The Impact of Different Portability Factors During the Life Cycle of an Educational Software Adaptation Project

Betty A. Collis and Italo De Diana
University of Twente, The Netherlands

Abstract

In this article an example is worked out that illustrates the interrelationship of the various factors that influence educational software portability. Five phases in the life cycle of a software product originally created in one setting but being adapted for use in another are considered, and in each of these phases some of the various portability factors are shown to be most critical. (Keywords: software portability, software development.)

In the first chapter of this Special Issue, the following research questions were identified as important:

1. How can we better understand the critical factors that influence the portability of educational software?
2. How are the factors interrelated? How are they related in time and cost to the portability of educational software?
3. How can we deal with these factors to improve the portability of educational software? (Collis & De Diana, 1990, p.156)

A collection of salient factors was identified in this chapter and has been referenced at the start of each of the sections within the Issue. Each of the articles, in different ways, has responded to the second and third research questions, but sometimes the response has been implicit rather than explicit. Also, the authors have chosen to emphasize some factors more than others in their analyses.

In this article, all of the factors, taken as an interrelated set, are considered in one systematic example. In order to better see the temporal relationships of the factors, their application within the life cycle of an educational software

product being ported from its original setting to a new context is described. In order to better see the conceptual interrelationship of the factors to one another, an adaptation of Nielsen's (1986) seven-level virtual protocol model for human-computer interactions is developed and used to help identify opportunities for improving the successful portability of a product at each of the stages of its life cycle.

ORGANIZING THE FACTORS

In Collis & De Diana (1990, pp. 155-156), the following factors were identified as having an impact on the likelihood that a product can be successfully used in an educational setting other than that for which it was developed:

1. Technical factors, relating to the:
   - Physical input of data and mechanisms for human-computer interactions;
   - Program architecture and algorithms;
   - Authoring languages and tools used in developing the software;
   - Operating system level;
   - Hardware characteristics;
   - Network level (i.e.,related to connecting the software with other objects, such as peripherals or other computers)
2. Educational factors:
   - Educational needs and relevance
   - Curriculum content, globally and in detail;
   - Instructional approach, at the micro level (i.e., How are individual questions handled? What reinforcement is given? etc.), and at the macro level (i.e., How is the class and lesson generally organized for instruction? What is the general approach of the software?);
   - "Tone and style" of educational interactions;
   - Social and physical setting of the learning environment in which the software is assumed to be used
   - Considerations relative to teachers and their experience with computer use
3. Social/cultural factors:
   - Language in which the learner interacts with the software;
   - Tone and style of communication
   - Cultural identity;
   - Political sensitivities;
   - Cultural opinions about the roles of teachers and students
   - Cultural references and assumptions

4. Organizational factors:
    - Institutional procedures affecting software-related decision making and implementation;
    - Copyright and ownership;
    - Cost-related issues, including pricing, development costs, and cost effectiveness
    - Marketing and distribution issues;
    - Maintenance
    - Management of design, development, and distribution of software

Given this list of factors, it is valuable to find a way to visualize their interrelationships. Nielsen's seven-level approach to human-computer interaction (1986) can be adapted so that these factors are reorganized in a way that shows some of the interrelationships among them. The seven layers can be redefined in terminology specific for educational software and the factors aligned with them as shown in Figure 1 (De Diana & Collis, 1990).

These levels reflect Neilsen's model and also can help to integrate the factors identified as particularly influencing the portability of software used in educational settings. The integration of levels and factors that is visualized in Figure 1 can be expanded as follows:

Level 1: *Content to be Manipulated by Software*
Data as physical input, Language as data

Level 2: *Strategies for Handling Data (Design Level)*
Program architecture and algorithms

Level 3: *Strategies for Handling Data and Interactions (Implementation Level)*
Authoring language and tools; Operating system; Networks and peripherals; Hardware characteristics; Instructional approaches (micro level); Tone and style of interactions and communication (micro level); Curriculum (micro level); Teacher involvement in design

Level 4: *Organization of the Overall Learning Experience (Within the Software)*
Curriculum; Instructional approach (general); Tone and style of educational interactions and communications; Teacher involvement in design

Level 5: *Organization of Learner-Computer Interaction (External to the Software)*
Social organization of the learning environment; Instructional approach (general approach favored by teacher); Technical factors relating to interaction (i.e., networks; connections with monitors, printers, other peripherals); Teacher characteristics

