first versions of algorithm visualizations were implemented using Macromedia Flash. For the next version, we are considering using HTML pages to display more complex and interactive visualizations. In order to make our system usable, we are also planning on performing evaluations in class with students from computer science at various universities.

References

BRAUNE, B., AND WILHELM, R. 2000. Focusing in Algorithm Animation. *IEEE Transactions on Visualization and Computer Graphics 6*, 1, 1–7.

DIEHL, S., AND KERREN, A. 2002. Reification of Program Points for Visual Execution. In *Proceedings of the First IEEE International Workshop on Visualizing Software for Understanding and Analysis (VisSoft '02)*, IEEE Computing Society Press, Paris, Frankreich, IEEE, 100–109.

HANSEN, S. R., NARAYANAN, N. H., AND HEGARTY, M. 2002. Designing Educationally Effective Algorithm Visualizations: Embedding Analogies and Animations in Hypermedia. *Journal of Visual Languages and Computing 13*, 3, 291–317.

HUNDHAUSEN, C., DOUGLAS, S., AND STASKO, J. 2002. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing 13*, 3, 259–290.

JSAMBA, 2006. Project Homepage. `http://www-static.cc.gatech.edu/gvu/softviz/algoanim/jsamba/`.

KERREN, A., AND STASKO, J. T. 2002. Algorithm Animation – Chapter Introduction. In *Software Visualization*, S. Diehl, Ed., vol. 2269 of *LNCS State-of-the-Art Survey*. Springer, 1–15.

MÜLDNER, T., AND SHAKSHUKI, E. 2004. On Visualization and Implementation of Algorithms. In *Proceedings of the 5th International Conference on Information Technology Based Higher Education and Training (ITHET '04)*, IEEE Computer Society Press, Istanbul, Turkey, IEEE, 138–143.

MÜLDNER, T., SHAKSHUKI, E., KERREN, A., SHEN, Z., AND BAI, X. 2005. Using Structured Hypermedia to Explain Algorithms. In *Proceedings of the 3rd IADIS International Conference e-Society '05*, IADIS, 499–503.

SHALEX, 2006. Project Homepage. `http://cs.acadiau.ca/~solid/ae.htm`.

STASKO, J. T., DOMINGUE, J., BROWN, M. H., AND PRICE, B. A. 1998. *Software Visualization*. MIT Press.