Level 6: *Organizational Context of Software Use (External to the Learner and the Software)*
Curriculum; Physical organization of the learning environment in which computer use occurs; Institutional procedures for decision making and implementing computer use; Copyright and ownership; Cost factors and marketing; Maintenance

Level 7: *Socio-Cultural Environment*
Cultural identity, Political sensitivities, Language (*in context—reflecting cultural and* social norms), Tone and style of educational interactions and communication (overall level); Teacher characteristics

The first of the general research questions asks what factors influence the portability of educational software. Figure 1 has shown a global selection of the factors and suggests a way to sequence the relationships among them. Bork's prediction (1976) about the relative complexity of technical and human factors in software portability is at least visually supported by this approach, in that the human factors permeate all but the first and second levels. But for the model to be useful, it must help to address the other research questions—Can this way of thinking help to improve decision making relative to portable educational software?

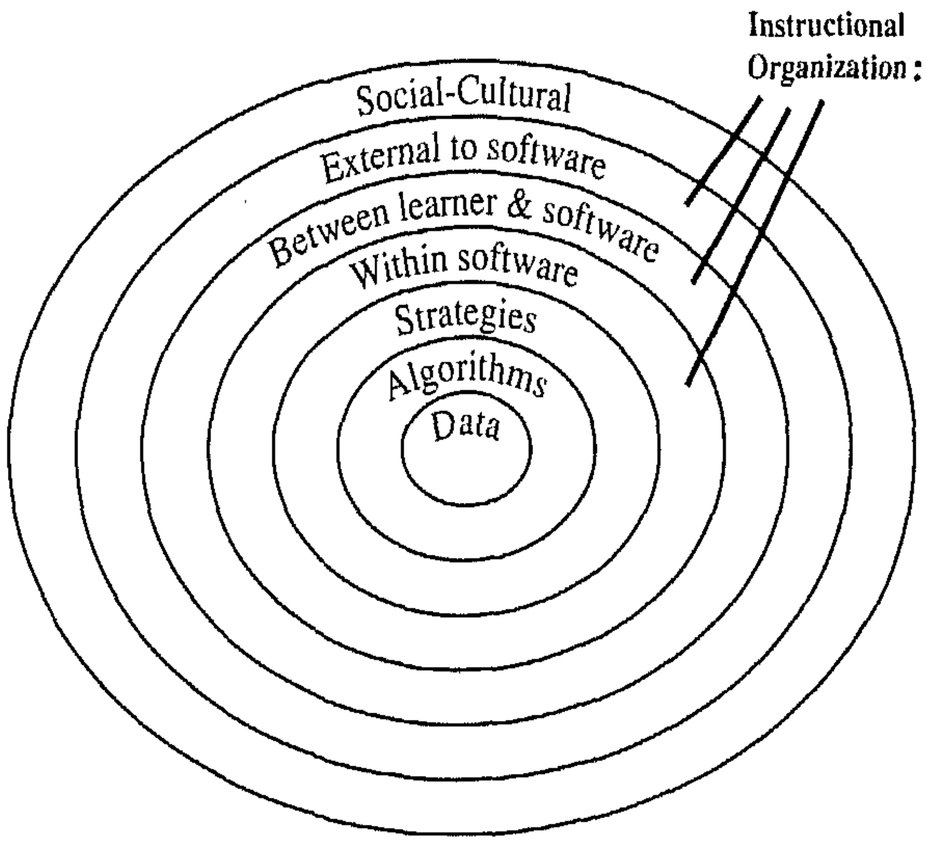

Figure 1. A seven-level model for factors that influence the portability of educational software.

APPLYING THE FACTORS: RESEARCH QUESTIONS 2 AND 3

Analyzing a situation is only a first step toward improving it. The second and third research questions ask: How can a better understanding of factors that influence portability help the educational software developer produce more portable software? How can it help the educational decision maker make more informed judgments about the portability, and the cost of adapting, an existing product? The seven layers in Figure 1 can be applied to considerations in the life cycle of a product involved in a portability experience in a way that may be helpful for both designers and decision makers.

A Five-Phase Life Cycle for a Product Being Transferred from One Setting to Another

The life cycle of a software product originally produced in one setting but now involved in a portability decision relative to a new setting includes the following five phases: the initial-decision phase, the global-redesign phase, the adaptation phase, the distribution and maintenance phase, and the implementation phase. Within each of these phases, some factors will have relatively more influence than others on the activity of the phase and on the possibility that the software transfer process will fail to proceed. In the following sections, a prediction of which factors are most pertinent at each of the five phases will be made. Also, the meanings of the factors will emerge more clearly through the examples.

The initial-decision phase. During this phase, the potential acquirer of the software must make initial decisions about the need for the software, and whether it is best to acquire it from somewhere else and adapt it or produce it locally. (Political and economic considerations such as those described by Murray-Lasso (1990) and Oliveira (1990) are particularly pertinent here). Figure 2 shows the levels that appear to be most influential during this phase.

Within theses levels, the following variables can be hypothesized as being most influential during the initial decision phase:

Level 3:
Hardware characteristics (general), operating system
Level 4:
Instructional approach (general)
Level 6:
Curriculum (general), Cost factors and marketing
Level 7:
Cultural identity, Political sensitivities, Overall tone and style of educational interactions

Probably at this first phase the factors related to fit with available equipment, cost, cultural, and political considerations have the capacity to most strongly

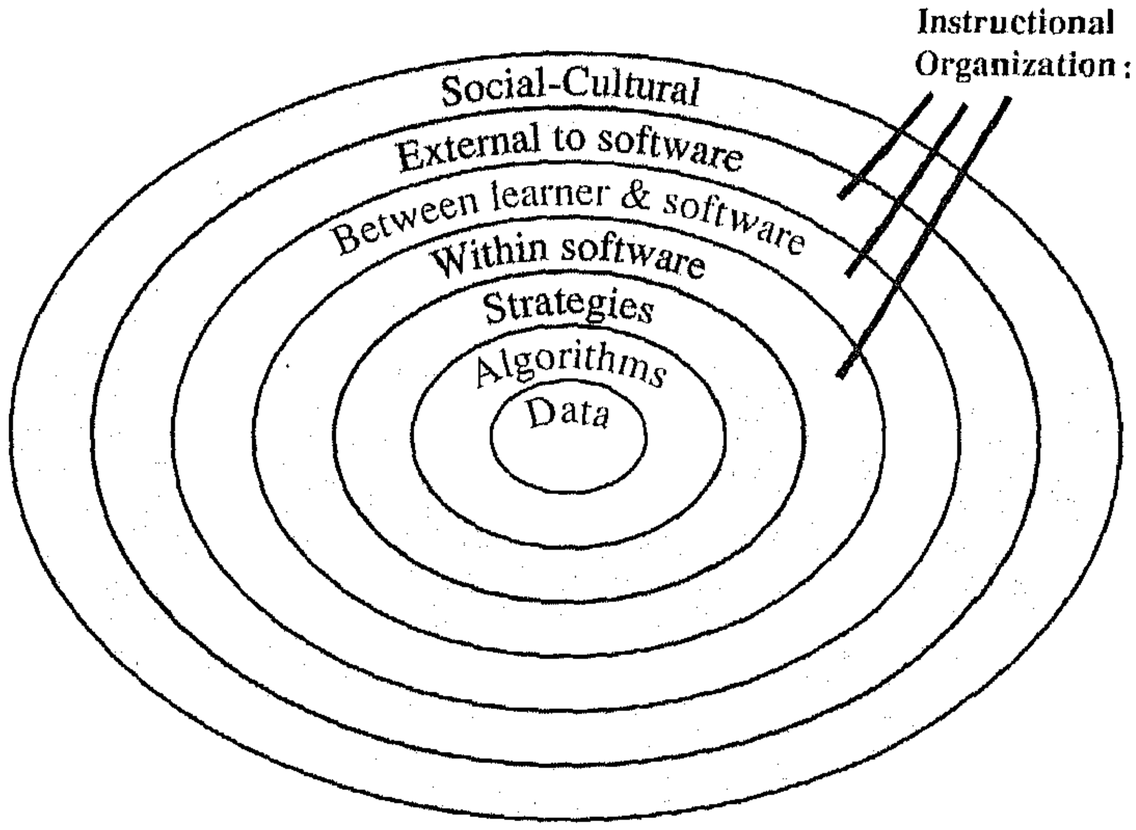

Figure 2. Factors most influential at the initial-decision phase.

influence the first round of decision making relative to the portability of an existing product. Political issues, the matter of cultural identity, and organizational jurisdiction, for example, can strongly influence an initial decision about software adaptation if a local software development industry is to be given priority in software-acquisition decisions. Also, at a global level, lack of fit with the general curriculum and discomfort with the tone of the educational interactions in the program can defeat any further interest in the software. Cost, if it goes above a certain point, can also suddenly kill any further interest in a software package.

Which of these critical factors can be anticipated and manipulated by the software designer? Lack of fit with curriculum can be minimized through content-free software or software with teacher-adaptable data sets. Tone and style of interaction and general instructional approach, however, are harder to alter unless the software will be redesigned and redeveloped. Cost may be controlled through artificial means, such as subsidies, or through redesign of the software into smaller or simpler components. In general, the political and cultural sensitivities toward acquiring software from "outside" are probably beyond the manipulation of the designer, unless he or she works from the start as part of a local team.

The global-redesign phase. In this phase a preliminary decision has already been made to adapt software developed somewhere else. Before actual adaptation can occur, the software must be examined more carefully to make a further assessment of its portability feasibility. While the first phase of decision making often occurs after a brief, even superficial sort of assessment, this

second phase is often done by someone with professional experience and after a more careful examination. What factors and levels are most likely to convince this person not to continue with a software adaptation project at this time? Perhaps the following, as visualized in Figure 3 and involving the following variables:

Level 1:
Data characteristics (character sets)
Level 3:
Program language; Hardware requirements (specific);
Level 6:
Curriculum (specific to a subject area), Copyright and ownership, Pricing and marketing,
Level 7:
Language (in context—the overall level of vocabulary and tone of language use)

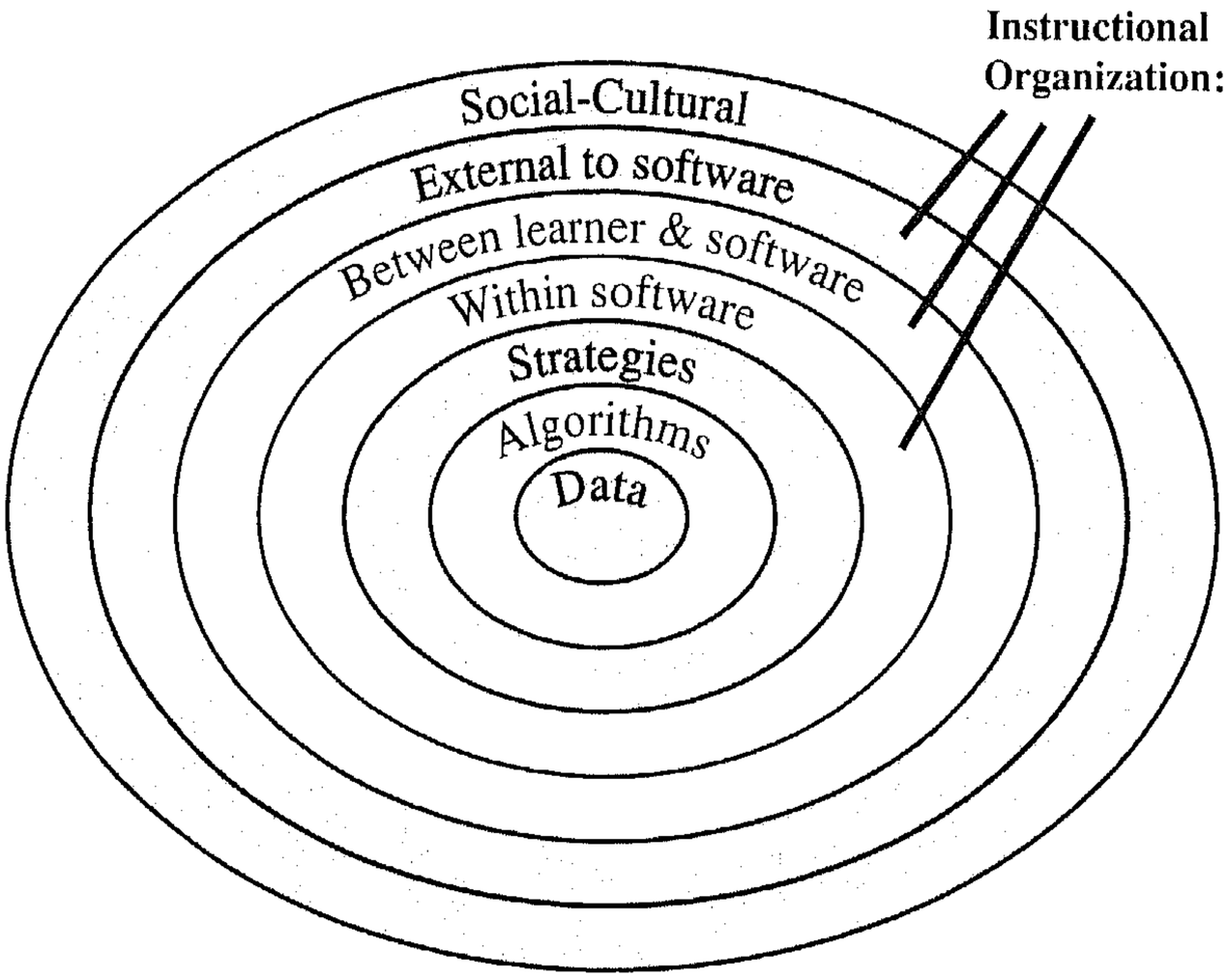

Figure 3. Influential factors at the global-redesign phase.

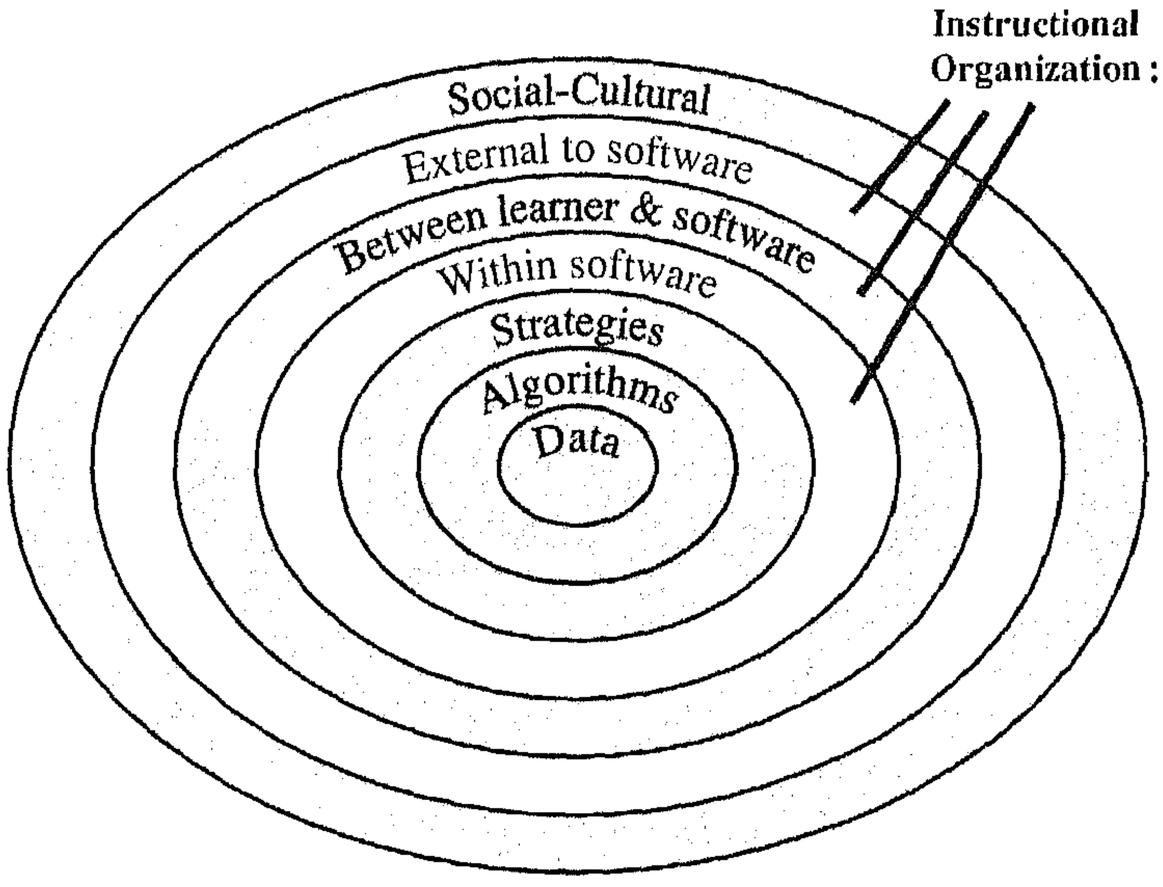

Figure 4. Factors most influential at the product-adaptation phase.

Curriculum fit will be reassessed, but from a more specific perspective than at the first initial-decision stage. Also the instructional approach suggested by the software, both in terms of classroom organization and at the microlevel in the software, can threaten a portability decision. Hardware characteristics and program language and architecture can strongly influence decision making when more detailed time and cost appraisals are being made.

Which of these variables can be anticipated by the designer or developer? Operating system compatibility, the focus of the EASI Project in Ontario (Vraets, 1990), can be reduced as a problem when more interfaces such as EASI are in use or when the industry itself improves operating system portability. A lack of fit with the instructional approach familiar in a culture and/or favored by a particular teacher, particularly at the macrolevel may be more difficult to reconcile. Software based on a model of an exploratory environment, such as a database or a Logo-type language, may have no likely transfer in national or classroom cultures where more formal, rule-based learning is the norm. And finally, copyright and ownership issues, interrelated with cost, can end portability considerations unless resolved at this stage.

Product adaptation. If a product makes it to this phase, it means the decision maker is reasonably convinced that the factors discussed during the previous two phases are not of critical enough proportion so as to have stopped the project. During the product-adaptation phase, the product is actually being adapted for the local use. Which factors and levels are most critical here? Figure 4 gives a visualization.

The variables most likely to be critical during the product adaptation phase could be:

Level 1:
Data as physical input, Language as data
Level 2:
Algorithmic level
Level 3:
Authoring language and tools, Hardware characteristics, Instructional approach (microlevel), Tone and style of communications and interactions (microlevel)
Level 5:
Technical factors relating to connections with other hardware
Level 7:
Cultural references, Language (in context)

In this phase the technical considerations are uppermost. A poorly designed program where, for example, algorithms are not well documented or data are not kept separate from instructions, can prohibit portability if the time and effort needed to adapt the software outweigh the available resources or the value of the program. Screen-display language translation is a major difficulty here, both at the meaning level and the data and presentation level.

Fortunately there are approaches to software design and development initially that can anticipate and lessen the difficulties of the technical adaptation level (see the articles by Kearsley (1990) and Bork (1990) for fuller discussions). An approach to program design in which Levels 1 to 5 of Figure 1 are kept as separate and well defined as possible can make a significant contribution to improving the likelihood of portability at the technical-realization level.

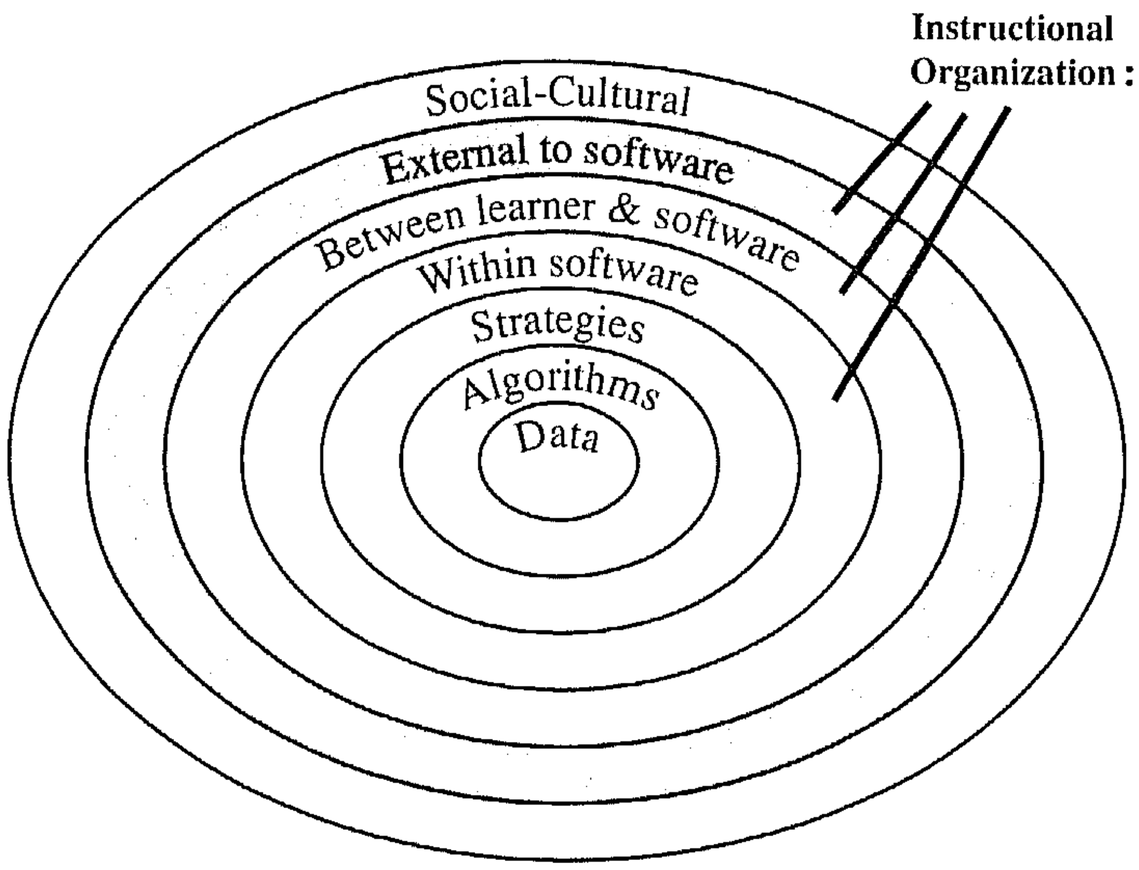

Figure 5. Factors most influential at the distribution and maintenance phase.

Distribution and maintenance. Assuming a product is adapted so that it runs on at least some of the hardware systems available in the acquirer's country, there is still a fourth phase that can constrain the portability of software. This phase relates to distribution and maintenance. How do prospective users find out about the software? How can they evaluate it? How do they order it? How long must they wait to get it? What happens if there is a problem with the software? How is maintenance handled? At this phase, figure 5 shows the level that appears to dominate the portability issue.

Dominant factors during the Distribution and Maintenance Phase:

> *Level 6:*
> Institutional procedures, Copyright and ownership, Cost factors, Marketing and distribution, Maintenance, Local management

It can be seen at this level that the problems are primarily human and organizational. The same problems can also thwart the distribution of locally developed educational software. Cost and copyright again are central issues. There appears to be very little that the designer or developer can do to forestall portability barriers at this level.

Implementation. Assuming an adapted product does finally make it into a target classroom, the product may still be rejected by the individual teacher for reasons that relate to his or her own preferred way of organizing instruction in his or her classroom. Variables that are critical at the local implementation level also include those relative to the hardware available in the particular school, as can be seen in the following factors and levels that are predicted to be most influential during the implementation phase. Figure 6 visualizes these levels.

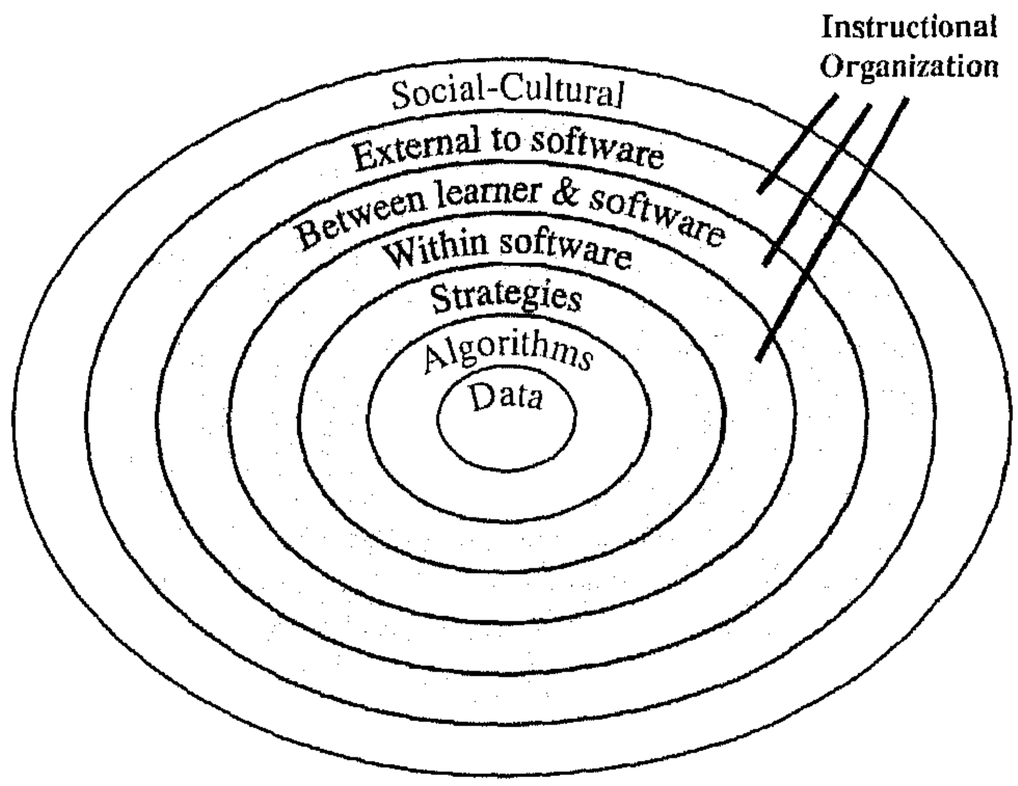

Figure 6. Factors most influential at the implementation phase.

Dominant factors during the Local Implementation Phase:

Level 3:
Operating system, Hardware characteristics, Instructional approach (microlevel)

Level 4:
Curriculum (specific), Instructional approach (general), Tone and style of interactions and communication

Level 5:
Social organization of the learning environment

Level 6:
Curriculum, Physical organization of the learning environment, Teacher training and experience

Also, the appropriateness of documentation can become a critical factor affecting implementation decisions, certainly at Levels 4 and 6.

It can be seen that many factors that looked as if they had been solved in earlier phases of the adaptation process can now reappear and effectively kill the final step of successful portability: the actual usage of a product in a setting different from that in which it was developed. Some of these factors can be addressed through software with options for teacher modification of the software. Others, like social organization of the classroom, can be addressed through accompanying software with better documentation and lesson ideas for teachers. But some factors are probably too integral to alter at the microlevel (e.g., software designed around a "discovery approach" philosophy for a teacher who emphasizes mastery learning). Some software will never be portable to some classrooms regardless of any anticipatory actions.

CONCLUSION

The last exercise has suggested one approach to how the factors and their interdependencies can be used to help focus considerations relative to software design and to decision making about software portability. The example, however, was only a suggestion, not a well developed exercise to test the usefulness of the factor approach. What are needed are what occurs throughout this Special Issue—careful and detailed considerations of the research questions and the factors from the perspectives of professionals with international experience in the portability of computer-related educational resources.

Contributors

Betty Collis and Italo DeDiana are members of the Division of Educational Instrumentation Technology at the University of Twente in The Netherlands. Dr. Collis works at the international level with the implementation and evaluation of computer use in education and training. Dr. DeDiana specializes in the methodology of the design and authoring of educational software. Both are jointly involved in a 4-year international research project investigating the problems and possibilities of educational software portability. (Address: B. A. Collis, Department of Education, University of Twente, Postbus 217, 7500 AE Enschede, The Netherlands; FAX: 31-53-356531.)

References

Bork, A. (1976). *Transferability of computer-based learning materials.* Report with limited circulation, University of California, Irvine.

Bork, A. (1990). International development of technology-based learning courses. *Journal of Research on Computing in Education, 23*(2), 173-183.

Collis, B. A., & De Diana, I. (1990). The portability of computer-related educational resources. *Journal of Research on Computing in Education, 23*(2), 147-159.

De Diana, I., & Collis, B. A. (1990, March). *Toward a model of educational software portability.* Unpublished report, Department of Education, University of Twente, Enschede, The Netherlands.

Kearsley, G. (1990). Designing educational software for international use. *Journal of Research on Computing in Education, 23*(2), 242-250.

Murray-Lasso,M. (1990). Social and cultural constraints on portability. *Journal of Research on Computing in Education, 23*(2), 252-271.

Nagtegaal, C. (1990). The POCO Project: A response to the portability dilemma. *Journal of Research on Computing in Education, 23*(2), 184-194.

Nielsen, J. (1986). A virtual protocol model for computer-human interaction. *International Journal of Man-Machine Studies, 24*, 301-312.

Oliveira, J. B. A. (1990). The economics of educational software portability. *Journal of Research on Computing in Education, 23*(2), 318-333.

Vraets, J. W. (1990). Software portability: The Ontario approach. In N. Estes, J. Heene, & D. Leclercq (Eds.), *Proceedings, The Seventh International Conference on Technology and Education* (pp. 578-580). Edinburgh: CEP Consultants.